IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2019, No. 1, pp. 330–394. DOI:10.13154/tosc.v2019.i1.330-394

PEIGEN – a Platform for Evaluation, Implementation, and Generation of S-boxes

Zhenzhen Bao[1,2], Jian Guo[1], San Ling[1] and Yu Sasaki[3]

[1] Division of Mathematical Sciences, School of Physical and Mathematical Sciences,Nanyang Technological University, Singapore, Singapore
{zzbao,guojian,lingsan}@ntu.edu.sg
[2] Strategic Centre for Research in Privacy-Preserving Technologies and Systems, Nanyang Technological University, Singapore, Singapore
[3] NTT Secure Platform Laboratories, 3-9-11, Midori-cho Musashino-shi, Tokyo 180-8585, Japan.
yu.sasaki.sk@hco.ntt.co.jp

Abstract. In this paper, a platform named PEIGEN is presented to evaluate security, find efficient software/hardware implementations, and generate cryptographic S-boxes. Continuously developed for decades, S-boxes are constantly evolving in terms of the design criteria for both security requirements and software/hardware performances. PEIGEN is aimed to be a platform covering a comprehensive check-list of design criteria of S-boxes appearing in the literature. To do so, the security requirements are first intensively surveyed, existing tools of S-boxes are then comprehensively compared, and finally our platform PEIGEN is presented. The survey part is aimed to be a systematic reference for the theoretical study of S-boxes. The platform is aimed to be an assistant tool for the experimental study and practical use of S-boxes. PEIGEN not only integrates most of the features in existing tools, but also equips with functionalities to evaluate new security-related properties, improves the efficiency of the search algorithms for optimized implementations in several aspects. With the help of this powerful platform, many interesting observations are made in-between the security notations, as well as on the S-boxes used in the existing symmetric-key cryptographic primitives. PEIGEN will become an open platform and welcomes contributions from all parties to help the community to facilitate the research and use of S-boxes.

Keywords: S-box · Survey · Design criteria · Implementation criteria · New platform

1 Introduction

The substitution-box, or *S-box* for short, is commonly used in the design of symmetric cryptography primitives to offer non-linearity. In general, an S-box is a nonlinear mapping defined on $\mathbb{F}_2^n \to \mathbb{F}_2^m$, which takes as input a value of n bits and outputs a value of m bits. For instance, the data encryption standard (DES) S-boxes are mappings with $(n = 6, m = 4)$ and the advanced encryption standard (AES) S-box is a permutation with $(n = m = 8)$. S-boxes have two key aspects: the security strength and the performance in software/hardware. In many cases, the two merits conflict, and hence trade-offs are made to fit the overall design's priorities.

The security requirements for S-boxes, depending on the design strategy of the overall primitive, evolve over time. Each security requirement corresponds to a goal of resisting some cryptographic attacks, *e.g.,* the high algebraic degree of an S-box can be used as arguments against algebraic attacks such as integral attacks. As new attacks are devised, new properties are added into the pool of S-box security requirements, *e.g.,* invariant

Received: 2018-11-23, Accepted: 2019-01-23, Published: 2019-03-08

subspace attacks [LAAZ11, GJN+16] against PRINTcipher [KLPR10] and Midori [BBI+15] triggered research/constraints on the combination of S-boxes and round constants, which did not exist before 2011. Meanwhile, not all security requirements have to be fulfilled by every design, since some attacks can be resisted by other design components, *e.g.,* round constants can be used to break symmetry and to resist slide attacks. Thus, there is a trade-off between the subset of the required security properties for S-boxes and the complexity of other components of the overall primitive.

The implementation aspects of S-boxes cover both software and hardware. There are several ways to implement S-boxes in software depending on the platform and resources available. One common method is the table-lookup, which pre-computes a Look-Up Table (LUT) by exhausting all possible input values and storing the outputs in a table. The S-box is subsequently performed by accessing the table, usually indexed by the input values. The LUT method is subject to cache-timing attacks [Ber05]. To resist them, there are (close to) *constant-time* implementation methods such as algebraic implementations and bitsliced implementations. Both express the S-box by its algebraic form, and hence the execution time is independent of the input value. The difference is that bitsliced implementations usually take multiple parallel inputs and process all simultaneously by utilizing long registers such as streaming SIMD extensions (SSE) registers, by which the relative performance per input can be significantly improved.

Depending on the target use, the most commonly known Integrated Circuits (ICs) are Application Specific IC (ASIC) and Field Programmable Gate Array (FPGA). The implementation and performance of S-boxes on either IC can be very different. On both types of IC, there are also different ways of optimization, which will result in very different performance. For example, on ASIC and FPGA, there is the so-called serialized implementation aiming for smallest area occupied, and the depth optimized implementation usually taking a relatively larger area but resulting in higher frequency and throughput.

To make a better trade-off between the security and performance, there is a line of research developing tools for finding the optimal implementations. Given an S-box, Osvik [Osv00] tried to find the software implementation optimized in terms of CPU cycles. Given an S-box, Guo *et al.* [GJN+16] proposed a method to find an implementation in ASIC optimized in terms of circuit depth. Given an S-box and the cost of each unit operation, Jean *et al.* [JPST17] built a tool named `LIGHTER` to find implementations with small area in ASIC or with small number of instructions in Software. More related works are listed in Table 5.

Our contributions. Constant efforts are made by our community to research S-boxes, in the hope of providing the best design choices to yield ciphers that are not only strong enough to resist attacks but also efficient to be of practical use. Such efforts include those made to remedy the situation after occasional bursts of striking attacks, and those made to seek the best possible instances to enrich the portfolio of design candidates. In this paper, we combine results achieved by these efforts. We first follow the line of research in developing design criteria for S-boxes. Rich results have been obtained by our community in this line. Some earlier results turned out to be fundamental theories, while some newly proposed notions need to be further developed. We try to provide a comprehensive exhibit. The main line of our exhibition follows the line of attacks. Many known results proposed or implied criteria for avoiding each type of attack. Some criteria can generally apply to any S-box based designs whereas others assume a particular type of round function structure. Both types are taken into account in this paper to form a comprehensive check list for designers. Besides the security, we consider the S-box criteria to be efficiently implemented. One aspect we also considered are equivalent classes for the S-box and the link between different classes. These will help the designers to identify the search space of the S-box that satisfies certain criteria. A huge amount of previous research on S-boxes makes our

exhibit quite long. However, integration of the many criteria appearing in the long history of S-box design will be useful for future S-box designers and researchers.

In the later part of this paper, we follow the line of research in developing tools for evaluating and implementing S-box. We start by comprehensively surveying the state-of-the-art in tool development. We found that different tools are written in different languages with different interfaces. Some of them are not publicly accessible. Each tool focuses on its target criteria, thus we need to use/implement multiple tools to evaluate multiple criteria, which is already non-trivial. In particular, security and implementation aspects are hard to consider simultaneously. This motivates us to present a platform named PEIGEN for evaluating the security properties, finding optimal implementations for given S-boxes, and generating all suitable S-boxes when the security and performance requirements are given.

PEIGEN is built upon existing tools. After making a comprehensive comparison, we decide to build PEIGEN using the implementation model of `LIGHTER` proposed by Jean *et al.* [JPST17]. More explicitly, PEIGEN was built on the basis of a sub-module of `LIGHTER` which is for finding software/hardware implementations that are good in terms of Bitslice Gate Complexity (BGC), Gate Equivalent Complexity (GEC), and Multiplicative Complexity (MC), for 4-bit bijective S-boxes. PEIGEN inherits all functionalities provided by `LIGHTER`, and at the same time, improves the search efficiency using algorithmic-level optimizations, *e.g.,* the composition and concatenation method and the pre-computation mechanism. Therefore, PEIGEN is more efficient for a large set of S-boxes. Besides supporting 4-bit S-boxes, PEIGEN supports n-bit S-boxes, where $3 \leq n \leq 8$ (but still only feasible for finding implementations for 3- and 4-bit S-boxes).

Most importantly, compared with `LIGHTER`, PEIGEN is more versatile: Besides finding implementations good in terms of `BGC`, `GEC`, and `MC`, PEIGEN can also find implementations good in terms of circuit depth (`Depth`); Apart from finding good implementations, PEIGEN can also evaluate security-related properties and identify equivalence relationships; Besides, PEIGEN can generate new S-boxes fulfilling given security-related and/or performance-related criteria. Here, we summarize the advantages and limitations of PEIGEN.

PEIGEN has rich functionalities which are of three aspects:

1. **Evaluation**: given a set of n-bit S-boxes ($3 \leq n \leq 8$ if not otherwise stated), PEIGEN evaluates most of their security-related properties (*e.g.,* Differential Distribution Tables (DDT), Boomerang Connectivity Tables (BCT), Linear Approximation Tables (LAT), Algebraic Normal Forms (ANF), Auto-Correlation Tables (ACT), Linear Structures (LS), (v, w)-linearity, a table representation of $\mathcal{V}_S(u)$ for all u indicating the appearance of monomials in the ANFs of $x \mapsto \pi_v(S(x))$ for $v \in \mathbb{F}_2^n$, and many detailed criteria related to these tables. Given n-bit S-boxes, PEIGEN evaluated their equivalence relations, including Permutation-XOR equivalence (PXE), linear Equivalence (LE), Affine Equivalence (AE). Besides, for a given 4-bit S-box, it can partition its AE-class into PXE-classes.

2. **Implementation**: given a set of n-bit S-boxes and the specific implementation configuration (available gates and costs for each gate), PEIGEN can generate implementations which are good in terms of Bitslice Gate Complexity (`BGC`), Gate Equivalent Complexity (`GEC`), Multiplicative Complexity (`MC`), and Depth Complexity (`Depth`).

3. **Generation**: given a set of criteria together with a set of S-boxes, PEIGEN filters out good S-boxes fulfilling the set of criteria; given merely a set of criteria, PEIGEN generates new S-boxes fulfilling the set of criteria (both security-related and implementation-related properties).

In addition to the many features, PEIGEN is developed with efficiency, expandability and compatibility in mind. It is very efficient when evaluating the security properties

even for a large set of n-bit S-boxes for $3 \leq n \leq 8$. When finding implementations and generating new n-bit S-boxes, it is more efficient than existing tools for $n = 3, 4$. Although, it only supports $3 \leq n \leq 8$-bit S-boxes, which is mainly because of the application of specific optimization tricks, it can be extended to support larger n.

The main limitation of Peigen is that, it supports but often is infeasible to find implementations and generating n-bit S-boxes for $5 \leq n \leq 8$, except for very lightweight 5-bit S-boxes.

Utilizing the platform, we make some interesting observations on the inter-link between some security and performance merits. Besides developing the platform, we provide algorithmic improvement over previous tools, which allows us to find better S-boxes or better S-box implementations than previous tools.

Roadmap. The rest of the paper is organized as follows. Section 2 gives the necessary preliminaries and notations used in this paper. Section 3 lists all possible design criteria including security and performance considerations. Section 4 presents our tool Peigen, followed by some evaluation results in Section 5. Finally, Section 6 discusses some open problems and concludes the paper. Some notations are postponed to the Appendix.

2 Preliminaries and Notations

2.1 Notations

We list some notations used in this section as follows. A complete list of notations can be found in Appendix A.

$\oplus, \bigoplus$ and $+, \sum$	To make a distinction, we use $\oplus, \bigoplus$ to represent *addition* and *summation* of $\mathbb{F}_2^n$, and use $+, \sum$ to represent addition and summation of $\mathbb{Z}$.
a	A *binary vector*, for $a \in \mathbb{F}_2^n$, $a = (a_1, a_2, \ldots, a_n)$ where $a_i \in \mathbb{F}_2$ is the *coordinate* of a with index i.
$\mathrm{wt}(a)$	The *Hamming weight (or simply, weight)* of a binary vector $a \in \mathbb{F}_2^n$, $\mathrm{wt}(a) \triangleq \sum_{i=1}^{n} a_i$.
$\mathrm{supp}(a)$	The *support* of a binary vector $a \in \mathbb{F}_2^n$ is the set of all labels i such that $a_i \neq 0$.
$a \cdot b$	The *inner product* of two binary vectors $a, b \in \mathbb{F}_2^n$, $a \cdot b \triangleq \bigoplus_{i=1}^{n} a_i \cdot b_i$.

2.2 Preliminaries

Boolean functions are very common and useful mathematical tools used to design and analyze symmetric-key cryptographic primitives. The cryptographic criteria measured on Boolean functions are closely related to criteria measured on symmetric-key primitives, especially for S-boxes. Thus, in the following sections, to introduce criteria on S-boxes, we usually first provide some definitions and criteria measurements on Boolean functions.

Boolean functions. A *Boolean function* in n binary variables maps from $\mathbb{F}_2^n$ into $\mathbb{F}_2$. There are 2^{2^n} n-bit Boolean functions in total. Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a Boolean function. It can be represented in several ways.

Directly, we can use truth tables. We sometimes use f to directly denote its *value*

vector [Can16], which is the vector corresponding to all values taken by f when we use the lexicographical order on the inputs. Since there are 2^n inputs, the value vector of f is in $\mathbb{F}_2^{2^n}$. The n-variate Boolean function f is said to be *balanced* if the Hamming weight of its value vector $\mathrm{wt}(f)$ equals 2^{n-1}, *i.e.,* the output is uniformly distributed. Balancedness is generally a basic requirement for Boolean functions used in cryptographic primitives.

Mathematically, we can use multi-variate polynomials. A conventional form to represent a Boolean function is the algebraic normal form.

Definition 1 (Algebraic Normal Form (ANF) of a Boolean function [Can16])**.** A Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ can be uniquely represented by an n-variate polynomial over $\mathbb{F}_2$, named the *algebraic normal form* of f:

$$f(x_1, \ldots, x_n) = \bigoplus_{u \in \mathbb{F}_2^n} \alpha_u \prod_{i=1}^{n} x_i^{u_i}, \text{ where } \alpha_u \in \mathbb{F}_2.$$

This representation is essentially an element of $\mathbb{F}_2[x_1, \cdots, x_n]/(x_1^2 \oplus x_1, \cdots, x_n^2 \oplus x_n)$.

To compute the ANF of a given value vector f, or to recover the value vector from its ANF, one can use the following transformation equations between these two representations:

$$\alpha_u = \bigoplus_{x \preceq u} f(x), \text{ and } f(x) = \bigoplus_{u \preceq x} \alpha_u,$$

where, x satisfies the relationship $x \preceq u$ if and only if $x_i \leq u_i$ for all $1 \leq i \leq n$. This transformation between the coefficients of the ANF and the value vector is essentially a kind of binary Möbius transform. Both directions can be computed using a divide-and-conquer butterfly algorithm whose implementation involves simple `AND`, `SHIFT`, and `XOR` operations and can be very fast [Car10b, Can16].

The most well-understood types of Boolean functions are the *linear Boolean functions* $x \mapsto \alpha \cdot x$, which can be denoted by φ_α for $\alpha \in \mathbb{F}_2^n$. Note that, the algebraic normal form of a linear Boolean function is $\varphi_\alpha(x_1, \ldots, x_n) = \bigoplus_{i=1}^n \alpha_i \cdot x_i$. Together with their complements, they form the set of all affine Boolean functions $A_n \triangleq \{\varphi_\alpha(x_1, \ldots, x_n) = \bigoplus_{i=1}^n \alpha_i \cdot x_i \oplus \alpha_0 \mid a_0, \ldots, a_n \in \mathbb{F}_2\}$ [CCCF01].

Vectorial Boolean Functions and S-boxes. In specifications of cryptographic algorithms, the S-box is usually specified by using a LUT (see Table 7), because from the implementation point of view, the size of the domain of an S-box is relatively small and the S-box usually operates on local groups of bits among the whole state, and is relatively complex to describe mathematically (unless intrinsically designed from mathematical primitives).

However, from the cryptanalysis point of view, mathematical descriptions are very important. Specifically, representation in ANF allows us to manipulate them and to deeply study their properties by using algebra theories.

Mathematically, an S-box mapping n bits to m bits can be described as *a vectorial Boolean function* in n input variables and with m output bits:

$$S : \mathbb{F}_2^n \to \mathbb{F}_2^m.$$

Similar to a Boolean function, a vectorial Boolean function can also be uniquely represented in its *algebraic normal form* (ANF), which is an n-variable polynomial representation. Different from the ANF of a Boolean function in which the coefficients are in $\mathbb{F}_2$, coefficients in this polynomial are in $\mathbb{F}_2^m$. Formally, we have

Definition 2 (Algebraic Normal Form (ANF) of a vectorial Boolean function [Car10b])**.** A vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ can be uniquely represented by an n-variable

polynomial, named the *algebraic normal form* of S:

$$S(x_1, \ldots, x_n) = \bigoplus_{u \in \mathbb{F}_2^n} \alpha_u \prod_{i=1}^{n} x_i^{u_i}, \text{ where } \alpha_u \in \mathbb{F}_2^m.$$

Conventionally, we have:

Definition 3 (Coordinates of S [Nyb94])**.** An S-box S with m output bits has m coordinates, denoted as S_{e_i} for $1 \leq i \leq m$, where S_{e_i} is the Boolean function in n binary variables and represents the i-th output bit of S:

$$S_{e_i} : \mathbb{F}_2^n \to \mathbb{F}_2,$$

where $\{e_i\}_{i<m}$ is the standard basis for $\mathbb{F}_2^m$.

Definition 4 (Components of S [Nyb94])**.** An S-box S with n input bits and m output bits has 2^m components, which are the linear combinations of its m coordinates, and can be denoted as S_λ for $\lambda \in \mathbb{F}_2^m$:

$$\begin{array}{rccc} S_\lambda : & \mathbb{F}_2^n & \to & \mathbb{F}_2 \\ & x & \mapsto & \lambda \cdot S(x) \end{array}$$

where $a \cdot b$ is the inner product of a and b, *i.e.,* $\bigoplus_{i=1}^{n} a_i \cdot b_i$. In particular, S_0 is the null function and the m coordinate functions S_{e_i} form the basis of the linear space containing all S_λ.

The terms *components* and *coordinates* of S are some-times confused (some studies use them synonymously). However, they need to be distinguished because some important properties of the S-box cannot be described merely by using the term coordinates, as shown by *e.g.,* [Nyb91, Nyb94].

See Figure 6 for an example of the ANFs of the coordinates and components of an S-box.

Similar to the balancedness of Boolean functions, the balancedness of vectorial Boolean functions is generally a basic requirement for them to be used in cryptographic primitives. A vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is said to be *balanced* if its outputs are uniformly distributed, or more precisely, it takes every value of $\mathbb{F}_2^m$ the same number 2^{n-m} of times. The balancedness of a vectorial Boolean function is characterized by the balancedness of its component functions:

Proposition 1 ([Car10b])**.** *A vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is balanced if and only if all its non-trivial component functions are balanced.*

The balanced vectorial Boolean functions mapping $\mathbb{F}_2^n$ to itself, namely the n-bit permutations, are of particular importance for the design of block ciphers and hash functions. This is because most S-boxes used in block ciphers are permutations.

3 S-box Design Criteria

In this section, we provide a check-list of S-box design criteria. This does not mean that a chosen S-box must fulfill all the criteria, but rather, when a designer trades some criteria for other benefits, the designer should carefully consider any undesired property caused by invalidating a design criterion and should check whether other components in the design could remedy the weakness without too much cost. The art of these trade-offs has been shown in several designs. A popular example is KECCAK, in which the non-linear

component seen as a 5-bit S-box is not strong if assessed using the criteria listed in Sect. 3.1 and 3.2 (*e.g.,*, differential uniformity, linearity, algebraic degree). However, assessed using the implementation criteria listed in Sect. 3.3, it is superior. We view this as follows: the design choice of KECCAK S-box trades some security for performance, while the carefully designed linear layer, the just right combination between different components, and the number of rounds remedy the weakness of the non-linear layer.

Design criteria for S-boxes can be extended into design criteria for round functions of the entire algorithm. It is usually feasible to apply the criteria for the S-box to a small number of rounds, *e.g.,* two rounds, which is useful to find potential weaknesses of the design.

Criteria we focus on are those for which simple[1] transformations on the S-boxes leave them invariant. For other criteria, we just list them and do not take them as general, because they can be easily changed under simple transformation (*e.g.,* the number of fixed-points, the number of nonlinear terms in their ANF).

3.1 General Security Criteria (common for all types of linear layer)

Criteria listed in this subsection are general for all cipher designs based on S-boxes. We say that these criteria are general because they are imposed to resist widely applicable attacks.

The invention of the two most powerful attacks – differential attack [BS90] and linear attack [Mat93] – imposes conventional criteria on the design of S-box (*e.g., differential uniformity* and *linearity* described in detail in the sequel). One can see, in the design rationale of most block ciphers and hash functions using S-boxes, the commonly and clearly listed criteria for S-boxes are differential uniformity and linearity (*e.g.,* Serpent [RA98], Luffa [DCSW08], PRESENT [BKL+07], PRINCE [BCG+12]). For these criteria, the designers have to concern and evaluate carefully even though one can make a little trade-off, *e.g.,* in KECCAK [BDPVA] and GIFT [BPP+17]. Usually, all other components of the cipher are linear, and linear computations propagate differences and correlations in a deterministic manner. The non-linear S-box is the only source of uncertainty for the propagation of differences and correlations. The feasibility of both differential and linear attacks mainly depends on the local statistical property of the S-boxes, which can be extended to the entire cipher and finally reveals relations among plaintexts, ciphertexts, and the fixed secret key. Thus, the primary design criteria for S-box are to provide resistance against differential and linear attacks.

Besides, non-statistical attacks, *e.g.,* algebraic attacks, also impose (general or detailed) criteria for the S-box design. Some criteria (*e.g.,* algebraic degree) are so general that they make sense to resist several attacks under different names; other criteria (*e.g.,* differential branch number) are relatively narrowed to be meaningful regarding some particular attacks on particular ciphers.

We list general criteria in this subsection and defer the description of special criteria for specific designs to the next subsection.

3.1.1 Resistance to Differential Attack

The differential attack [BS90] exploits the non-uniform distribution of the output differences when the inputs are chosen with a fixed difference. Although linear components can efficiently diffuse differences, they cannot help to reduce the non-uniformity regarding differences. Thus, a uniform differential distribution mainly comes from non-linear components.

[1] "simple" is in the sense that they are easy to be analyzed cryptographically, *e.g.,* XOR constant, bit-permutation, linear, and affine transformations.

Definition 5 ([Nyb91])**.** The *derivative function (or simply, derivative)* of a Boolean function f to the direction $a \in \mathbb{F}_2^n$ is defined as:

$$\begin{aligned} D_a f : \mathbb{F}_2^n &\to \mathbb{F}_2 \\ x &\mapsto f(x \oplus a) \oplus f(x),\ a \in \mathbb{F}_2^n \end{aligned}$$

The number of inputs satisfying the derivative equation $f(x \oplus a) \oplus f(x) = b$ of a Boolean function f at $a \in \mathbb{F}_2^n$ and $b \in \mathbb{F}_2$ is defined as:

$$\delta_f(a, b) \triangleq \#\{x \in \mathbb{F}_2^n \mid f(x) \oplus f(x \oplus a) = b\} = \#\{D_a f^{-1}(b)\},$$

where $D_a f^{-1}(b)$ means the set of preimages of b under the derivative function $D_a f$.

Definition 6 (Perfect nonlinear Boolean function [MS89])**.** A Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is perfect non-linear if for every nonzero vector $a \in \mathbb{F}_2^n$, $\delta_f(a, 0) = \delta_f(a, 1) = 2^{n-1}$.

Perfect nonlinearity implies an earlier design criterion for S-boxes, namely the strict avalanche criterion (SAC). SAC is essentially a diffusion criterion. It requires the S-box satisfies that, a change in a single input bit results in output changes with probability 1/2. Accordingly, SAC can be described as follows: for every vector $a \in \mathbb{F}_2^n$ and $\mathrm{wt}(a) = 1$, $\delta_f(a, 1) = 2^{n-1}$. Thus, perfect nonlinearity is a stronger requirement than SAC [MS89]. A generalized criterion on SAC and perfect nonlinearity criterion is the *propagation criterion of degree k* [PLL+90]. An n-variable Boolean function is said to satisfy the propagation criterion of degree k (denoted by PC(k)), if when any i ($1 \le i \le k$) bits of the input are changed, the output changes with probability 1/2. Thus, SAC is equivalent to PC(1), and perfect nonlinearity implies PC(k) for $1 \le k \le n$. A tool that can be used to test SAC and PC for a Boolean function is the *autocorrelation*.

Definition 7 (Autocorrelation)**.** The autocorrelation coefficient of a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ on $a \in \mathbb{F}_2^n$ is defined by

$$r_f(a) \triangleq \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)} (-1)^{f(x \oplus a)} = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x) \oplus f(x \oplus a)}.$$

Proposition 2 ([PLL+90])**.** *A Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ satisfies* PC(k) *if and only if*

$$r_f(a) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x) \oplus f(x \oplus a)} = 0 \quad \text{for } 1 \le \mathrm{wt}(a) \le k.$$

From these definitions and criteria on Boolean functions, we can extend to corresponding definitions and criteria on vectorial Boolean functions (*i.e.,* the S-boxes).

Definition 8 (Derivative of S [Nyb91])**.** For a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$, the derivative of S to the direction $a \in \mathbb{F}_2^n$ is defined as

$$\begin{aligned} D_a S : \quad \mathbb{F}_2^n &\to \mathbb{F}_2^m \\ x &\mapsto S(x) \oplus S(x \oplus a) \end{aligned}$$

The number of inputs satisfying the derivative equation $S(x) \oplus S(x \oplus a) = b$ of a vectorial Boolean function S at $a \in \mathbb{F}_2^n$ and $b \in \mathbb{F}_2^m$ is defined as [LP07]:

$$\delta_S(a, b) \triangleq \#\{x \in \mathbb{F}_2^n \mid S(x) \oplus S(x \oplus a) = b\} = |D_a S^{-1}(b)|.$$

If the number $\delta_S(a, b)$ is divided by the domain size, *i.e.,* $\frac{\delta_S(a,b)}{2^n}$, this ratio provides the probability that $S(x) \oplus S(x') = b$ if an input pair (x, x') is chosen uniformly from the

set of all input pairs with difference a. Thus, this ratio is widely used under the name "*differential probability*" of the difference propagation (a, b) in S. The values of $\delta_S(a, b)$ for all pairs $(a, b) \in \mathbb{F}_2^n \times \mathbb{F}_2^m$ can be arranged in a $2^n \times 2^m$ table, named the *differential distribution table* (DDT) of S, in which the element $\mathrm{DDT}(a, b)$ in row a and column b is equal to $\delta_S(a, b)$. For those zero elements in DDT, *i.e.,* $\delta_S(a, b) = 0$, the difference propagation (a, b) is called *invalid*, and input difference a and output difference b are said to be *incompatible* through S. If S is a permutation, then for the value $\delta_S(a, b)$ to be nonzero, either the input difference a and output difference b are both zero or nonzero.

Differential Uniformity. To get an impression on whether an S-box based cipher resists differential attacks, one can estimate the maximum expected differential probability (MEDP). A shortcut method to approximate MEDP is to compute the maximum differential probability of the S-box to the power of the minimum number of active S-boxes (with nonzero difference propagation through the S-box) in the differential propagation through the cipher. Thus, the *maximum differential probability* of the S-box is the most manifest feature to identify the goodness of an S-box regarding differential attacks. The corresponding maximum δ_S (multiply the maximum differential probability by 2^n) is named the differential uniformity (or uniformity for short) of an S-box. A well-known example of the utility of the differential uniformity of an S-box is the proof of security against the differential attack for AES: there are at least 25 active S-boxes for 4-round AES, and the differential uniformity of the AES S-box is $2^{-6} \times 2^8$. Thus, the probability of any differential trail can be upper bounded by using $2^{-6\times 25}$ [HLL+00, DR02]. The differential uniformity is formally defined as:

Definition 9 (Differential Uniformity [Nyb93])**.** Let S be a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$, the differential uniformity of S is defined as:

$$\mathcal{U}(S) \triangleq \max_{a \in \mathbb{F}_2^n \setminus \{0\}, b \in \mathbb{F}_2^m} \delta_S(a, b).$$

If $\mathcal{U}(S) \leq \delta$, S is called differentially δ-uniform.

We define $\mathcal{U}_{\mathrm{Freq}}(S) \triangleq \#\{(a, b) \mid \delta_S(a, b) = \mathcal{U}(S),\ a \in \mathbb{F}_2^n \setminus \{0\},\ b \in \mathbb{F}_2^m\}$ as the frequency; the number of occurrences in the DDT of an S-box.

Low differential uniformity is advantageous for S-boxes. Those S-boxes reaching the minimum possible differential uniformity are called – *perfect non-linear S-boxes.*

Definition 10 (Perfect Non-linear S-boxes [Nyb91])**.** A vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is perfect non-linear if for every $a \in \mathbb{F}_2^n \setminus \{0\}$, $\delta_S(a, b) = 2^{n-m}$ for all $b \in \mathbb{F}_2^m$.

Proposition 3 ([Nyb91])**.** *A vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is perfect non-linear if and only if all its non-trivial components are perfect non-linear in the sense of Definition 6.*

Proposition 4 ([Nyb91])**.** *For a perfect non-linear S-box $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$, $n \geq 2m$.*

Accordingly, there is no perfect non-linear permutation. However, it is possible to be a permutation (balanced) and at the same time be *almost perfect non-linear.*

Proposition 5 (Almost Perfect Non-linear S-boxes [NK92])**.** *Let S be a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^n$. Then, $\mathcal{U}(S) \geq 2$. Those n-bit S-boxes with $\mathcal{U}(S) = 2$, i.e., reach the minimum, are called almost perfect nonlinear (APN) functions.*

The existence of n-bit APN permutation is implied by the existence of n-bit almost Bent (AB) functions [CV94, Nyb93] when n is odd. However, when n is even, there is no general conclusion on the existence of an APN permutation. As noted by Leander and

Poschmann [LP07], there is no APN permutation on $\mathbb{F}_2^4$. For a 4-bit bijective S-box S, the optimal $\mathcal{U}(S)$ is 4. It was conjectured for a long time that no APN permutation could exist on $\mathbb{F}_2^n$ if n is even, until Dillon exhibited an APN permutation on $\mathbb{F}_2^6$ [BDMW10]. However, whether an APN permutation exists on $\mathbb{F}_2^n$ if n is even and $n \geq 8$ is still an open problem. Note that, as for 8-bit permutations, the AES S-box possesses the best *known* differential uniformity, which is $\mathcal{U}(S) = 4$.

Note that, differential uniformity is invariant under inversion and affine transformations [Nyb93]. The most general form of function equivalence that is known to preserve differential uniformity is CCZ-equivalence which is a more general notion than affine equivalence [CCZ98, CP18].

Differential Spectrum. Besides the maximum $\mathcal{U}$, the frequency of the maximum $\mathcal{U}_{\text{Freq}}$ (or the number of zero entries in DDT) also impacts resistance against differential attacks, *e.g.,* multiple differential attacks and impossible differential attacks. More importantly, the frequency in the DDT provides more accurate estimation of the maximum expected differential probability than that provided merely by the differential uniformity [PSLL03]. In other words, S-boxes that have the same differential uniformity but different differential spectra can perform differently in terms of resistance against differential attacks.

Thus, some design criteria impose restrictions on the differential spectra of the S-box. More formally, we have

Definition 11 (Differential Spectrum [BCC10, CR15])**.** The *differential spectrum* of a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is the multiset

$$\mathcal{D}_{\text{spec}}(S) \triangleq \{\delta_S(a, b) \mid a \in \mathbb{F}_2^n \setminus \{0\}, b \in \mathbb{F}_2^m\}.$$

Differential spectrum is invariant under affine transformation, so the upper bound on the MEDP computed by using a differential spectrum will be identical for affine equivalent S-boxes. However, affine equivalent S-boxes can be in-equivalent regarding the MEDP of the resulting cipher [CR15]. Thus, using information of the whole DDT of the S-box, one could more accurately estimate the MEDP of the cipher and thus more accurately evaluate the resistance against differential attacks.

This implies that the whole DDT of the S-box can be used in theoretically computing the MEDP. Since the traditional differential attack merely drives exploitable differential characteristics from the DDT of the S-box, two S-boxes with the same DDT (different S-boxes can have the same DDT) are regarded as providing the same level of resistance against differential attacks. Hence, it is reasonable to invent an equivalent relationship between S-boxes. These are the two newly proposed notions – DDT-equivalent and γ-equivalent on S-boxes [BCJS18]. The corresponding valuable work is to reconstruct the class of DDT-equivalent S-boxes from a given DDT [BCJS18, DH18].

Note that in some papers, the DDT of an S-box also embeds the information of correct input/output pairs supporting those differences. In fact, there are improved differential attacks exploiting the real value of the correct pairs instead of using merely the difference [SWW18]. That indicates that the approximation on MEDP obtained from information of the DDT of the S-box is only a lower bound.

See Figure 2 for an example of the DDT, $\mathcal{U}$, $\mathcal{D}_{\text{spec}}$ of a 4-bit S-box.

3.1.2 Resistance to Linear Attack

The linear attack [Mat93] exploits that for a cipher $E_K(P) = C$, there exists a linear combination on the bits of plaintexts P, bits of the ciphertexts C, and bits of a key K, which form a Boolean function that behaves non-randomly (unbalanced) on a random set of P. The linear combination is called a *linear approximation* of the cipher (denoted by a

triple (α, β, γ)), and is of the form:

$$\alpha \cdot P \oplus \beta \cdot C \oplus \gamma \cdot K,$$

where α is a linear mask on the input (plaintext), β is a linear mask on the output (ciphertext), γ is a linear mask on the round keys, and "$\cdot$" denotes the inner product. When we say "linear mask on X," we mean that the mask is a binary vector, selects the bits of X corresponding to its nonzero coordinates and sums (linearly combines) them together (thus, it is essentially a linear Boolean function). The non-randomness is measured by using the term *bias*, which is $|\Pr[\alpha \cdot P \oplus \beta \cdot C \oplus \gamma \cdot K = 0 \mid E_K(P) = C] - 1/2|$. If a function is linear, the bias is maximized, *i.e.,* 1/2. If a function is random (balanced), the bias is minimized, *i.e.,* 0. Thus, the bias is an indicator of the randomness of the linear approximation or say, distance from a linear function. It can be seen that linear components in a cipher cannot provide bias for any linear approximation. Non-linear components in a cipher are the source of biases for all possible linear approximations. As S-boxes are generally the only non-linear components in most S-box based ciphers, the biases of all linear approximations of the S-box are very important for resisting linear attack.

For an $n \times m$-bit S-box S, the linear approximations (denoted by the pair (α, β)) are of the form $\alpha \cdot x \oplus \beta \cdot S(x) = \alpha \cdot x \oplus S_\beta(x)$. The bias of a linear approximation (α, β) of S is then $\varepsilon_S(\alpha, \beta) = \left| \frac{\#\{x | S_\beta(x) = \alpha \cdot x\}}{2^n} - 1/2 \right|$. The values of $\varepsilon_S(\alpha, \beta)$ for all pairs (α, β) can be arranged in a $2^n \times 2^m$ table, named the *linear approximation table* (LAT) of S, in which the element $\mathrm{LAT}(\alpha, \beta)$ in row α and column β is equal to $\varepsilon_S(\alpha, \beta)$. As will be seen, the biases of all linear approximations of S correspond to the Walsh spectrum of S, in which the maximum absolute value is defined as the linearity of S. The linearity is thus a consistent notion with the maximum bias of all linear approximations of the S-boxes.

Note that linearity and nonlinearity of vectorial Boolean functions are notions studied by the community much earlier than the notion of biases of linear approximations. Earlier (before the invention of the well-known linear attack on DES), a variety of criteria were proposed to measure linearity and nonlinearity, in the common belief that measuring linearity and nonlinearity enables some confusion ability of an S-box to be quantified. The generally accepted criterion to measure nonlinearity of an S-box is defined as the Hamming distance between the set of all its non-trivial components to the set of all affine functions. Formally, we have the following definitions (beginning from definitions on Boolean functions, we go through to generalized definitions on vectorial Boolean function, *i.e.,* S-boxes).

Definition 12 ([MS89, PLL+90])**.** The *Hamming distance* between two Boolean functions f and g is the number of function values in which they differ. If we directly denote the value vectors by f and g, then $\mathrm{d}(f, g) \triangleq \mathrm{wt}(f \oplus g)$.

Definition 13 (Nonlinearity of a Boolean function [MS89, CCCF01])**.** The *nonlinearity* of a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is the minimum Hamming distance between f and all affine functions. Denote the set of all affine functions by $A(n) = \{\varphi_\alpha, \varphi_\alpha \oplus 1 \mid \varphi_\alpha : x \mapsto \alpha \cdot x,\ \alpha \in \mathbb{F}_2^n\}$, then we have,

$$\mathcal{NL}(f) \triangleq \min_{g \in A(n)} \mathrm{d}(f, g) = \min_{\alpha \in \mathbb{F}_2^n} |\mathrm{wt}(f \oplus \varphi_\alpha)| .$$

A very convenient tool to study nonlinearity and linearity is the Walsh transform, which is actually an essential tool for studying (vectorial) Boolean functions.

Definition 14 ([PLL+90, CCCF01])**.** The discrete *Fourier transform (a.k.a., Walsh transform) at point* 0 of the sign function $(-1)^f$ of a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is

denoted by

$$\mathcal{F}(f) \triangleq \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)}.$$

According to definitions, we have $\mathcal{F}(f) = 2^n - 2\,\mathrm{wt}(f)$.

Definition 15 ([PLL+90, CCCF01])**.** The *Walsh transform (aka., Fourier transform)* of a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is defined as:

$$\mathcal{W}_f(\alpha) \triangleq \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x) \oplus \alpha \cdot x}, \quad \alpha \in \mathbb{F}_2^n$$

According to definitions, we have $\mathcal{W}_f(\alpha) = \mathcal{F}(f \oplus \varphi_\alpha) = 2^n - 2\,\mathrm{d}(f, \varphi_\alpha)$.
The value taken by the transform at point α is called the *Walsh coefficient* of f at point α.

Definition 16 ([PLL+90, CCCF01])**.** The *Walsh spectrum (aka. Fourier spectrum)* of a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is the multi-set

$$\mathcal{W}_{\mathrm{spec}}(f) \triangleq \{\mathcal{W}_f(\alpha) \mid \alpha \in \mathbb{F}_2^n\}.$$

The *extended Walsh spectrum* of f is the multi-set of the absolute of the values in $\mathcal{W}_{\mathrm{spec}}(f)$.

Note that a (vectorial) Boolean function is completely specified by its Walsh coefficients. From the truth table, there is a fast algorithm to compute all Walsh coefficients [Can16].

The Walsh transform of a Boolean function satisfies a primary theorem:

Proposition 6 (Parseval's relation)**.** *For a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$, the Walsh transform of f satisfies:*

$$\bigoplus_{\alpha \in \mathbb{F}_2^n} \mathcal{W}_f(\alpha)^2 = 2^{2n}.$$

Definition 17 ([Dob94])**.** The *linearity* of a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is defined as

$$\mathcal{L}(f) \triangleq \max_{\alpha \in \mathbb{F}_2^n} |2^n - 2\,\mathrm{wt}(f \oplus \varphi_\alpha)| = \max_{\alpha \in \mathbb{F}_2^n} |\mathcal{W}_f(\alpha)|\,.$$

Accordingly, the nonlinearity and linearity of a Boolean function are related by:

$$\mathcal{NL}(f) = 2^{n-1} - \frac{1}{2} \max_{\alpha \in \mathbb{F}_2^n} |\mathcal{W}_f(\alpha)| = 2^{n-1} - \frac{1}{2}\,\mathcal{L}(f).$$

From the Parseval relation, we have the following.

Proposition 7. *For a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$, the linearity (resp. nonlinearity) of f satisfies:*

$$\mathcal{L}(f) \geq 2^{n/2}, \text{ and } \mathcal{NL}(f) \leq 2^{n-1} - 2^{n/2-1}.$$

The Boolean functions with linearity reaching the lower bound, *i.e.,* with the highest nonlinearity, are Bent functions.

Definition 18 (Bent Boolean function [MS89])**.** A Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is called Bent function if and only if for every $\alpha \in \mathbb{F}_2^n$, $|\,\mathcal{W}_f(\alpha)| = 2^{n/2}$.

As shown by Meier and Staffelbach [MS89], the class of perfect nonlinear Boolean functions coincides with the class of Bent Boolean functions. Note that, Bent Boolean functions only exist for *even* number of variables, they are not balanced, and their algebraic degree is always upper bounded by $n/2$.

The generalization of these definitions on a Boolean function to that of a vectorial Boolean function (S-box) is quite direct:

Definition 19 (Walsh transform and Walsh spectrum of an S-box [Car10b])**.** The *Walsh transform* of a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is defined as:

$$\mathcal{W}_S(\alpha, \beta) = \mathcal{W}_{S_\beta}(\alpha) = \sum_{x \in \mathbb{F}_2^n} (-1)^{\beta \cdot S(x) \oplus \alpha \cdot x}, \quad \alpha \in \mathbb{F}_2^n,\ \beta \in \mathbb{F}_2^m.$$

The value taken by the transform at point (α, β) is called the *Walsh coefficient* of S at point (α, β). The *Walsh spectrum* of S is the multiset

$$\mathcal{W}_{\text{spec}}(S) \triangleq \{\mathcal{W}_S(\alpha, \beta) \mid \alpha \in \mathbb{F}_2^n, \beta \in \mathbb{F}_2^m \setminus \{0\}\}.$$

The *extended Walsh spectrum* of S is the multi-set of the absolute of values in $\mathcal{W}_{\text{spec}}(S)$. The Walsh support of S is those (α, β) such that $\mathcal{W}(\alpha, \beta) \neq 0$.

Back to the notion of bias of linear approximations, we have:

$$\mathcal{W}_S(\alpha, \beta) = 2^{n+1} \cdot \varepsilon_S(\alpha, \beta).$$

Accordingly, the linear approximation table (LAT) has equivalence with the Walsh transform (up to multiplication of a constant 2^{n+1}). Thus, in this paper, we use *the table formed by all Walsh coefficients* $\mathcal{W}_S(\alpha, \beta)$ as the LAT instead of using the values of biases $\varepsilon_S(\alpha, \beta)$. Note that, besides the bias and Walsh coefficient, there is also a consistence notion of *correlation coefficient* $C(f, g)$ [DGV94], which associates a pair of Boolean functions f and g by $C(f, g) = 2\Pr[f(x) = g(x)] - 1$. Let one Boolean function be $f(x) = \alpha \cdot x$ which is a linear combination of input bits, and let the other be $g(x) = \beta \cdot S(x)$, which is a linear combination of output bits. Then the correlation coefficient $\mathcal{C}(\alpha \cdot x, \beta \cdot S(x))$ is related to $\mathcal{W}_S(\alpha, \beta)$ by

$$\mathcal{W}_S(\alpha, \beta) = 2^n \cdot \mathcal{C}(\alpha \cdot x, \beta \cdot S(x)).$$

The correlation coefficients $\mathcal{C}(\alpha \cdot x, \beta \cdot S(x))$ for all $\alpha \in \mathbb{F}_2^n$ and $\beta \in \mathbb{F}_2^m$ can be arranged in a matrix, named *correlation matrix* [DGV94], which is also equivalent to LAT (up to multiplication with a constant).

See Figure 3 for an example of the LAT, $\mathcal{L}$, $\mathcal{W}_{\text{spec}}$ of a 4-bit S-box.

Note that an S-box is completely specified by its Walsh Spectrum, *i.e.,* LAT. It can be recovered from its Walsh Spectrum:

Proposition 8 ([Per17])**.** *Let S be a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$. Then each coordinate S_{e_i} (for $1 \leq i \leq m$) can be recovered by using:*

$$S_{e_i}(x) = \frac{1}{2} - \frac{1}{2^{n+1}} \sum_{a \in \mathbb{F}_2^n} \mathcal{W}_S(a, 2^i)(-1)^{a \cdot x}.$$

Definition 20 (Linearity of an S-box [Nyb94])**.** The *linearity* of a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is the maximum linearity of its non-trivial components $\{S_\beta \mid \beta \in \mathbb{F}_2^m \setminus \{0\}\}$.

$$\mathcal{L}(S) = \max_{\lambda \in \mathbb{F}_2^m \setminus \{0\}} \mathcal{L}(S_\beta) = \max_{\alpha \in \mathbb{F}_2^n, \beta \in \mathbb{F}_2^m \setminus \{0\}} |\mathcal{W}_S(\alpha, \beta)|.$$

We define $\mathcal{L}_{\text{Freq}} \triangleq \#\{(\alpha, \beta) \mid \mathcal{W}_S(\alpha, \beta) = \mathcal{L}(S),\ \alpha \in \mathbb{F}_2^n,\ \beta \in \mathbb{F}_2^m \setminus \{0\}\}$ as the frequency of the maximum occurs in the LAT of an S-box.

Definition 21 (Nonlinearity of an S-box [Nyb94])**.** The *nonlinearity* of a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is the Hamming distance between the set of its non-trivial components $\{S_\lambda \mid \lambda \in \mathbb{F}_2^m \setminus \{0\}\}$ and the set of all affine functions A_n.

$$\mathcal{NL}(S) = \min_{\lambda \in \mathbb{F}_2^m \setminus \{0\}} \mathcal{NL}(S_\lambda) = 2^{n-1} - \frac{1}{2}\mathcal{L}(S).$$

High nonlinearity (low linearity) of an S-box is a desired property for the cipher to resist linear attack. Similar to the definitions of PN and APN functions related to the resistance to differential attacks, the corresponding definitions of Bent functions and almost Bent functions are closely related to the resistance to linear attacks (although the original proposition of the Bent function [Rot76] is not related to linear attacks).

Definition 22 (Bent vectorial Boolean function)**.** A vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is called a Bent function if and only if all of its non-trivial component functions are Bent. This is equivalent to $|\mathcal{W}_S(\alpha, \beta)| = 2^{n/2}$ for all $\alpha \in \mathbb{F}_2^n$ and $\beta \in \mathbb{F}_2^m \setminus \{0\}$.

Bent vectorial Boolean functions with linearity (resp. nonlinearity) reach the minimum (resp. maximum). However, as mentioned above, Bent functions are not balanced and only exist for an even numbers of variables. Similar to the proposition of APN, to achieve balancedness and at the same time to possess some merit of Bent functions, the almost Bent function is proposed:

Definition 23 (Almost Bent vectorial Boolean function [CV94])**.** Let S be a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^n$. Then,

$$\mathcal{L}(S) \geq 2^{(n+1)/2}, \text{ and } \mathcal{NL}(f) \leq 2^{n-1} - 2^{(n+1)/2-1}.$$

Those S with linearity (resp. nonlinearity) reaches the lower (resp. upper) bound are called Almost Bent (AB) functions. Moreover, an almost Bent (AB) function is almost perfect nonlinear (APN) as well.

Almost Bent functions only exist for *odd* number of variables. Examples are the power polynomials $S(x) = x^{2^k+1}$ in $\mathbb{F}_2^n$, n is odd and $1 < k < n$ and $\gcd(n, k) = 1$, which was proposed by Nyberg [Nyb93]. For *even* number of variables, the linearity of the S-boxes are all strictly larger than $2^{(n+1)/2}$. However, the tight lower bound on the linearity is not known. As noted by Leander and Poschmann [LP07], for 4-bit bijective S-box S, the optimal linearity $\mathcal{L}(S) = 8$. Brinkmann and Leander [BL08] provided all the four non-AE classes of 5-bit APN permutations with $\mathcal{L}(S) = 8$. For 8-bit permutations, the AES S-box possesses the best *known* linearity, which is $\mathcal{L}(S) = 32$.

At this point, we can distinguish those S-boxes possessing the optimal differential uniformity and optimal linearity as follows:

Definition 24 ([LP07])**.** Let S be a vectorial Boolean function $S : \mathbb{F}_2^4 \to \mathbb{F}_2^4$. If S fulfills the following conditions, we call S an optimal 4-bit S-box:

1. S is a bijection.
2. $\mathcal{L}(S) = 8$.
3. $\mathcal{U}(S) = 4$.

Following the definition on an optimal 4-bit S-box, we can call an S-box from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$ an optimal n-bit S-box if it possesses the optimal differential uniformity, possesses the optimal linearity, and is bijective. However, we currently do not even known the exact value of the optimal differential uniformity and/or the optimal linearity for $n \geq 8$.

3.1.3 Resistance to Boomerang Attack

The boomerang attack [Wag99] intuitively combines two independent differential trails in two consecutive bijective functions. Let E_0 and E_1 be two consecutive functions that allow a differential propagation α to β with probability p and γ to δ with probability q. The boomerang attack detects a certain differential propagation for $E_1 \circ E_0$ that occurs with probability p^2q^2.

It has been pointed out several times in the literature that the behavior around the border of two trails is not completely independent from the other part. Dunkelman *et al.* [DKS10] formulated its probability during their attack against KASUMI by setting the middle part E_m. That is, the target is divided into three parts $E_1 \circ E_m \circ E_0$ to point out that the probability for the middle part is not squared. Cid *et al.* [CHP+18] focused on the designs in which E_m is a single S-box-application layer and proposed constructing a precomputed table to evaluate this probability. The table resembles DDT, and Cid *et al.* revealed several relationships between DDT and their table called the Boomerang Connectivity Table (BCT), which is defined as follows.

Definition 25 (Boomerang Connectivity Table (BCT) of an invertible $n \times n$ S-box S [CHP+18])**.** Let $a \in \mathbb{F}_2^n$ and $b \in \mathbb{F}_2^n$ be the output difference from E_0 (equivalently input difference to the S-box) and input difference to E_1 (equivalently output difference from the S-box), respectively. BCT is a $2^n \times 2^n$ table that precomputes the following quantity for all (a, b): $\beta_S(a,b) \triangleq \#\left\{x \in \mathbb{F}_2^n \mid S^{-1}\big(S(x) \oplus b\big) \oplus S^{-1}\big(S(x \oplus a) \oplus b\big) = a\right\}$.

The *boomerang uniformity*, denoted by $\mathcal{BU}(S)$, is the highest value in the BCT excluding the entry $(0,0)$: $\mathcal{BU}(S) = \max_{a,b \in \mathbb{F}_2^n \setminus \{0\}} \beta_S(a,b)$. The *boomerang differential spectrum* is the multiset $\mathcal{BD}_{\text{spec}}(S) \triangleq \{\beta_S(a,b) \mid a \in \mathbb{F}_2^n \setminus \{0\}, b \in \mathbb{F}_2^n\}$.

See Figure 4 for an example of the BCT, $\mathcal{BU}$, $\mathcal{BD}_{\text{spec}}$ of a 4-bit S-box.

Dunkelman [Dun18] provided a method to efficiently construct the BCT. Boura and Canteaut [BC18] provided an in-depth analysis to show that two families of differentially 4-uniform S-boxes are also optimal with respect to boomerang attacks.

3.1.4 Resistance to Algebraic Attacks

The rule of thumb to evaluate the resistance of a cipher to algebraic attacks is to calculate the algebraic degree. Generally speaking, the higher the algebraic degree, the higher the resistance to algebraic attacks. For ciphers based on S-boxes, the algebraic degree of the S-box provides an upper bound of the algebraic degree of the whole cipher.

Algebraic Degrees.

Definition 26 (Algebraic degree of a Boolean function $\deg(f)$)**.** For a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$, let

$$\text{ANF}_f = \bigoplus_{u \in \mathbb{F}_2^n} \alpha_u \prod_{i=0}^{n-1} x_i^{u_i}, \text{ where } \alpha_u \in \mathbb{F}_2, \text{ then}$$

$$\deg(f) \triangleq \max\{\text{wt}(u) \mid u \in \mathbb{F}_2^n \text{ and } \alpha_u \neq 0 \in \mathbb{F}_2 \text{ in } \text{ANF}_f\}.$$

The direct generalization of the algebraic degree of a Boolean function to the algebraic degree of a vectorial Boolean function is as follows:

Definition 27 (Algebraic degree of a vectorial Boolean function $\text{Deg}(S)$)**.** For a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$, let

$$\text{ANF}_S = \bigoplus_{u \in \mathbb{F}_2^n} \alpha_u \prod_{i=0}^{n-1} x_i^{u_i}, \text{ where } \alpha_u \in \mathbb{F}_2^m, \text{ then}$$

$$\text{Deg}(S) \triangleq \max\{\text{wt}(u) \mid u \in \mathbb{F}_2^n \text{ and } \alpha_u \neq 0 \in \mathbb{F}_2^m \text{ in } \text{ANF}_S\}.$$

That is, the *algebraic degree* of a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is defined as the global degree of its ANF. It equals the maximum among all degrees of the coordinate functions (and also equals the maximum among all degrees of the component functions):

$$\text{Deg}(S) = \max_{i \in \{0,\cdots,n-1\}} \deg(S_{e_i}) = \max_{\lambda \in \mathbb{F}_2^m \setminus \{0\}} \deg(S_\lambda).$$

We define $\mathrm{Deg}_{\mathrm{Freq}} \triangleq \#\{\lambda \mid \deg(S_\lambda) = \mathrm{Deg}(S),\ \lambda \in \mathbb{F}_2^m \setminus \{0\}\}$ as the number of non-trivial components of S with the maximal degree.

Apart from the maximum degree, the minimum is also important regarding algebraic attacks. Here we denote by $\min\deg(S)$ the *minimum degree* of S,

$$\min\deg(S) \triangleq \min_{\lambda \in \mathbb{F}_2^m \setminus \{0\}} \deg(S_\lambda).$$

Note that, different from the notion of algebraic degree, the minimum among all degrees of the coordinate functions does not equal the minimum among all degrees of the component functions. This is because even when all coordinates possess the maximum degree, some of their linear combinations (components) can be with smaller degrees (in the case that the highest order terms in their ANFs are canceled when summed together). Thus, we distinguish the following two definitions, *i.e.,* $\mathrm{Deg}_{\mathrm{spec}}(S)$ and $\mathrm{Deg}_{\mathrm{spec}_{cor}}(S)$.

Definition 28 ($\mathrm{Deg}_{\mathrm{spec}}(S)$)**.** The *degree spectrum* of a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is a multiset $\mathrm{Deg}_{\mathrm{spec}}(S) \triangleq \{\deg(S_\lambda) \mid \lambda \in \mathbb{F}_2^m \setminus \{0\}\}$, where S_λ are component functions of S.

Definition 29 ($\mathrm{Deg}_{\mathrm{spec}_{cor}}(S)$)**.** The *degree spectrum of the coordinates* of a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is a multiset $\mathrm{Deg}_{\mathrm{spec}_{cor}}(S) \triangleq \{\deg(S_{e_i}) \mid 1 \le i \le m\}$, where S_{e_i} are coordinate functions of S.

See Figure 6 for an example of the ANFs, $\mathrm{Deg}(S)$, $\min\deg(S)$ and $\mathrm{Deg}_{\mathrm{spec}}(S)$ of a S-box.

Besides the linear combinations of coordinates, the product of coordinates is also important. The degrees of the product of any k coordinates of the S-box (and its inverse) can provide a tighter upper bound on the degree of the entire cipher, as stated and shown by using examples (Luffa, AES, Keccak, JH etc.) in [BCC11, BC13b].

Definition 30 (Maximal degree of the product of k coordinates)**.** Let S be a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$. For any integer k, $1 \le k \le m$, $d_k(S)$ denotes the maximal algebraic degree of the product of any k (or fewer) coordinates of S

$$d_k(S) = \max_{K \subseteq \{1,\ldots,m\}, |K| \le k} \deg\Big(\prod_{i \in K} S_{e_i}\Big).$$

In particular, $d_1(S) = \deg(S)$.

See Table 8 for an example of the d_k of a 4-bit S-box.

Theorem 1 (Degree of the composition $G \circ F$ [BCC11, BC13b])**.** *Let $F : \mathbb{F}_2^{nt} \to \mathbb{F}_2^{nt}$ corresponding to the concatenation of t smaller balanced S-boxes, $S_1, \ldots, S_t$, defined over $\mathbb{F}_2^n$. Let d_k be the maximal degree of the product of any k coordinates of anyone of these smaller S-boxes. Then, for any function G from $\mathbb{F}_2^{nt}$ into $\mathbb{F}_2^\ell$, we have*

$$\deg(G \circ F) \le nt - \frac{nt - \deg(G)}{\gamma}, \quad \text{where } \gamma = \max_{1 \le i \le n-1} \frac{n-i}{n - \max_{1 \le j \le t} d_i(S_j)}.$$

From this theorem, it can be seen that the degree of product of coordinates of the S-boxes provides upper bounds on the maximum degree of a composition function. Thus, $d_k(S)$ are closely related to resistance against higher-order differential attacks.

The maximum value the degree $d_k(S)$ can take is limited by the divisibility of the Walsh spectrum of S.

Theorem 2 ([CV02])**.** *If all $\mathcal{W}_{S_v}(u)$ are divisible by 2^ℓ, then the product of any k coordinates of S has degree $d_k \le n + k - \ell$.*

From this theorem, one can conclude that Bent functions (resp. almost Bent functions) on $\mathbb{F}_2^n$ have degrees at most $n/2$ (resp. $(n+1)/2$).

Apart from the influence on the algebraic degree of the whole cipher, d_k of the S-box (and more directly, d_k of the inverse S-box) is also closely related to the division property of the S-box.

Division Property. The division property [Tod15b] is a notion proposed in 2015. Let d be the algebraic degree of the S-box. Then, it is possible to choose 2^{d+1} inputs to the S-box so that the sum of the S-box outputs can be 0. This evaluation is natural in algebraic analysis, whereas simple integral attacks do not take it into account. Intuitively, the division property combines degree counting with the integral attack.

Let $\pi_u : \mathbb{F}_2^n \to \mathbb{F}_2$ be a bit-product function for any $u \in \mathbb{F}_2^u$, and let $x \in \mathbb{F}_2^n$ be an input to π_u. We also denote the i-th bits of u and x by u_i and x_i respectively. Then, π_u is defined as $\pi_u(x) := \Pi_{i=1}^n x_i^{u_i}$.

Definition 31 (Division Property)**.** Let $\mathbb{X}$ be a multiset whose elements take a value of $\mathbb{F}_2^n$, and k takes a value between 0 and n. When the multiset $\mathbb{X}$ has the division property $\mathcal{D}_k^n$, it fulfills the following conditions. The parity of $\pi_u(x)$ for all $x \in \mathbb{X}$ is always even for any u whose Hamming weight is less than k. Moreover, the parity becomes unknown for any u whose Hamming weight is greater than or equal to k.

After the strike of the division-property-based integral attack on MISTY [Tod15a], Boura and Canteaut [BC16] and Gölöglu *et al.* [GRW16] discussed the security criterion for S-boxes related to resistance against division-property-based integral attacks. In particular, one suggestion was to focus on the appearance of monomials in the ANFs of $x \mapsto \pi_v(S(x))$ for $v \in \mathbb{F}_2^n$, which is defined as a set

$$\mathcal{V}_S(u) \triangleq \bigcup_{w \in \mathtt{Succ}(u)} V_S(w) \quad \text{and} \quad V_S(w) \triangleq \{v \in \mathbb{F}_2^n : \pi_v(S(x)) \text{ contains } \pi_w(x)\},$$

where $\mathtt{Succ}(u) = \{x \in \mathbb{F}_2^n : u \preceq x\}$ which is an affine subspace of dimension $(n - \mathrm{wt}(u))$.

A table representation of $\mathcal{V}_S(u)$ for all u is useful to understand the resistance against division-property-based attacks. Such a table is recommended to not contain columns or rows that are too sparse. See Figure 5 for an example of a table representation of $\mathcal{V}_S(u)$ for all u of an S-box.

A strong indicator for the number of elements of weight 1 in $\mathcal{V}_S(u)$ is its $\min \deg(S)$ as stated by the following proposition:

Proposition 9 ([BC16])**.** *Let S be a permutation of $\mathbb{F}_2^n$ such that all its non-trivial component functions S_λ for $\lambda \in \mathbb{F}_2^n$ have the maximal degree $(n-1)$. Then, for any $u \in \mathbb{F}_2^n$, $\mathcal{V}_S(u)$ contains at least $(n - \mathrm{wt}(u))$ elements of weight 1.*

Accordingly, to resist division-property-based integral attacks, it is better that all components have the maximal degree.

Univariate Degree. Besides the ANF representation, an S-box can also be uniquely represented as a univariate polynomial:

Definition 32 (Univariate polynomial representation)**.** Let $S : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be any n-bit S-box. The vectors of $\mathbb{F}_2^n$ can be interpreted as elements of a finite field $\mathbb{F}_{2^n}$, and S can be written as a unique univariate polynomial of $\mathbb{F}_{2^n}[X]$:

$$S(X) = \sum_{i=0}^{2^n-1} v_i X^i.$$

This representation can be obtained by using Lagrange interpolation [MLCA], thus it is also named an *interpolation polynomial*.

Definition 33 (Univariate degree)**.** The univariate degree of an n-bit S-box $S : X \mapsto \sum_{i=0}^{2^n-1} v_i X^i$ is

$$\max(\{i, v_i \neq 0\}).$$

The relation between the univariate degree and the algebraic degree of an S-box is given by Canteaut [Can16] as $\mathrm{Deg}(S) = \max(\{\mathrm{wt}(i), v_i \neq 0\})$.

The univariate degree of the S-box indicates the resistance to interpolation attacks for the cipher. If the univariate degree of the S-box is too low, the distance of the S-box to the set of low univariate degree functions is too small, or the number of terms in the polynomial representation is too small, it may lead to efficient interpolation attacks [JK01].

3.1.5 Resistance to Truncated Differential and Subspace Trail Attacks

Linear structures were proposed in the early era of cryptanalysis of block ciphers. In the seminal attack they presented [CE85], Chaum and Evertse first found linear structures in individual rounds of DES. These linear structures are six-bit blocks that when are xor-ed to the input of an S-box, the output is always changed by the same value. Then, by chaining these linear structures that yield a sequence of linear factors over more rounds, attackers are able to attack DES up to six rounds. This linear structure-based attack can be seen as the predecessor of the well known differential attack on DES [BS90], because the later can be seen as the probabilistic version of the former. Formally, we have:

Definition 34 (Linear structures of a Boolean function [Eve87, MS89])**.** The *linear space* of a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is the linear subspace of those a such that $D_a f$ is a constant function, *i.e.,*

$$\mathrm{LS}(f) \triangleq \{a \in \mathbb{F}_2^n \mid D_a f = c, \text{ where } c \text{ is constant in } \mathbb{F}_2\}.$$

Such a, $a \neq 0$, is said to be a *c-linear structure* of f.

Definition 35 (Linear structures of an S-box [Eve87, Lai94, Dub01])**.** A *linear structure* of a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is a triple (λ, a, c) such that a is a c-linear structure of the *component function* $S_\lambda(x)$, *i.e.,*

$$S_\lambda(x) \oplus S_\lambda(x \oplus a) = c \text{ for } \forall x \in \mathbb{F}_2^n.$$

Let $\#\,\mathrm{LS}$ denote the number of linear structures of an S-box.

See Table 9 for examples of linear structures of some 4-bit S-boxes. It shows that for optimal 4-bit S-boxes, the degree spectrum has some relationship with the number of linear structures.

A special type of linear structure was proposed named *undisturbed bits* (the invariant bits in all compatible output differences corresponding to a particular nonzero input difference) [MT14], which is the linear structure of coordinates instead of components of the S-box. Noticing the relation between undisturbed bits and linear structures, as well as their relations to the derivative and autocorrelation of coordinates of the S-box, Makarim and Tezcan [MT14] proposed a way to efficiently find all linear structures of an S-box by using its *autocorrelation table* (ACT) (see Thm. 3). The ACT of an S-box $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is a $2^n \times 2^m$ matrix, in which the element $\mathrm{ACT}_S(a, \lambda)$ in row a and column λ is equal to the autocorrelation coefficient of the component function S_λ on a, *i.e.,* $r_{S_\lambda}(a)$ [ZZI00].

Theorem 3 ([MT14])**.** *A vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ has a linear structure (λ, a, c) if and only if there exists $a \in \mathbb{F}_2^n \backslash \{0\}$ and $\lambda \in \mathbb{F}_2^m \backslash \{0\}$ such that $|\mathrm{ACT}_S(a, \lambda)| = 2^n$. If $\mathrm{ACT}_S(a, \lambda) = +2^n$ (resp. -2^n), $c = 0$ (resp. $c = 1$).*

The linear structure (λ, a, c) of an S-box can be interpreted in another way: for a particular input difference a, one bit of information on its compatible output differences is determined. The determined bit of information is on the parity of bits selected by λ from the output differences. From this interpretation, the undisturbed bits can be seen as truncated differential (with probability 1), *i.e.,* differences that are only partially determined [Knu94]. Accordingly, Makarim and Tezcan [MT14] proposed that ACT can be viewed as a counterpart of the DDT for truncated differential cryptanalysis. Exploiting the undisturbed bits, Tezcan *et al.* proposed several of the best known truncated and impossible differential attacks on some symmetric ciphers [TTD14, Tez14, Tez16]. However, they pointed out that it remains unclear that whether the existence of linear structures in component functions of an S-box other than the coordinate functions could be exploited to improve truncated differential attacks.

Recently, Grassi *et al.* [GRR16] introduced the subspace trail attack, which is a generalization of the invariant subspace attack [LAAZ11]. They pointed out that a subspace trail attack includes special cases of techniques based on impossible or truncated differentials and integrals. Leander *et al.* [LTW18], by considering all linear structures of components of an S-box instead of restricting to that of the coordinates (*i.e.,* undisturbed bit), were able to describe the influence of linear structures on subspace trails: If the S-box used in the cipher does not have any linear structure, ignoring the details of the S-box and using their proposed approach can result in the strongest subspace trail. If the S-box used in the cipher does have linear structures, one should take the linear structures into account to find the strongest subspace trail or to provably bound the longest subspace trail.

Interestingly, the distance to linear structures (the set of all Boolean functions with a linear structure) was already used (as another way apart from the distance to affine functions) to measure the nonlinearity of Boolean functions in earlier eras:

Definition 36 (Distance to linear structures [MS89])**.** Let $\mathrm{LS}(n)$ denote the subset of Boolean functions having linear structures. For a Boolean function f, the distance to linear structures is defined as the Hamming distance of f to the set $\mathrm{LS}(n)$:

$$\mathrm{d}(f, \mathrm{LS}(n)) \triangleq \min_{\ell \in \mathrm{LS}(n)} \mathrm{d}(f, \ell).$$

If the nonlinearity is measured by using the distance to all affine functions (which is the generally used measurement, see Def. 13 and 21), even if a function has a linear structure, it can still have high nonlinearity. Thus, the distance to a linear structure is a stricter criterion on nonlinearity than the distance to affine function.

3.1.6 Resistance to Cube or Cube-like Attacks

Boura and Canteaut [BC13a] proposed a notion to quantify the ability of an S-box to propagate the affine relations in the input to the output, named (v, w)-linearity. An S-box is (v, w)-linear means that 2^w components are affine on all cosets of a v-dimensional subspace. If an S-box is (v, w)-linear and v and w are large, it means that by fixing only a small number of input bits to arbitrary values, a large number of the output bits becomes linearly dependent on the remaining input bits. In this case, this S-box might not be strong enough to resist a cube attack [DS09] or cube-like attack [Fuh10]. Formally,

Definition 37 ((v, w)-linearity [BC13a])**.** A vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is said to be (v, w)-linear if there exist linear subspaces $V \subset \mathbb{F}_2^n$ and $W \subset \mathbb{F}_2^m$ with $\dim V = v$ and $\dim W = w$, such that, for all $\lambda \in W$, S_λ has a degree at most 1 on all cosets of V.

Note that (v, w)-linear implies (v', w')-linear for all $v' \leq v$ and $w' \leq w$. We denote by $\max_v(v, w)$-linear the maximal v such that the S-box is (v, w)-linear, and $\max_w(v, w)$-linear the maximal w such that the S-box is (v, w)-linear.

The notion (v, w)-linearity is shown to be related to two well-known cryptographic properties — the algebraic degree and the linearity.

Proposition 10 ((v, w)-linear and degree [BC13a])**.** *Let S be an vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$. If S is (v, w)-linear w.r.t. (V, W), then all its components S_λ, where $\lambda \in W$ have degree at most $n + 1 - v$. Thus,* $\mathrm{Deg}(S) \leq n + 1 - v$.

Proposition 11 ((v, w)-linear and linearity [BC13a])**.** *Let S be a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$. If S is (v, w)-linear w.r.t. (V, W), then all its components S_λ, where $\lambda \in W$, have linearity $\mathcal{L}(S_\lambda) = 2^v$. Thus, $\mathcal{L}(S) \geq 2^v$.*

Note that these two propositions are not under necessary and sufficient conditions. Although the converse propositions of these propositions are not necessarily true, the contrapositives of these propositions provide assurance on the non-(v, w)-linearities for an S-box if its degree is larger than $n + 1 - v$ or if its linearity is smaller than v. For example, one can conclude that:

Proposition 12. *Let S be a permutation of $\mathbb{F}_2^n$ such that all its non-trivial component functions S_λ for $\lambda \in \mathbb{F}_2^n$ have the maximal degree $(n - 1)$. Then S is not (v, w)-linear for all $v \geq 3$ and $w \geq 1$.*

Note that, if a 4-bit S-box is not (v, w)-linear for all $v \geq 3$ and $w \geq 1$, this will help the whole cipher to resist against the cube-like attacks in [Fuh10, BC13a].

Furthermore, for $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ and $m = n$, Boura and Canteaut point out that:

Proposition 13 ([BC13a])**.** *Let S be a permutation of $\mathbb{F}_2^n$ such that all its non-trivial component functions S_λ for $\lambda \in \mathbb{F}_2^n$ have the maximal degree $(n - 1)$. Then, S is not $(2, n - 1)$-linear.*

For a Boolean function, there is a necessary and sufficient condition between $(n - 1, 1)$-linear and the degree and linearity.

Proposition 14 ($(n - 1, 1)$-linear [BC13a])**.** *Let f be a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$. Then f is $(n - 1, 1)$-linear if and only if* $\deg(f) \leq 2$ *and $\mathcal{L}(f) \geq 2^{n-1}$. Moreover, if* $\deg(f) = 2$ *and $\mathcal{L}(f) \geq 2^{n-1}$, there exist exactly three distinct hyperplanes H such that f has degree at most 1 on both H and $\bar{H}$.*

This proposition implies that if any components S_λ in an S-box S satisfy $\deg(S_\lambda) \leq 2$ and $\mathcal{L}(S_\lambda) \geq 2^{n-1}$, then the S-box is $(n - 1, 1)$-linear. Moreover, if those components are linear on cosets of the same subspace V with $\dim(V) = n - 1$, and they form a subspace W with $\dim(W) = w$, and the S-box is $(n - 1, w)$-linear, which is not desired.

As pointed by Boura and Canteaut [BC13a] and Liu and Rijmen [LR18], the notion of (v, w)-linearity seems also related to the invariant subspace attack [LAAZ11] that may exploit the property of an n-bit S-box that there *exists* an affine subspace $a + V$ with dimensions no larger than n such that the image under the S-box also forms an affine subspace $b + W$. Resistance to invariant subspace attacks can be achieved together with the choices of round constants [GJN+16, BCLR17] to avoid iterative subspaces.

The (v, w)-linearity and the concrete number of (V, W) pairs with respect to which an S-box is (v, w)-linear are invariant under affine transformation. However, unlike the differential uniformity and linearity, they are not invariant under CCZ-equivalence.

See Table 10 for an example of (v, w)-linearity of an S-box.

3.1.7 Hash Function Settings

The discussion so far mainly considered the security in the keyed setting, in which the adversary can only choose plaintexts and ciphertexts to force oracles to output the

corresponding ciphertexts and plaintexts. In contrast, in the key-less setting like the analysis of hash functions, the adversary can choose the internal state values and thus a different type of security should be considered.

Rebound attacks [MRST09, LMS+15] aim to build efficient differential trails for the sequence of the P-layer, S-layer and another P-layer. The attacks choose the difference between the first P-layer and the S-layer so that the propagation through the inverse of the P-layer has a good property. The attacks then independently choose the difference between the S-layer and the second P-layer so that the propagation through the P-layer has a good property. As a result, the attacks expect that the independently generated pair of input and output differences for the S-layer have actual values to satisfy those differences for all the S-boxes.

The notion of cardinality for DDT is useful to evaluate the difficulty of this strategy.

Definition 38. $\mathrm{CardD}(S) \triangleq \#\mathrm{SetDiff}(S)$, *i.e.,* the number of non-zero entries in DDT, where $\mathrm{SetDiff}(S) \triangleq \{(a,b) \mid |\delta_S(a,b)| \neq 0\}$, *i.e.,* the set of non-zero entries in (or say, support of) DDT.

If $\mathrm{CardD}(S)$ is low, it becomes hard to generate such differences suitable for the rebound attacks.

3.1.8 Others

There are other important cryptographic properties of (vectorial) Boolean functions focused on more in the research of stream ciphers. Examples include resiliency and algebraic immunity, for which one has to make trade-offs with algebraic degree and nonlinearity. As these notions are less relevant in the design of S-boxes used in block ciphers and hash functions, we refer interested readers to the work of Carlet [Car10a, Car10b] for a comprehensive introduction.

3.2 Special Security Criteria (for some specific linear layers)

In this section, we list security criteria that are useful for a particular type of (yet important) linear layers.

3.2.1 For Linear Layers only Composed of Bit-Permutation

The first type of linear layers we consider is the bit-permutation. Suppose that all bits of the state are updated by parallelly applying the S-box in the S-layer. Then, performing a bit-permutation can be sufficient for the diffusion. PRESENT, RECTANGLE, and GIFT are examples using this type of linear layers. More generally, the designs that are optimized for bitsliced implementation, *e.g.,* Serpent and LS-design, often share the same security criteria.

Resistance to Differential and Linear Attacks. The same spirit of differential and linear attacks against general diffusion layers can still be applied to those with specific linear layers. Hence, we do not define their concept from scratch, but focus our attention on the important part of the previous security criteria.

Given that no bit-permutation changes the number of active bits in the context of either differential or linear cryptanalysis, S-boxes in those designs must provide not only confusion but also diffusion. In other words, S-boxes should activate many bits. Thus, the branch number of the S-box defined below is crucial for those designs.

Definition 39 ($\mathcal{BN}_{\mathrm{D}}(S)$)**.** The *differential branch number* of an S-box S (a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$), $\mathcal{BN}_{\mathrm{D}}(S) = \min\{\mathrm{wt}(a) + \mathrm{wt}(b) \mid \delta_S(a,b) \neq 0,\ a \in \mathbb{F}_2^n \setminus \{0\},\ b \in \mathbb{F}_2^m\}$.

Definition 40 ($\mathcal{BN}_{\mathrm{L}}(S)$)**.** The *linear branch number* of an S-box S (a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$), $\mathcal{BN}_{\mathrm{L}}(S) = \min\{\mathrm{wt}(u) + \mathrm{wt}(v) \mid \mathcal{W}_S(u,v) \neq 0,\ u \in \mathbb{F}_2^n,\ v \in \mathbb{F}_2^m \setminus \{0\}\}$.

If the branch number of the S-box is 2, the differential/linear trail can only have 1 active bit throughout the encryption/decryption process. This motivates us to focus on single-bit differences/linear masks of DDT and LAT.

Definition 41 ($\mathrm{DDT}_1(S)$)**.** The sub-table of DDT containing entries (a,b) where $\mathrm{wt}(a) = \mathrm{wt}(b) = 1$.

Definition 42 ($\mathrm{LAT}_1(S)$)**.** The sub-table of LAT containing entries (u,v) where $\mathrm{wt}(u) = \mathrm{wt}(v) = 1$.

The highest value in $\mathrm{DDT}_1(S)$ and $\mathrm{LAT}_1(S)$ is defined as follows.

Definition 43. $\mathcal{U}_1(S) \triangleq \max_{a \in \mathbb{F}_2^n \setminus \{0\}, b \in \mathbb{F}_2^m} \{\delta(a,b) \mid \mathrm{wt}(a) = \mathrm{wt}(b) = 1\}$.

Definition 44. $\mathcal{L}_1(S) \triangleq \max_{\alpha \in \mathbb{F}_2^n, \lambda \in \mathbb{F}_2^m \setminus \{0\}} \{|\mathcal{W}_S(\alpha,\lambda)| \mid \mathrm{wt}(\alpha) = \mathrm{wt}(\lambda) = 1\}$.

See Figure 2 and 3 for examples of DDT_1, $\mathcal{U}_1$, LAT_1 and $\mathcal{L}_1$ of a 4-bit S-box.

The 4-bit S-boxes of SERPENT [RA98] and PRESENT [BKL+07] are examples whose design criteria involve not only $\mathcal{U}$ and $\mathcal{L}$ but also $\mathcal{U}_1$ and $\mathcal{L}_1$. Essentially, they are optimal on these four criteria because bijective 4-bit S-boxes can achieve the following: $\mathcal{U} = 4$, $\mathcal{L} = 8$, $\mathcal{U}_1 = 0$, $\mathcal{L}_1 = 4$. The optimality of the former two criteria is shown by Leander and Poschmann [LP07] as mentioned in Sect. 3.1, and the optimality of $\mathcal{L}_1$ can be implied by Proposition 15.

Besides the maximum values in DDT_1 and LAT_1, later attacks (*e.g.,* [Ohk09, Cho10]) show that the number of nonzero entries in DDT_1 and LAT_1 should also be considered. This is because a large number of nonzero entries increases the possibility that many single-bit trails result in hulls. That motivated Zhang *et al.* [ZBRL15] to consider the following definitions:

Definition 45. $\mathrm{SetDiff1}(S) \triangleq \{(a,b) \mid |\delta_S(a,b)| \neq 0,\ \mathrm{wt}(a) = \mathrm{wt}(b) = 1\}$, *i.e.,* the set of non-zero entries in (or say, support of) DDT_1. $\mathrm{CardD1}(S) \triangleq \#\,\mathrm{SetDiff1}(S)$, *i.e.,* the number of non-zero entries in DDT_1.

Definition 46. $\mathrm{SetLin1}(S) \triangleq \{(u,v) \mid |\mathcal{W}(u,v)| \neq 0,\ \mathrm{wt}(u) = \mathrm{wt}(v) = 1\}$, *i.e.,* the set of non-zero entries in (or say, support of) LAT_1. $\mathrm{CardL1}(S) \triangleq \#\,\mathrm{SetLin1}(S)$, *i.e.,* the number of non-zero entries in LAT_1.

On these definitions, Zhang *et al.* [ZBRL15] identify three platinum categories among optimal 4-bit S-boxes, which fulfill $\mathrm{CardD1} + \mathrm{CardL1} \leq 4$. Under the naming rule (CardD1, CardL1)-Num1-DL, they are called (0, 4)-Num1-DL, (1, 3)-Num1-DL, and (2, 2)-Num1-DL. The 4-bit S-box of RECTANGLE [ZBL+15] is selected under this criteria and belongs to the (2, 2)-Num1-DL category. RECTANGLE is shown to have good resistance to differential and linear attacks even considering the possibility of trail clustering.

Note that there is no optimal 4-bit S-box with $\mathcal{BN}_{\mathrm{L}} > 2$ according to the following proposition:

Proposition 15 ([ZBRL15])**.** *Let $S : \mathbb{F}_2^4 \to \mathbb{F}_2^4$ be an optimal 4-bit S-box, then* $\mathrm{CardL1}(S) \geq 2$. *This implies* $\mathcal{L}_1(S) \geq 4$.

Although ensuring optimal $\mathcal{U}$ and $\mathcal{U}_1$, optimal $\mathcal{L}$ and $\mathcal{L}_1$, optimal $\mathcal{BN}_{\mathrm{D}}$ and $\mathcal{BN}_{\mathrm{L}}$ is good for those designs, it does not exclude the possibility of designing secure primitives with non-optimality. Indeed GIFT [BPP+17], the latest design in this direction, ensures the primitive's security even using an S-box with $\mathcal{U}(S) = 6$ and $\mathcal{BN}_{\mathrm{D}}(S) = 2$, which are not optimal. The core idea is

- allowing only a few (≤ 2) differential propagations satisfying $\delta_S(a,b) > 4$.
- for differential propagations satisfying $\delta_S(a,b) > 4$, their number of active bits should be large, *i.e.,* $\mathrm{wt}(a) + \mathrm{wt}(b) \geq 4$.
- designing a bit-permutation such that an active output bit from any (both differential and linear mask) propagation (a,b) satisfying $\mathrm{wt}(a) + \mathrm{wt}(b) = 2$ does not move to an active input bit to any propagation (a',b') satisfying $\mathrm{wt}(a') + \mathrm{wt}(b') = 2$.

The idea can be examined quantitatively by evaluating the following properties.

Definition 47 (GI / GO / BI / BO(S))**.** GI / GO / BI / BO(S) are abbreviations of "Good Input", "Good Output", "Bad Input", "Bad Output", respectively.

- $\mathrm{GI}(S) \triangleq \{a \mid |\delta_S(a,b)| = 0,\ \mathrm{wt}(a) = \mathrm{wt}(b) = 1\}$, namely a set of the single-bit input difference to S such that the entries in DDT_1 is 0 for any output difference.
- $\mathrm{GO}(S) \triangleq \{b \mid |\delta_S(a,b)| = 0,\ \mathrm{wt}(a) = \mathrm{wt}(b) = 1\}$, namely a set of the single-bit output difference to S such that the entries in DDT_1 is 0 for any input difference.
- $\mathrm{BI}(S) \triangleq \{a \mid |\exists b, \delta_S(a,b)| \neq 0,\ \mathrm{wt}(a) = \mathrm{wt}(b) = 1\}$.
- $\mathrm{BO}(S) \triangleq \{b \mid |\exists a, \delta_S(a,b)| \neq 0,\ \mathrm{wt}(a) = \mathrm{wt}(b) = 1\}$.

Definition 48 (Dscore(S))**.** $|\,\mathrm{GI}\,| + |\,\mathrm{GO}\,|$ observed from DDT_1.

Definition 49 (Lscore(S))**.** $|\,\mathrm{GI}\,| + |\,\mathrm{GO}\,|$ observed from LAT_1.

Those are useful evaluation criteria when the permutation layer is designed.

3.2.2 For Linear Layers Using Binary Orthogonal Matrices

Another type of linear layer we consider is the combination of cell-wise permutation and multiplication by a binary orthogonal matrix. In such a construction, security against differential and linear attacks is identical because of the duality of those two attacks [CV94], which eases the designer's evaluation workload. SCREAM is an example of adopting this design. Some almost-maximum distance seperable (MDS) binary matrices are also orthogonal and thus fall into this type, *e.g.,* MIDORI64.

Resistance to Non-linear Invariant Subspace Attack. Todo *et al.* [TLS16] proposed a new type of attack using non-linear masks for the S-box. In particular, most 4-bit S-boxes and even some 8-bit S-box have non-linear masks in which the masked value does not change before and after the S-box. This is called non-linear invariant, which is defined as follows.

Definition 50. Non-linear invariants [TLS16] For a given S-box S, a non-linear Boolean function g is non-linear invariant if $g(x) \oplus g(S(x)) = c$ for any input value x, where c is a constant in $\mathbb{F}_2$.

See Figure 7 for examples of nonlinear invariants of two 4-bit S-boxes.

Todo *et al.* proved that if the linear transformation consists of cell-wise permutation and multiplications by binary orthogonal matrices and if there is a quadratic invariant for the S-box, $\oplus_{i=1}^{t} g(\mathbf{x_i})$ is non-linear invariant for the entire cipher. Thus, for ciphers with a binary orthogonal linear function, the number of quadratic invariants for the S-box might be a relevant criterion.

Todo *et al.* also introduced two quantities: $n_{\mathrm{Circ}(S)}$ and $\mathrm{Circ}(S)$. $n_{\mathrm{Circ}(S)}$ denotes the number of cycles in the functional graph representation of S and $\mathrm{Circ}(S)$ denotes the maximum cycle length in the functional graph representation of S. These are used to explain some relationships between the non-linear invariant attack and the invariant subspace attack.

3.3 Implementation Criteria

Efficiency of the implementation is as important as the security for the S-box. These two aspects of criteria form a trade-off and are often impossible to both be satisfied at a good level. Moreover, there are several evaluation criteria for efficiency, which forms another level of trade-off. For example, parallel implementation minimizes the latency of the hardware implementation while requiring a large area. In contrast, serial implementation minimizes the area size while imposing a larger latency. In this paper, we particularly focus on the following implementation criteria.

Gate Size. The gate size measures the area occupied by the logic circuit in hardware. Area is one of the most fundamental quantities to evaluate the hardware implementation cost of the S-box. Since the available gates and the area cost of different gates depend on technologies, *e.g.,* UMC/180nm and TMSC/65nm, measuring and comparing the area consumption of implementations requires a standard unit. A usual unit is Gate Equivalent (GE), where one GE equals the area of a 2-input `NAND` gate. The area of other gates can be normalized by using the ratio between their real area and the area of one `NAND` gate (see Table 2 and 3 for details). Accordingly, the area of a whole implementation involving various gates can also be normalized to be the number of GEs. Using the terms of GE, we can define the gate complexity of an S-box.

Note that, for software implementations, there is also a notion for gate complexity, named Bitslice Gate Complexity (`BGC`) [Sto16]. Under this notion, each available gate costs the same. Thus, BGC is essentially the smallest number of logic operations (available in an instruction set, *e.g.,* the most common set of logic instructions {`AND`, `OR`, `XOR`, `NOT`}) required to implement an S-box. Considering that `ANDN` is also common in various central processing units (CPUs), in this paper, we include it in the instruction set for software implementations. One should note that, in some studies (*e.g.,* [Sto16]), the difference between the notion of gate complexity and bitslice gate complexity only lies in the basic logic operations available in the set (apart from logical gates allowed in `BGC`, *i.e.,* {`AND`, `OR`, `XOR`, `NOT`}, for `GC`, additional gates are allowed, *i.e.,* `NAND`, `NOR`, `XNOR`), without considering the fact that different gates require different amounts of area in hardware. Here, we treat the notion of gate complexity in terms of GE, and denoted the notion by `GEC` in the formal definition. Thus, the notion is different from that of Stoffelen [Sto16] and is the same as that of Jean *et al.* [JPST17]. To distinguish these two notions, we denote the notion of gate complexity without considering GE by `GC`.

Definition 51 (Gate Equivalent complexity (`GEC`) [JPST17])**.** The smallest number of Gate Equivalents (GEs) required to implement an S-box, given the cost of atomic operations, *e.g.,* Table 3.

Definition 52 (Bitslice Gate Complexity (`BGC`) [CHM11, Sto16])**.** The smallest number of operations in {`AND`, `OR`, `XOR`, `NOT`} required to implement an S-box.

Depth. Depth of the circuit implementation of an S-box is closely related to latency and energy consumption of the circuit. In some studies, circuit depth complexity is defined as the length of the longest paths from an input gate to an output gate [BP12, Sto16]. However, counting the number of gates to estimate the delay of a circuit path is not accurate, because different gates delay differently.

The designers of the MIDORI block cipher [BBI+15] extensively studied the depth of S-boxes. They suggested roughly evaluating the depth of the Boolean function by weighing the logical operations in accordance with simple rules.

Definition 53 (Depth complexity (`Depth`) [BBI+15])**.** The `Depth` is defined as the sum of sequential path delays of basic operations (see Table 2) in the critical path .

Table 2: List of atomic operations implemented by standard cells from the libraries, recall that $\wedge, \vee, \oplus, \neg$ respectively stand for: logical and, or, exclusive or, not. [JPST17])

Operation	Function	Operation	Function
NAND	$(a,b) \to \neg(a \wedge b)$	XOR	$(a,b) \to a \oplus b$
NOR	$(a,b) \to \neg(a \vee b)$	XNOR	$(a,b) \to \neg(a \oplus b)$
AND	$(a,b) \to a \wedge b$	NAND3	$(a,b,c) \to \neg(a \wedge b \wedge c)$
OR	$(a,b) \to a \vee b$	NOR3	$(a,b,c) \to \neg(a \vee b \vee c)$
NOT	$a \to \neg a$	ANDN	$(a,b) \to \neg a \wedge b$
MAOI1	$(a,b,c,d) \to \neg((a \wedge b) \vee (\neg(c \vee d)))$	ORN	$(a,b) \to \neg a \vee b$
MOAI1	$(a,b,c,d) \to \neg((a \vee b) \wedge (\neg(c \wedge d)))$		

Note that $\texttt{MAOI1}(a,b,a,b) = \neg((a \wedge b) \vee (\neg(a \vee b))) = (\neg a \vee \neg b) \wedge (a \vee b) = \texttt{XOR}(a,b)$
Note that $\texttt{MOAI1}(a,b,a,b) = \neg((a \vee b) \wedge (\neg(a \wedge b))) = (\neg a \vee b) \wedge (a \vee \neg b) = \texttt{XNOR}(a,b)$

Table 3: Cost of atomic operations under various techniques (some are referred from [JPST17])

Tech.	NAND NOR	AND OR	NOT	XOR	XNOR	ANDN	ORN	NAND3 NOR3	MAOI1	MOAI1
UMC 180nm	1.00	1.33	0.67	3.00	3.00	1.67	1.67	1.33	2.67	2.00
TSMC 65nm	1.00	1.50	0.50	3.00	3.00	1.50	1.50	1.50	2.50	2.50
Software	-	1.00	1.00	1.00	-	1.00	-	-	-	-
Depth (GEs)	1.00	1.50	0.50	2.00	2.00	-	-	-	-	-
Depth (Soft.)	1.00	1.00	1.00	1.00	1.00	-	-	-	-	-
Multiplicative	-	1.00	0.00	0.00	-	-	-	-	-	-

It is reasonable to assume that depths of basic operations equal their GEs, because delays depend on the number of the transistors to be sequentially proceeded in the operation [BBI+15]. Note that the depth of critical path changes under different technologies, *e.g.,* hardware TSMC65nm, hardware UMC180nm, and software (see Table 3).

Multiplicative Complexity. The designers of LowMC [ARS+15] studied symmetric-key primitives that minimize the multiplicative size and depth of their descriptions.

Definition 54 (Multiplicative complexity (MC) [BPP00])**.** The multiplicative complexity $\texttt{MC}(f_1, f_2, \ldots, f_m)$ of a set of Boolean function $f_1, \ldots, f_r \in B_n$ is the smallest integer t for which there exist Boolean functions $g_i, h_i, k_i \in B_n (i = 1, \ldots, t)$ such that $h_1, k_1 \in \langle x_1, \ldots, x_n, 1\rangle, g_1 = h_1 k_1$ and $h_i, k_i \in \langle g_1, \ldots, g_{i-1}, x_1, \ldots, x_n, 1\rangle, g_i = h_i k_i$ for $i = 2, \ldots, t$. $f_1, \ldots, f_r \in \langle g_1, \ldots, g_t, x_1, \ldots, x_n, 1\rangle$. This recursion describes an XOR-AND circuit that has $x_1, \ldots, x_n$ as its inputs and outputs $f_1, \ldots, f_r$. The value t is the minimum number of AND gates necessary.

The cost of the masking for side-channel analysis countermeasure also depends on the MC. Moreover, as researched by several papers e.g. [PRC12, JS17], the cost of the higher-order masking grows quadratically to MC.

3.4 Invariant Properties under Simple Transformations

Many cryptographic properties (differential uniformity, linearity, differential spectrum, extended Walsh spectrum, algebraic degree, (v,w)-linearity, etc.) are invariant under simple transformations. These transformations include XOR constants, bit permutations at the input/output, linear transformation, and affine transformation.

Definition 55 (XOR-equivalent (XE))**.** Two functions $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ and $G : \mathbb{F}_2^n \to \mathbb{F}_2^m$ are XOR-equivalent if there exist two constants $c_1 \in \mathbb{F}_2^n$ and $c_2 \in \mathbb{F}_2^m$, s.t.

$$G(x) = F(x \oplus c_1) \oplus c_2.$$

Table 4: Known function equivalence that preserves particular criteria

Criteria	Equivalence	Criteria	Equivalence	Criteria	Equivalence
$\mathcal{U}$, $\mathcal{D}_{\text{spec}}$	CCZ [CP18]	$\mathcal{L}$, $\mathcal{W}_{\text{spec}}$	CCZ [CP18]	Deg, Deg_{spec}	EA [CP18]
$\mathcal{U}_1$, $\mathcal{D}_{\text{spec}_1}$	PXE (obvious)	$\mathcal{L}_1$, $\mathcal{W}_{\text{spec}_1}$	PXE (obvious)	$\text{Deg}_{\text{spec}_{cor}}$	PXE (obvious)
d_k	AE [GRW16]	# LS	AE [MS89]	$(v,w)-$linearities	AE [BC13a]
`MC`	EA (obvious)	`BGC/GEC`	PE (obvious)	`Depth`	PE (obvious)

Definition 56 (Permutation-equivalent (PE))**.** Two functions $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ and $G : \mathbb{F}_2^n \to \mathbb{F}_2^m$ are Permutation-equivalent, if there exist two bit permutations $P_1 : \mathbb{F}_2^n \to \mathbb{F}_2^n$ and $P_2 : \mathbb{F}_2^m \to \mathbb{F}_2^m$, s.t.

$$G(x) = (P_2 \circ F \circ P_1)(x).$$

Definition 57 (Permutation-XOR-equivalent (PXE))**.** Two functions $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ and $G : \mathbb{F}_2^n \to \mathbb{F}_2^m$ are Permutation-XOR-equivalent if there exist two bit permutations $P_1 : \mathbb{F}_2^n \to \mathbb{F}_2^n$ and $P_2 : \mathbb{F}_2^m \to \mathbb{F}_2^m$ and two constants $c_1 \in \mathbb{F}_2^n$ and $c_2 \in \mathbb{F}_2^m$, s.t.

$$G(x) = (P_2 \circ F \circ P_1)(x \oplus c_1) \oplus c_2.$$

Definition 58 (Linear-equivalent (LE))**.** Two functions $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ and $G : \mathbb{F}_2^n \to \mathbb{F}_2^m$ are Linear-equivalent if there exist two linear permutations $L_1 : \mathbb{F}_2^n \to \mathbb{F}_2^n$ and $L_2 : \mathbb{F}_2^m \to \mathbb{F}_2^m$, s.t.

$$G(x) = (L_2 \circ F \circ L_1)(x).$$

Definition 59 (Affine-equivalent (AE))**.** Two functions $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ and $G : \mathbb{F}_2^n \to \mathbb{F}_2^m$ are Affine-equivalent if there exist two affine permutations $A_1 : \mathbb{F}_2^n \to \mathbb{F}_2^n$ and $A_2 : \mathbb{F}_2^m \to \mathbb{F}_2^m$, s.t.

$$G(x) = (A_2 \circ F \circ A_1)(x).$$

Definition 60 (Extended-Affine equivalent (EA))**.** Two functions $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ and $G : \mathbb{F}_2^n \to \mathbb{F}_2^m$ are Extended-Affine equivalent if there exist two affine permutations $A_1 : \mathbb{F}_2^n \to \mathbb{F}_2^n$ and $A_2 : \mathbb{F}_2^m \to \mathbb{F}_2^m$ and an affine function $C : \mathbb{F}_2^n \to \mathbb{F}_2^m$, s.t.

$$G(x) = (A_2 \circ F \circ A_1)(x) \oplus C(x).$$

Definition 61 (Carlet-Charpin-Zinoviev equivalent (CCZ) [CCZ98])**.** Two functions $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ and $G : \mathbb{F}_2^n \to \mathbb{F}_2^m$ are CCZ equivalent if there exists an affine permutation A of $\mathbb{F}_2^n \times \mathbb{F}_2^m$, s.t., the graph of F is mapped to the graph of G, i.e.,

$$\{(x, F(x)) \mid x \in \mathbb{F}_2^n\} \xrightarrow{A} \{(x, G(x)) \mid x \in \mathbb{F}_2^n\}.$$

These different notions of equivalence have the following implication relations:

$$\begin{array}{ccccccccc} \text{PE} & \Longrightarrow & \text{LE} & & & & & & \\ & \searrow & & \searrow & & & & & \\ \text{XE} & \Longrightarrow & \text{PXE} & \Longrightarrow & \text{AE} & \Longrightarrow & \text{EA} & \Longrightarrow & \text{CCZ} \end{array}$$

Knowledge on the largest transformation group leaving a criterion invariant is important for studying the criterion and S-boxes fulfilling the criterion. Table 4 lists the most general form of function equivalence that is known to preserve certain criteria.

Classifying S-boxes according to the specific equivalence helps to better understand S-boxes with important security characteristics. For example, to investigate bijective 4-bit S-boxes with the optimal differential uniformity and linearity, Leander and Poschmann [LP07] partition the space of 4-bit permutations according to affine equivalence. The offered seminal knowledge is that there are exactly *16* different optimal affine-equivalent classes

(AE-classes for short) ($G_0 \sim G_{15}$, see Table 11), which contain all S-boxes possessing the best possible differential uniformity $\mathcal{U}(S) = 4$ and linearity $\mathcal{L}(S) = 8$ for 4-bit bijective S-boxes [LP07].By that, one could characterize all the optimal S-boxes regarding differential uniformity and linearity by using representatives of the AE-classes. Further, these 16 optimal AE-classes can be classified into 7 different CCZ equivalent classes (CCZ-classes for short)[2]. The corresponding knowledge on 5-bit S-boxes offered by Brinkmann and Leander [BL08] is that there are exactly 5 APN permutations in $\mathbb{F}_2^5$ up to affine equivalence. By further calculating, one will find 4 out of the 5 APN S-boxes possess optimal linearity, *i.e.,* $\mathcal{L}(S) = 8$. Accordingly, there are exactly 4 different AE-classes, which contain all S-boxes possessing the best possible differential uniformity $\mathcal{U}(S) = 2$ and linearity $\mathcal{L}(S) = 8$ for 5-bit bijective S-boxes.

In addition to cryptographic properties, notions of equivalence are also important for studying implementation complexities of S-boxes. For example, the multiplicative complexity is a constant within an affine-equivalent class. The gate complexity is a constant within a permutation-equivalent class. Utilizing these equivalences, one could handle the task to find implementations for all small S-boxes, study classes of S-boxes with the best implementation complexities, and reduce the search space when finding optimal implementations of a given S-box [UDCI+11, BNN+12].

By using the equivalence to study S-boxes, the prerequisite is to be able to test the equivalence between given S-boxes. However, testing equivalence is not a trivial problem. Thanks to Biryukov *et al.* [BCBP03], testing linear equivalence between n-bit permutations has become practical for n up to 32 (with theoretical complexity $O(n^3 2^n)$) and affine equivalences for n up to 17 (with theoretical complexity $O(n^3 2^{2n})$). This scale has covered the domain size of most S-boxes suitable to be used in modern symmetric primitives following classical design strategies. For larger n up to 40, one can resort to the algorithm recently published by Dinur [Din18]. Brinkmann and Leander [BL08] offer techniques to test CCZ-equivalence between functions in $\mathbb{F}_2^n$ and classify all APN functions in dimension n for n up to 5. For small amounts of S-boxes, one can test CCZ-equivalence through testing equivalence between linear codes, which is offered by mathematical software Magma [Mag].

Among these equivalence notions, the knowledge on the most general and very important equivalence CCZ is the least clear, which motivates recent work on CCZ-equivalence [CP18], in which Canteaut and Perrin show the necessary conditions on the zeros in DDT (or LAT) of two S-boxes for them to be CCZ-equivalent.

3.5 Relations

Some criteria are not compatible with other cryptographic design criteria, *e.g.,* the most well-known example is that perfect nonlinearity (also Bent) is not compatible with balancedness. However, it is still interesting to clearly quantify the corresponding properties and construct S-boxes that are near optimal regarding an important criterion while satisfying other criteria.

As noted by Leander and Poschmann [LP07], an optimal S-box with respect to linear and differential properties is always optimal with respect to algebraic attacks (in the sense of possessing the maximal algebraic degree). The converse is not true. The following known relations between algebraic degree and Walsh transform might shed some light on this:

Proposition 16 ([Car93])**.** *Let f be a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ ($n \geq 2$), and let $1 \leq k \leq n$. If all $\mathcal{W}_f(a)$ for $a \in \mathbb{F}_2^n$ takes values divisible by 2^k, then f has degree* $\deg(f) \leq n - k + 1$.

[2]The original paper [LP07] contains a small typo stating that there are six non CCZ-classes when there are actually seven (a CCZ-class containing the single AE-class G_{13} is overlooked).

A natural question is that, are there relations between properties related to linear attacks and properties related to differential attack? The answer is yes. DDT is shown to be strongly related to the LAT of an S-box:

Theorem 4 ([CV94, BN13])**.** *Let S be a vectorial Boolean function from $\mathbb{F}_2^n \to \mathbb{F}_2^n$. Then, we have, for any $u, v \in \mathbb{F}_2^n$:*

$$\mathcal{W}_S^2(u,v) = \sum_{a\in\mathbb{F}_2^n} \sum_{b\in\mathbb{F}_2^n} (-1)^{a\cdot u \oplus b\cdot v} \delta_S(a,b).$$

Conversely, for any $a, b \in \mathbb{F}_2^n$:

$$\delta_S(a,b) = 2^{-2n} \sum_{u\in\mathbb{F}_2^n} \sum_{v\in\mathbb{F}_2^n} (-1)^{a\cdot u \oplus b\cdot v} \mathcal{W}_S^2(a,b).$$

Although a vectorial Boolean function is completely specified by its Walsh spectrum and one can recover it from its LAT (refer to Proposition 8), this is not true for the DDT. The DDT corresponds to the absolute LAT (actually, squared), which loses some information.

4 The PEIGEN Platform

In this section, we describe a tool to evaluate most of the properties (security-related and implementation-related) listed above. The goal is to develop a comprehensive and efficient **P**latform for **E**valuating, **I**mplementing, and **GEN**erating S-boxes (PEIGEN), automatically.

PEIGEN is built upon many existing tools, and the most closely related is `LIGHTER` proposed by Jean *et al.* [JPST17]. As will be introduced in Sect. 4.1, one of the two components in `LIGHTER` is dedicated to search for optimal implementations of bijective 4×4-bit S-boxes. Through the comparison among existing tools, `LIGHTER` possesses the most merits. It is more computationally efficient and more versatile. It covers various implementation aspects (*e.g.,*`BGC`, `GC`, `MC`), with flexibility of tuning the set of available gates and customizing the cost for each gate. Although its output implementations are not guaranteed to be optimal (only guaranteed to be optimal $\mathcal{B}$-implementations), they are compatible with results generated using other tools. Besides, one can directly get the implementations of the inverse functions from the generated results for forward functions, and the resulted implementations for software requires less temporary registers (see details in Sect. 4.1). Besides, it is coded in C/C++ and does not depend on external tools, which makes it open to be optimized, extended, and enhanced. With these considerations, and knowing that C/C++ programs can also be integrated as sub-modules to other tools (*e.g.,* SageMath), we finally decided to build PEIGEN on the basis of `LIGHTER` and implement it in the form of pure C++ template headers.

Our ultimate goal for PEIGEN is to become a comprehensive platform for the community to study S-boxes. We aim for enriched functionality, good extendibility, and high efficiency. In what follows, we survey existing tools related to S-boxes and make a comparison before describing PEIGEN in detail.

4.1 Existing Tools on S-box

Tools to study cryptographic properties. A comprehensive tool to study cryptographic properties of S-boxes is SageMath – a general purpose and open source mathematic tool. SageMath contains dedicated modules for cryptography, one of which is the module '`sage.crypto.sbox`' [MLCA]. This module provides a rich set of functionalities to

evaluate cryptographic properties of S-boxes, such as DDT, LAT, and algebraic degrees. Additionally, the module '`sage.crypto.sboxes`' [PW] provides a comprehensive list of S-boxes used in known ciphers. However, SageMath does not yet support some desired functionalities, for example, testing equivalences and testing (v, w)-linearities. Regarding efficiency, when evaluating a large set of S-boxes, SageMath's speed is not quite satisfactory. Apart from SageMath, the Magma Computational Algebra System [Mag] also includes modules for coding theory and cryptography, which is handy for studying S-boxes (especially the modules for coding theory).

A relatively new project in GitHub, named '`libapn`' [FJ], is another open source tool to study vectorial Boolean functions. Its main focus is on APN functions. Functionalities it provides include computing DDT, differential uniformity, and degrees and searching for APN (or δ-uniform) functions. Besides, it can output linear and affine representatives of a permutation. Apart from properties regarding differential attacks, `libapn` does not seem to consider other cryptographic properties or implementation-related functionalities.

Tools to find optimal implementations. Finding the optimal sequence of logical instructions to implement a function is still an open problem, especially for cryptographic functions. However, efficiently achieving this can be very beneficial, for both hardware and software implementations. For hardware, it can minimize the area cost or the latency of an implementation. For software, it can minimize the required number of instructions and reduce the memory consumption in an implementation based on the bitslice technique. The bitslice technique in software implementation was originally introduced by Biham [Bih97] to speed up software implementation of DES and can be used for speeding up the brute force attack. The main idea in the bitslice technique is to use logic operations on n-bit words in software simulating n gate-operations in hardware. Thus, a sequence of logical operations on n-bit words in a software implementation can be seen as n-way parallel gate-operations in a hardware implementation. Using the bitslice technique, one can implement a whole cipher by using logical instructions to avoid table lookups. These can avoid data-dependent memory access to prevent cache-timing side-channel attacks.

As for S-boxes, there are some tools to search for optimal implementations. The early tools date back to the competition for the Advanced Encryption Standard (AES). One candidate in the final list for AES is the block cipher Serpent, whose design comes with an innate bitslice idea and whose performance is good for both hardware and software. At that time, Gladman developed a C program for finding efficient Boolean function decompositions for the Serpent S-boxes and their inverses. The program is open source and can be found online [Gla]. The considered instruction set is $\{\texttt{XOR}, \texttt{AND}, \texttt{OR}, \texttt{NOT}\}$, which are common for modern CPUs. The program adopts depth-first-search (DFS) with heuristics. It can be directly applied for any 4×4-bit S-box. Although it does not necessarily output the best implementations for all 4-bit S-boxes, the outputs are generally good and the efficiency of this program is satisfactory (as can be seen later, and also in [CHM11, BLL15]).

Osvik [Osv00] described a program to search for software implementations of Serpent S-boxes. Reasonably, the target is to find implementations by using minimum CPU cycles with limited available CPU registers. The program takes care of the effect of destructive instructions (replacing one input with the output) of x86 CPU and the parallelizability of instructions. The results are some 2-way or 3-way parallel implementations of Serpent S-boxes costing $6 \sim 8$ CPU cycles. Although the total numbers of instructions in the resulting implementation are larger than those in other implementations, the resulting implementations are more efficient regarding CPU cycles. Unfortunately, this program is not openly accessible.

Inheriting some non-heuristic rules from Osvik [Osv00], Ullrich *et al.* [UDCI+11] presented an iterative deepening depth first search (ID-DFS) to find efficient bitsliced implementations of invertible 4×4-bit S-boxes. Affine equivalence among S-boxes is

considered to effectively prune branches during the search. Thus, the search aims to find the S-box that cost the least within a class of affine-equivalent S-boxes. Accordingly, this approach cannot be used to find the exact implementation of a given S-box. Considering that implementation cost is generally not invariant under affine transformation (also note that `XOR` gate generally costs more than `NAND/NOR/AND/OR` gates in terms of Gate Equivalent), this method is recommended to be used in accordance with the proposed special methodology to design efficient cryptographic primitives [UDCI+11].

Another approach is to adopt logic minimization techniques and the shortest linear straight-line program [BP10, BMP13, CHM11]. Boyar *et al.* [BMP13], showed a two-step process to optimize the logic circuit implementing the S-box of AES. The first step heuristically optimizes the non-linear part in terms of multiplicative complexity (`MC`), and the second step treats the optimization of the resulting linear part as the shortest linear program problem and deals with another heuristic approach. This two-step heuristic searching process can provide good solutions for large S-boxes (*e.g.,* 8×8-bit). For a small S-box (4×4-bit), the heuristic strategies in both steps can be replaced by a SAT solver, as proposed by Courtois *et al.* [CHM11]. Thus, for each step, the solution is optimal. However, the combined solutions are not necessarily optimal.

Stoffelen [Sto16] proposed a method to model the whole problem of finding optimal bitsliced implementation as a problem that can be solved by a SAT solver. The considered notions of implementation complexity include multiplicative complexity (`MC`), bitslice gate complexity (`BGC`), gate complexity (`GC`), and circuit depth complexity (`Depth`). By adaptively increasing from a lower bound of complexity until the SAT solver outputs a solution, the result can be proven to be optimal. The efficiency for searching for optimal implementations with respect to `MC` is quite satisfactory. However, with respect to `BGC`, `GC`, and `Depth`, the efficiency is not satisfactory when dealing with some "strong"[3] S-boxes compared with other heuristic methods (*e.g.,* [Gla] and [JPST17]). Besides, as mentioned in Sect. 3.3, Stoffelen's notions of gate complexity and circuit depth complexity [Sto16] are different from the gate equivalent complexity and the depth complexity that consider weighted-cost of gates, *i.e.,* different gates have different costs regarding area and latency in hardware.

Differentiation among different gates is tackled by Jean *et al.* [JPST17], but in a quite different approach from SAT-based methods. Their deliverable [JPST17] is a tool named `LIGHTER`. `LIGHTER` can generate efficient implementation of a small function given a certain set of available gates and their corresponding costs. Essentially, it handles various notions of implementation merits (`MC`, `BGC`, and `GEC`) in a unified way by assigning a non-fixed weight to each logic operation[4], which allows costs of gates to be customized. There are two independent components in `LIGHTER` to search for implementations: one is for linear functions, and the other is for non-linear functions. The non-linear-search part is dedicated to bijective 4×4-bit S-boxes, which is the part most relevant to this paper.

The approach in the non-linear part of `LIGHTER` is to apply a breath-first-search (BFS) using graphs, combined with a meet-in-the-middle (MITM) strategy. More concretely, it expands two graphs simultaneously: one starts from a root node encoding the identity function I, and the other starts from a root node encoding the target function S. The internal nodes in the two graphs are transformed from their predecessor nodes by using small invertible instructions constructed by atomic logic operations (atomic operations are combined for the sake of invertibility), and they form a set of basic invertible instructions named $\mathcal{B}$-set. An implementation using instructions restricted to a $\mathcal{B}$-set is called $\mathcal{B}$-implementation by Jean *et al.* [JPST17])[5]. The goal is to find a matched node M on the

[3] In the sense that it posses additional cryptographic properties, *e.g.,* $\mathrm{Deg}_{\mathrm{Freq}} = 2^n - 1$ like PRINCE S-boxes, or $\mathcal{BN}_{\mathrm{D}} > 2$ and $\mathcal{L}_1 \leq 4$ like Serpent S-boxes

[4] Here, the logic operations correspond to logic instructions in software or logic gates in hardware.

[5] For example, linear operations `XOR` and `NOT`, or functions of the type of one round Feistel, *e.g.,* $x_0 = x_0$ `XOR` $f(x_1, x_2, x_3)$ where f is composed of 1 or 2 non-linear operations like $f(x_1, x_2, x_3) =$

two increasingly expanded graphs, which connects the two roots I and S with the lowest weighted path (the weight is the sum of the cost of each logic operation on the path). Once a match node is found, one can immediately retrieve the entire sequence of logic operations transforming an identity function I into the target function S, which is essentially the logic implementation of S. The implementations found in this way by using a $\mathcal{B}$-set are optimally small $\mathcal{B}$-implementations (optimal $\mathcal{B}$-implementations).

The approach in `LIGHTER` has several advantages. One is that, from the implementation of S, one can directly obtain an implementation with the same cost for its inverse S^{-1} because of the combinations of invertible Feistel-style operations. Another bonus of the searching strategies regarding software implementations is that the generated implementations require very few registers. For CPU architectures possessing non-destructive instructions, *i.e.,* not replacing one input with the output (*e.g.,* three-operation advanced vector extension (AVX) instructions and reduced instruction set computer (RISC) instructions), the required extra registers in the generated implementations are minimized. Unlike in the results generated by methods of Gladman [Gla] and Stoffelen [Sto16], the hidden costs of `MOV` instructions for register scheduling are directly saved. This merit is not mentioned by Jean *et al.* [JPST17]. A concern on the search strategy in `LIGHTER` is that limiting the strategy to invertible combinations of logic operations results in solutions that are not necessarily optimal. However, by comparing some implementations generated by `LIGHTER` with those by SAT solvers (regarding `BGC`), one will find that the costs of the solutions differ little (essentially, there are no differences in the results on 4×4-bit S-boxes of Piccolo, LAC, Prøst, and RECTANGLE obtained by using the SAT solver). However, `LIGHTER` is more time efficient. For example, our experiment on a 24-core server showed that, for `BGC` of the first Serpent S-box, `LIGHTER` took 2.7 minutes using 24 threads to generate a 14-gate solution. In contrast, the Gladman's program [Gla] took 16.2 minutes using a single thread to generate a 15-gate solution, while the SAT-based tool using the parallel SAT solver Plingling with 24 threads did not find a 14-gate solution within one day.

A missing consideration of `LIGHTER` is on the metric `Depth`, which measures the latency of an implementation. The metric `Depth` is tackled in the SAT-based method [Sto16], but again, the efficiency of the solving procedure is not quite satisfactory and the measurement of `Depth` does not consider that different gates delay differently in hardware. Another tool targeted at evaluating the `Depth` complexity of S-boxes was developed by Guo *et al.* [GJN+16]. Essentially, the tool first generates `Depth`-minimized implementations for almost all balanced 4-variable Boolean functions, which can be used as a small database. For a given S-box, the tool queries the pre-computed database with the four coordinate functions of the S-box. Then, outputs the query results and the maximum among the depths of the four coordinates. This can be done very fast. The pre-computed database is openly accessible at [Qia]. Note that in the implementations stored in the database, the weights of the costs for `XOR`, `AND/OR`, `NAND/NOR`, and `NOT` are fixed to 2, 1.5, 1, and 0.5, respectively.

Essentially, these programs for searching for efficient implementations of a given S-box can be turned into programs for generating S-boxes with efficient implementations and good security properties. In the setting of searching for implementations, the target is the given S-box, whereas in the setting of generating S-boxes, the target is any S-box fulfilling given requirements. As explained in a talk given by Watanabe [WS10], this methodology was used to generate the 4-bit S-boxes of Luffa [DCSW08] and can be named "instruction based S-box design". Watanabe's target [WS10] are S-boxes with implementations optimized for CPU cycles (Intel Core2 with instruction parallelization and a limited number of registers) and with good security properties (including optimal differential uniformity, optimal linearity, high algebraic degree and no fixed point). The S-box of Luffa v1 is

$((x_1$ `AND` $x_2)$ `ORN` $x_3)$.

Table 5: Known tools on S-boxes

Source	Security	MC	BGC/ GC	GEC	Depth	CPU cycles	Method	Speed	Optimal	Open code
[Gla]	✘	✘	✔	✘	✘	✘	Heuristic DFS	✔	✘	✔
[Osv00]	✘	✘	✘	✘	✘	✔	Heuristic	-	✘	✘
[WS10]	✘	✘	✘	✘	✘	✔	Instr. first generation	✔	✔	✘
[UDCI+11]	✘	✘	✔	✘	✘	✘	ID-DFS + AE	-	✔	✘
[BMP13]	✘	✔	✔	✘	✘	✘	Two-step Heuristic	-	✘	✘
[CHM11]	✘	✔	✔	✘	✘	✘	Two-step SAT	-	✘	✘
[Sto16]	✘	✔	✔	✘	✔	✘	SAT	✘	✔	✔
[GJN+16]	✘	✘	✘	✘	✔	✘	LUT	✔	✔	✘
[JPST17]	✘	✔	✔	✔	✘	✘	MITM + BFS	✔	✘	✔
[MLCA]	✔	✘	✘	✘	✘	✘	-	✘	✘	✔
[Mag]	✔	✘	✘	✘	✘	✘	-	✘	✘	✘
[FJ]	✔	✘	✘	✘	✘	✘	-	✘	✘	✔

generated following a *strategic approach*. The strategic approach uses iterations on basic invertible functions like the generalized Feistel structure. Following this strategy, the generated S-boxes can cost minimal CPU cycles. However, because of the similarity between ANFs of its coordinates in the S-box of Luffa v1, the step-reduced versions of Luffa v1 suffer higher order differential attacks [WHYK10]. That is the motivation for the non-strategic approach in the generation of the S-box in Luffa v2, which uses random combinations of instructions combined with regular testing on the bijective and security properties.

Note that, the strategy used to find an efficient implementation of a given S-box by Jean *et al.* [JPST17] adopts invertible functions similar to those used in the strategic approach of Watanabe [WS10]. The difference is that each invertible function in Watanabe's approach [WS10] involves one non-linear operation, whereas an invertible function in Jean *et al.*'s approach [JPST17] can involve more than one non-linear operations. Besides, when combining those invertible functions, Watanabe's strategy [WS10] adopts identical Feistel shuffle (bit rotation), whereas Jean *et al.*'s strategy [JPST17] allows any bit-permutations on the inputs. Thus, the strategy of Jean *et al.* [JPST17] covers a far larger range (on the space of S-boxes) than the strategic approach of Watanabe [WS10].

To make pros and cons of existing tools on S-boxes clearer, we list their main characteristics in Table 5. From Table 5 and the above discussions, we can see that `LIGHTER` possesses the most merits, which is the main reason why we decided to build Peigen on the basis of `LIGHTER`.

4.2 Functionalities of Peigen

In this section, we briefly describe the functionalities currently provided by Peigen. On the whole, Peigen can efficiently evaluate most security-related properties of given S-boxes, find good implementations for given S-boxes under various techniques and merits, and automatically generate S-boxes fulfilling given criteria.

4.2.1 Evaluating

Given a set of n-bit S-boxes (for $3 \leq n \leq 8$), PEIGEN can evaluate their security-related properties. Specifically, properties currently included are as follows:

1. **Resistance to Differential attack**: computes (and prints when required) the Difference Distribution Table (DDT); and outputs the differential uniformity ($\mathcal{U}(S)$), the differential spectrum ($\mathcal{D}_{\text{spec}}(S)$), the maximum entry in DDT_1 ($\mathcal{U}_1(S)$), the numbers of nonzero entries in DDT_1 (CardD1(S)), and the differential spectrum observed from DDT_1 ($\mathcal{D}_{\text{spec}_1}(S)$).

2. **Resistance to Linear attack**: computes (and prints when required) the Linear Approximation Table (LAT); and outputs the linearity ($\mathcal{L}(S)$), the Walsh spectrum ($\mathcal{W}_{\text{spec}}(S)$), the maximum entry in LAT_1 ($\mathcal{L}_1(S)$), the numbers of nonzero entries in LAT_1 (CardL1(S)), and the Walsh spectrum observed from LAT_1 ($\mathcal{W}_{\text{spec}_1}(S)$).

3. **Resistance to Boomerang attack**: computes (and prints when required) the Boomerang Connectivity Table (BCT); outputs the boomerang uniformity ($\mathcal{BU}(S)$) and the boomerang spectrum $\mathcal{BD}_{\text{spec}}(S)$.

4. **Resistance to Algebraic attack**: computes (and prints when required) the ANF of all coordinate/component functions $S_0 \sim S_{2^n-1}$; and outputs the maximum algebraic degree ($\text{Deg}(S)$), the minimum algebraic degree ($\min\deg(S)$), the degree spectrum ($\text{Deg}_{\text{spec}}(S)$), the maximal degree of the product of k coordinates (d_k for $1 \leq k \leq n$), and the table representation of $\mathcal{V}_S(u)$ for all u indicating the appearance of monomials in the ANFs of $x \mapsto \pi_v(S(x))$ for $v \in \mathbb{F}_2^n$.

5. **Linear structures and (v, w)-linearity**: computes (and prints when required) the linear-structures $\text{LS}(S)$ and the total number, and the (v, w)-linearity table, *i.e.,* the number $N_{(v,w)}$ of subspaces V of dimension v for which there exists a w-dimensional subspace W such that the S-box is (v, w)-linear with respect to (V, W); print all non-trivial subspace (V, W) pairs, the maximal dimension max_v among subspaces V such that there exist a subspace W for which the S-box is (v, w)-linear, and the maximal dimension max_w among subspaces W such that there exists a subspace V for which the S-box is (v, w)-linear.

6. **Others**: output whether it is a permutation, whether it is an involution.

Given n-bit S-boxes (for $3 \leq n \leq 8$ if not otherwise stated), PEIGEN can evaluate their equivalence relations:

1. **Permutation-XOR equivalence (PE, XE, PXE)**: for a given S-box, outputs the representative of its PE- and PXE-class; for a given S-box, outputs all S-boxes PE, XE, and PXE with it; for two given S-boxes, outputs whether they are PXE.

2. **Linear equivalence (LE)**: for a given S-box, outputs the minimum representative of its LE-class; for two given S-boxes, outputs whether they are LE. If they are LE, outputs the linear transformations between them.

3. **Affine equivalence (AE)**: for two given S-boxes, outputs whether they are AE. If they are AE, outputs the affine transformations between them; for a given 4-bit S-box, outputs whether it belongs to one the 16 optimal classes. If it belongs to at least one of them, outputs to which optimal class it belongs.

4. **Partition an AE-class into PXE-classes**: for a given 4-bit S-box, outputs all PXE-representatives of the AE-class to which this S-box belongs, *i.e.,* split one AE-class into mutually non-PXE classes, outputs their representatives.

4.2.2 Implementing

Given a set of n-bit S-boxes and the specific implementation configuration (user-tuned set of available gates and costs for each gate), PEIGEN can provide their implementations under various techniques, which are good with respect to different merits (support for $3 \leq n \leq 8$, but the tool is only efficient for 3- and 4-bit S-boxes):

1. **Bitslice gate complexity (`BGC`), Gate equivalent complexity (`GEC`), and Multiplicative complexity (`MC`):** outputs implementations optimized for area and code size, *i.e.,* with minimized number of gates/equivalent gates under different implementation techniques (*e.g.,* software logic instructions, hardware gates TSMC65nm tech. and UMC180nm tech.), or with minimized number of non-linear operations;

2. **Depth complexity (`Depth`):** outputs implementations optimized for latency, *i.e.,* with minimized depth of critical path under different techniques (*e.g.,* hardware tech. TSMC65nm and UMC180nm, and software logic instructions); at the same time, keeps the area as small as possible (*i.e.,* for two implementations posses the same `Depth`, output the one with less `GEC`).

4.2.3 Generating

There are two usages in PEIGEN with respect to generating S-boxes from given criteria:

1. **Filtering out good S-boxes**: Given a set of n-bit S-boxes and a set of criteria (user-required security-related and implementation-related properties), PEIGEN filters out the S-boxes fulfilling the criteria, and at the same time, outputs the detailed evaluations of their security properties and their implementations under a given configuration on gates;

2. **Generating new S-boxes from scratch**: Given a set of criteria (user-required security-related and implementation-related properties listed above), PEIGEN

   (a) generates a set of S-boxes fulfilling the given criteria, and at the same time, outputs the detailed evaluations of their security properties and their implementations under a given configuration on gates;

   (b) classifies the generated S-boxes in accordance with their detailed properties by distributing the results on the generated S-boxes into different folders. Those folders are named after the property profiles of the S-boxes. Thus, the S-boxes with the same property profile are gathered together into the same folder. Doing this makes it much easier for users to read and make further observations on the results. The consideration on adding this functionality is that, PEIGEN can generate a large number of S-boxes fulfilling the given criteria. If all of them are output without distinction, they will be difficult for users to read. Thus, for each generated S-box, PEIGEN only outputs its LUT and the evaluations of their security properties, and identifies its permutation-equivalent (PE) representative. Then, PEIGEN only outputs implementations of one S-box among all generated S-boxes that are PE. Note that, if two S-boxes are PE, then given the implementation of one S-box, one can directly obtain the implementation of the other S-box by renaming the input/output variables from the original implementation.

4.3 Efficiency, Expandability and Compatibility of PEIGEN

PEIGEN is developed with efficiency in mind for the search on large sets of S-boxes. Apart from programming-level optimizations (*e.g.,* adopting fine-grained parallelism by performing

vectorization using single instruction, multiple data (SIMD) instructions, supporting higher-level parallelism by using OpenMP), we optimized the most time/memory-consuming parts in the algorithmic level.

Approach and performance for evaluation. For evaluating S-boxes, the most time-consuming computations are testing affine equivalence and generating pairs of subspaces (V, W) with respect to which S-box is (v, w)-linear. However, for linear/affine equivalence, our implementation of the primary algorithms offered by Biryukov *et al.* [BCBP03] turns out to be sufficiently efficient even for a large set of S-boxes (*e.g.,* 2^{14} 4-bit S-boxes). For (v, w)-linearity, an implementation performing exhaustive testing in accordance with the definition is also sufficiently efficient for most known S-boxes. Concretely, for evaluating a comprehensive list of known S-boxes provided by Perrin and Wiemer [PW] together with a few additions, PEIGEN takes less than 1 second for all 3-bit to 6-bit S-boxes, less than 5 seconds for a 7-bit S-box, and less than 2 minutes for most 8-bit S-boxes. The inefficiency is confined to 8-bit S-boxes with too many (V, W)-subspace pairs with respect to which the S-box is linear, so that PEIGEN takes more time to compute and output (*e.g.,* it takes about 12 minutes for the S-box of SKINNY-128 [BJK+16] and about 27 minutes for the S-box of CSS [PMA07]). We defer the details of the evaluation results to Sect. 5.1.

Approach and performance for implementation. Compared with evaluating, implementing and generating S-boxes requires more effort to achieve high efficiency, especially for a large set of S-boxes. The basic approach in PEIGEN to find an efficient implementation of a given S-box is based on the approach in `LIGHTER`. On the basis of its non-linear part introduced in Sect. 4.1, we propose the following optimizations:

1. **Composition and concatenation:** We notice that there is an isomorphism between the two graphs expanded from the two roots respectively encoding the identity function I and an target function S (we will use the same notations I and S to represent the functions and the nodes in the graphs). Essentially, the graphs expanded (using the same set of basic instructions and following the same procedure) from any root node encoding a permutation will be isomorphic. Accordingly, we only need to expand one graph from a root node I. For a given target function S, we compose S with each function represented by the node in the graph and use the composite function to match the nodes in the same graph. Thanks to the mechanism used to build the graph (using invertible basic instructions, for which both the forward and the inverse computations are trackable), once a match is found, one can backtrack both of the two sequences of instructions – the sequence generating the match node and the sequence generating the node composed to the target function. By concatenating the two instruction sequences, one can obtain a logic implementation of the target function S.

2. **Pre-computation:** Note that, in the composition and concatenation method, the graph is expanded from I without any given target. When there is a set of targets (*i.e.,* need to implement a set of S-boxes), this graph can be built once and for all. Thus, we can pre-compute the graph and store it in the binary form for reuse. This might also conduce cross-node parallelization.

3. **Other amendments:**

   (a) Note that there is an equivalence between different decompositions of an implementation (*i.e.,* a sequence of instructions). Specifically, if an implementation can be found by using the concatenation of two short instruction sequences $\texttt{Imp}_1 \| \texttt{Imp}_2$, then it can also be found by using the composition $\texttt{Imp}'_1 \| \texttt{Imp}'_2$, where $\texttt{Imp}'_1 = \texttt{Imp}_1 \| \texttt{Ins}_1$ and $\texttt{Imp}_2 = \texttt{Ins}_1 \| \texttt{Imp}'_2$. Thus, to speed up the search,

we exploit such equivalence to avoid duplicate matching. Note that different invertible instructions `Ins` are composed by different sets of atom gates, thus their costs are not continuous numbers. As a result, to deal with such discontinuity, some duplicated matchings cannot be completely avoided while finding the implementations with the least costs.

(b) We amended small issues related to `ANDN/ORN` in the non-linear part of `LIGHTER`:

i. Full support for `ANDN` and `ORN`: the original program does not fully support `ANDN/ORN` (without support for `MOAI1/MAOI1` combined with `ANDN/ORN`. With support for `XOR` ∘ `AND/OR` ∘ `ANDN/ORN`, but without dealing with the asymmetry of the `ANDN/ORN` operations), we amended these small issues. Some resulting implementations can achieve a slight improvement[6].

ii. Correct the print error relating `ANDN/ORN`: because it does not consider asymmetry of the `ANDN/ORN` operations, the original program occasionally prints the operators in the triple-gate instructions involving `ANDN/ORN` in the wrong order. We amended this small issue.

(c) We extended the cover range of implementation target from 4-bit S-boxes to 3 ∼ 8-bit S-boxes. Unfortunately, Peigen is only efficient for 3 and 4-bit S-boxes and barely able to handle 5-bit S-boxes. For larger S-boxes, heuristic strategies and further optimizations are required.

Approach and performance for generation. With its composition, concatenation, and pre-computation mechanisms, together with the evaluation functionality in Peigen, it is quite handy to build the module to efficiently generate S-boxes fulfilling given criteria.

The approach is to simply compose nodes in the graph to obtain a composite function and then test whether its security-related properties fulfill given criteria. If a criterion on implementation costs is provided, this criterion will be taken as an upper bound and Peigen only outputs those S-boxes with by far the lowest implementation costs. To minimize the cost of the implementation, the composition procedure starts from low layer nodes and increases to high layer nodes with updated lower- and upper-bounds on the implementation costs. Otherwise (heuristically starting from some high layer to perform matching), the procedure may rapidly find S-boxes fulfilling the given criteria but not assure the best implementation costs. Note that, when composing nodes, a bit-permutation is added inbetween. That is because the nodes in the graph are all non-PE (regarding output bits), *i.e.,* one node in the graph represent a class of output-permutation-equivalence functions[7]. To make the composition be functionally complete (*e.g.,* can express every bijective function as can be expressed by a single logic sequence without composition), all possible bit permutations should be considered in the between when composing two functions (*i.e.,* two nodes in the graph)[8].

In Sect. 5.3, we will present some examples of the generated S-boxes together with the time it took, from which it can be seen that the efficiency of generating S-box is practical.

Approach and performance for finding `Depth`-optimal implementations. Peigen can find optimal implementations with respect to `Depth` given a certain set of available gates and their corresponding latency. The approach differs greatly from the approach used to

[6]For example, for `BGC`, `DES_S2_2` changes from 15 to 14 GEs, `JH_S0` changes from 16 to 15 GEs; for `GEC` with TSMC65nm tech., `Whirlpool_E` from 26.5 to 26 GEs, `BLAKE_7` from 25 to 24.5 GEs; for `GEC` with UMC180nm tech., `DES_S4_2` from 22.99 to 22.33 GEs, `Twofish_Q1_T3` from 22 to 21.33 GEs.

[7]But still, implementations of all functions represented by this node can be retrieved by combining the sequence of instructions together with a permutation on the variable index.

[8]This is why a sequence of instructions for variable substitution appears in the middle of the generated implementations in our results. Note that, those variable substitutions can be manually removed and thus do not cause additional costs.

find efficient implementations regarding `BGC` and `GEC`. The MITM, and composition and concatenation strategies in searching efficient implementations regarding `BGC` and `GEC` are not straightforward to extend to the case of `Depth`, because the optimization objectives differ significantly, and more importantly, the inverse of an instruction sequence usually possesses different latency from the forward direction. Although the essential algorithm is modified, we adopt a similar framework of the program regarding `BGC` and `GEC` to inherit the merit of supporting a customizable set of available gates and user-tuned gate costs. In the kernel, we apply a similar method to that used by Guo *et al.* [GJN+16, Qia]. Concretely, the program generates `Depth`-minimized implementations for each of the coordinates of a given S-box. The `Depth` of the S-box is the maximum among depths of its coordinates. The generating process adopts BFS to increasingly expand a graph in which each node represents a Boolean function. This graph can also be built once and for all and stored for reuse later. An addition is that, besides `Depth`, we try to make the area as small as possible at the same time. However, we can only achieve the local optimal for `GEC`, *i.e.,* minimized `GEC` among minimized `Depth` for each coordinate. For global optimization (the S-box) for `GEC`, the best that can be achieved is to choose the implementation with the most sharable instruction-prefix between implementations of different coordinates. With this, we can achieve slightly improvement on the results obtained by Qiao [Qia] (with a fixed set of available gates and gate costs). Note that, for `Depth`, the results are guaranteed to be optimal.

As for efficiency, under different sets of available gates and gate cost, the time and memory consumption are generally different. For $3 \sim 4$-bit S-boxes, PEIGEN is generally satisfactory. For example, when the available gate set is {`XOR`/`XNOR` (2 GEs), `AND`/`OR` (1.5 GEs), `NAND`/`NOR` (1 GE), and `NOT` (0.5 GE)}, PEIGEN took 95 minutes on a 24-core server to find all the `Depth`-minimized implementations of the 206 known 4-bit S-boxes, during which building the graph took almost all the time, *i.e.,* if there are more given S-boxes, the required time will stay almost the same.

Expandability and compatibility. To make PEIGEN expandable, we use template-based C++ implementations to facilitate coverage of larger amount of S-boxes. Besides, it is very convenient to add new functionalities (*e.g.,* evaluating other security-related properties) to the template classes. Also, to make PEIGEN compatible with `SageMath` (the functional mathematics software as introduced in Sect. 4.1), we add `Cython` interfaces for classes of PEIGEN so that they can be compiled with "sage" command lines, and called as modules by sage.

4.4 Future Work on PEIGEN

To make PEIGEN a more comprehensive platform for researching S-boxes, there is still a lot of work to do in terms of functionality and efficiency. For functionalities related to cryptographic properties, the following work will be of great significance.

1. Reverse-engineering: given the LUT of an S-box, discover the hidden structures of the S-box that reveal unknown design criteria [BPU16];

2. Reconstructing: given the DDT, reconstruct all the S-boxes having this DDT (recover a class of DDT-equivalent S-box) [DH18, BCJS18]; This problem essentially belongs to a more general problem, that is, given a cryptographic property, generate all S-boxes (or equivalence classes of S-box) possessing this property. Similar work includes also, given the differential uniformity δ, generating all δ-uniform functions (when $\delta = 2$, this corresponds to finding APN functions);

3. Partitioning: partition the whole space of S-boxes with a certain domain size into equivalence classes (*e.g.,* CCZ-class, AE-class [LP07, BL08]); or partition a larger

equivalence class into smaller equivalence classes (*e.g.,* partition an CCZ-class into EA-classes [CP18], and an EA-class into PXE-classes as has been done for 4-bit S-boxes);

For functionalities related to implementations and for efficiency, the following work will also be of great value.

1. Generating good implementations with resistance to side-channel attacks: although implementations that are good regarding `MC` can be used to construct good implementations protected from side-channel attacks, the current generated implementations are plain logic sequences without protection. Fully and automatically generating protected implementations that are good with respect to software (for microprocessors) and hardware (for various implementation techniques) is part of our future work here.

2. Generating good implementations regarding CPU cycles taking instruction parallelism and register scheduling into consideration (for software on high-end CPUs).

3. Generating good implementations for large S-boxes: although the current tools support finding implementations for $5 \sim 8$-bit S-boxes, the efficiency is not satisfactory. Effective heuristic strategies need to be devised and applied.

5 Evaluation Results

5.1 Summarizations on Cryptographic Properties of Existing S-boxes

With the evaluation component of Peigen, we evaluated cryptographic properties of existing S-boxes (3-bit to 8-bit) in most known block ciphers. As mentioned above, a comprehensive list of known S-boxes is provided by Perrin and Wiemer [PW]. We added a few missing ones to the list and evaluated them all. For readers who are interested in the resulting summary on cryptographic properties of existing S-boxes, details are provided in the supplementary materials, which provide concisely summarized and thoroughly detailed evaluations.

A note on high (v, w)-linearity and large number of linear structures It is worth noting that results regarding the high (v, w)-linearity (also the large number of linear structures) of some S-boxes might be worth deeper analysis because they might reveal some undesired properties of the S-boxes. For example, for 5-bit S-boxes, both the S-box of KECCAK [BDPVA] and the S-box of Ascon [DEMS16] have large (V, W) subspace pairs restricted to which the S-box is linear. Concretely, both of them are $(4, 2)$-linear and $(3, 4)$-linear, with a large number (91) of linear structures; for 8-bit S-boxes, the S-boxes of CSS [PMA07], SKINNY-128 [BJK+16], and Fantomas [GLSV14] also have large subspace pairs restricted to which they become linear. Concretely, the S-box of CSS is $(7, 5)$-linear and $(4, 7)$-linear, with 6465 linear structures; the S-box of SKINNY-128 is $(7, 2)$-linear and $(3, 7)$-linear, with 601 linear structures; the S-box of Fantomas is $(6, 2)$-linear and $(2, 7)$-linear, with 83 linear structures. That is to say, by fixing only a few input bits (even a single bit) with an arbitrary value, a large part of the output bits (even all but a single bit) becomes linearly dependent on the remaining input bits. This can be described as linearizable/partially-linearizable property of the S-box. This linearizable/partially-linearizable property of the S-box of KECCAK has been exploited to improve cube-attacks and collision attacks [DMP+15, GLS16, QSLG17, SLG17]. Two questions naturally arise. By fully exploiting those subspace pairs (V, W) restricted to which the S-box is linear, can the complexities of these attacks be further reduced? For other ciphers with S-boxes possessing large dimension (V, W) subspace pairs restricted to which they become linear,

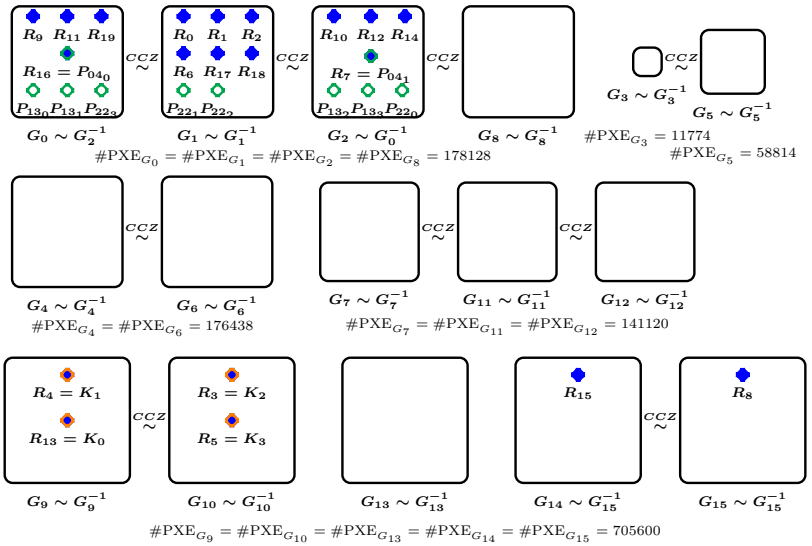

Figure 1: Relations between known equivalent classes of 4-bit S-boxes. In this figure, G_0 to G_{15} (in black frame) are the 16 non-AE optimal 4-bit S-boxes in [LP07] (see Table 11); R_0 to R_{19} (in blue) are the 20 non-PXE Serpent-type S-boxes in [LP07] (see Table 12); K_0 to K_3 (in orange frame) are the 4 non-PXE Golden S-boxes in [Saa11] (see Table 13); P_{ij_k} (in green frame) are the 10 non-PXE Platinum S-boxes in [ZBRL15] (see Table 14). Besides, $\#\text{PXE}_{G_i}$ represents the number of non-PXE S-boxes with class G_i.

can this property be exploited to launch efficient attacks on the ciphers? We leave these as open problems.

Relations between known equivalent classes of 4-bit S-boxes With the functionalities on equivalence relations of PEIGEN and the knowledge of good classes of 4-bit S-boxes, we find the relations between some known classes of 4-bit S-boxes proposed in different studies.

For example, one can easily confirm those PXE-classes within the 16 optimal classes of 4-bit S-boxes, which are pointed out in the literature, *i.e.,* the 20 non-PXE Serpent-type S-boxes [LP07], the 4 non-PXE Golden S-boxes [Saa11], and the 10 non-PXE (three) Platinum S-boxes [ZBRL15]. This is achieved by first partitioning each of the 16 optimal AE-classes into several PXE-classes. Then, the 'filtering' functionality of PEIGEN is used to filter out those PXE-representatives fulfilling given criteria (definitions of those good S-boxes). Moreover, Fig. 1 points out the overlaps between those known PXE-classes. Concretely, the 4 non-PXE Golden S-boxes [Saa11] are actually 4 out of the 20 Serpent-type S-boxes [LP07], and the 2 out of the 10 non-PXE Platinum S-boxes [ZBRL15] are actually 2 out of the 20 Serpent-type S-boxes. There is no overlap between the 4 Golden S-boxes and the 10 Platinum S-boxes.

Using similar method, one can find all PXE-classes with other specific criteria. For example, with the maximum value $\mathcal{U}_1 \leq 2$ in DDT_1 and $\mathcal{L}_1 \leq 4$ in LAT_1 instead of $\text{CardD1} + \text{CardL1} \leq 4$, one can find 112882 non-PXE optimal S-boxes.

5.2 Implemented S-boxes

Using the components for finding efficient implementations in PEIGEN, we implement invertible S-boxes (3-bit and 4-bit) in known block ciphers under various techniques.

Specifically, we conduct efficient software implementations corresponding to BGC; efficient hardware implementations corresponding to GEC under TSMC65nm tech. and UMC180nm tech.; and Depth-optimal software implementations, Depth-optimal hardware implementations corresponding to TSMC65nm tech., UMC180nm tech., and STM65nm tech. The summarized results and detailed implementations can be found in our supplementary materials.

5.3 Generated S-boxes

With the generation component of PEIGEN, we try to generate more efficient S-boxes fulfilling design criteria of known ciphers. When generating S-boxes, we only output the implementation of one S-box among its PE-class, because implementation cost is invariant under permutation of input/output bits, and from the implementation of one S-box, one can directly obtain that of others in the same PE-class. Note that, the generated S-boxes might be XE and we do not restrict generated S-boxes to having no fixed points because fixed points can be removed by XOR constants. Although, implementation cost is not invariant under XOR constants, we can claim that the floating range among implementation costs of XE S-boxes is small.

3-bit S-boxes For 3-bit S-boxes, the best properties regarding differential and linear attack are $\mathcal{U} = 2$ and $\mathcal{L} = 4$. The optimal BGC[9] for those optimal 3-bit S-boxes is 6 gates. There are 40 non-PE optimal classes with this minimum software cost. For these 40 non-PE S-boxes, their GECs (hardware costs) are $10.5 \sim 11.5$ GEs under TSMC65nm tech., are $9 \sim 10.67$ GEs under UMC180nm tech..

4-bit S-boxes For 4-bit S-boxes, we run the generator with different set of criteria. For each set, we limited the run time to be one day (the run was terminated if it did not finish). When the criteria are set to be:

1. CriteriaSet0 = $\{\mathcal{U} \leq 4,\ \mathcal{L} \leq 8,\ \texttt{BGC} \leq 8\}$: all such S-boxes can be generated within 10 minutes with a pre-computed graph. There are in total 851 non-PE S-boxes, which are all from $\{G_0, G_2, G_8\}$. Note that, all permutations generated by using fewer than 9 gates have been tested, and no permutation with fewer than 8 gates fulfills $\{\mathcal{U} \leq 4,\ \mathcal{L} \leq 8\}$. Thus, 8 is the optimal BGC for optimal 4-bit S-boxes. Also note that, the generated S-boxes under this criteria set might be comparable with known S-boxes, *e.g.,*PRIDE, Prøst, Piccolo, and SKINNY-4-bit S-boxes. For these generated 851 non-PE S-boxes, their GEC are $14 \sim 15.5$ GEs under TSMC65nm tech., are $12 \sim 14.34$ GEs under UMC180nm tech..

2. CriteriaSet1 = $\{\mathcal{U} \leq 4,\ \mathcal{L} \leq 8,\ \mathcal{U}_1 \leq 2\ , \mathcal{L}_1 \leq 4,\ \texttt{BGC} \leq 11\}$, the S-boxes generated within one day (1256 non-PE S-boxes, not complete) are all from $\{G_0, G_1, G_2, G_8\}$. Note that, the set $\{G_0, G_1, G_2, G_8\}$ forms one of the 7 non-CCZ-equivalent classes for optimal 4-bit S-boxes. Without the restriction on BGC, all the 16 optimal classes have some S-boxes fulfilling this set of criteria. For the generated 1256 non-PE S-boxes, their GEC are $21.5 \sim 22.5$ GEs under TSMC65nm tech., are $18 \sim 20.01$ GEs under UMC180nm tech..

3. CriteriaSet2 = $\{\mathcal{U} \leq 4,\ \mathcal{L} \leq 8,\ \mathcal{U}_{\text{Freq}} \leq 18,\ \texttt{BGC} \leq 11\}$, the S-boxes generated within one day (14530 non-PE S-boxes, not complete) are all from $\{G_9, G_{10}, G_{14}, G_{15}\}$. Note that, for optimal 4-bit S-boxes, the criteria in the set $\{\mathcal{U}_{\text{Freq}} = 18,\ \mathcal{L}_{\text{Freq}} =$

[9] In the sequel, by optimal, we mean in the sense of optimal $\mathcal{B}$-implementation; By BGC, we mean using the gate set {NOT, AND/OR, XOR, ANDN } and each gate has the same cost. We include ANDN in this set because BGC closely corresponds to software implementation and many processor architectures support ANDN.

32, $\mathrm{Deg}_{\mathrm{Freq}} = 14$, $\#\,\mathrm{LS} = 3\}$ are simultaneously fulfilled, and essentially characterizes the classes $\{G_9, G_{10}, G_{14}, G_{15}\}$. Thus, for S-boxes in these classes, we claim that the optimal `BGC` is 11 gates (we did not find an S-box fulfilling this set of criteria with 10 gates, and the searching limited by 10 gates is completed). For the generated 14530 non-PE S-boxes, their `GEC` are $18 \sim 20$ GEs under TSMC65nm tech., are $15.33 \sim 18.67$ GEs under UMC180nm tech..

4. CriteriaSet3 = $\{\mathcal{U} \leq 4,\ \mathcal{L} \leq 8,\ \mathcal{U}_{\mathrm{Freq}} \leq 15,\ \texttt{BGC} \leq 12\}$, the S-boxes generated within one day (10892 non-PE S-boxes, not complete) are all from $\{G_4, G_{12}\}$. Note that, for optimal 4-bit S-boxes, the criteria in the set $\{\mathcal{U}_{\mathrm{Freq}} = 15,\ \mathcal{L}_{\mathrm{Freq}} = 30,\ \mathrm{Deg}_{\mathrm{Freq}} = 15,\ \#\,\mathrm{LS} = 0\}$ are simultaneously fulfilled, and essentially characterizes the classes $\{G_3, G_4, G_5, G_6, G_7, G_{11}, G_{12}, G_{13}\}$. Also note that, this set of criteria is essentially the design choice of PRINCE S-boxes. Thus, for S-boxes within these classes or PRINCE-type S-boxes, the optimal `BGC` is 12 gates (we did not find any S-box fulfilling the set of criteria when limiting `BGC` to be 11 gates). For the generated 10892 non-PE S-boxes, their `GEC` are $19 \sim 24$ GEs under TSMC65nm tech., are $16.33 \sim 21.34$ GEs under UMC180nm tech..

5. CriteriaSet4 = $\{\mathcal{U} \leq 4,\ \mathcal{L} \leq 8,\ \mathcal{U}_1 = 0,\ \mathcal{L}_1 \leq 4,\ \texttt{BGC} \leq 11\}$, the S-boxes generated within 16 hours (32 non-PE S-boxes, completed) are all from $\{G_1\}$. Note that, except the criterion on `BGC`, this set of criteria is essentially the design choice of Serpent[10] and PRESENT S-boxes. For the generated S-boxes, their `GEC` are $23.5 \sim 24$ GEs under TSMC65nm tech., are $20 \sim 21$ GEs under UMC180nm tech..

6. CriteriaSet5 = $\{\mathcal{U} \leq 4,\ \mathcal{L} \leq 8,\ \mathrm{CardD1} \leq 2,\ \mathrm{CardL1} \leq 2,\ \texttt{BGC} \leq 11\}$, the S-boxes generated within 6 hours (64 non-PE S-boxes, completed) are from $\{G_0, G_1, G_2\}$. Note that, except the criterion on `BGC` and fixed points, this set of criteria is the design choice of RECTANGLE S-boxes. For the generated S-boxes, their `GEC`s are $21.5 \sim 22$ GEs under TSMC65nm tech., are $18 \sim 19.34$ GEs under UMC180nm tech..

Note that, many of the generated S-boxes have lower cost but the same security level (fulfilling the design criteria) as the S-boxes used in known ciphers.

5-bit S-boxes For 5-bit S-boxes, we run the generator with the following set of criteria:

1. CriteriaSet0 = $\{\mathcal{U} \leq 8,\ \mathcal{L} \leq 16,\ \texttt{GEC}$ (TSMC65nm tech.) $\leq 18\}$: for this set of criteria, the security level regarding differential uniformity and linearity is the same as that of the KECCAK and Ascon S-box. Within one day, the generator generated 11471 non-PE S-boxes fulfilling this set of criteria. Some generated S-boxes are involution, and some have degree 4. Unfortunately, all generated S-boxes (both themselves and their inverse) have quadratic components.

2. CriteriaSet1 = $\{\mathcal{U} \leq 8,\ \mathcal{L} \leq 16,\ \min\deg \geq 3,\ \texttt{GEC}$ (TSMC65nm tech.) $\leq 21.5\}$: Note that, this are the criteria fulfilled by inverse S-boxes of KECCAK and Ascon. Within one day, it generated 6220 non-PE S-boxes fulfilling this set of criteria. Some generated S-boxes have maximum degree 4 and minimum degree 3 among all components. For all generated S-boxes, their inverse have maximum degree larger than 3 but also quadratic components.

Considering that, APN and 4-uniform 5-bit S-boxes are so sparse (also true for larger S-boxes), one is unlikely to be found by this instruction-first design method without heuristics. Therefore, we give up trying and call for a more rational approach.

We refer the readers who are interested in the generated S-boxes to our supplementary materials for more details.

[10] Note that, the design criteria of Serpent S-boxes include $\mathcal{L}_1 \leq 4$, which is excluded in the definition of Serpent-type S-box in [LP07]

5.4 Observations and Discussions on Inclusive and Exclusive Criteria

Observing the evaluated results on 4-bit S-boxes, we find that some criteria are inclusive, which allows designers to focus their efforts on the superior criterion; some are exclusive, which requires designers to make careful trade-offs.

Inclusive Criteria

1. Although the criteria on platinum S-boxes are imposed to limit the number of nonzero entries in DDT_1 and LAT_1, concretely $\mathrm{CardD1} + \mathrm{CardL1} \leq 4$, we find that they all simultaneously satisfy another merit, namely, $\mathcal{U}_1 \leq 2$ and $\mathcal{L}_1 \leq 4$. In other words, if we limit the number of 1-bit to 1-bit differential propagations (resp. linear approximations) to be as small as possible, we can simultaneously assure their probabilities (resp. bias) to be small as well (not the maximum $\mathcal{U} = 4$ and $\mathcal{L} = 8$).

   More generally, we find that for all optimal 4-bit S-boxes, by minimizing CardL1, the $\mathcal{L}_1$ is automatically minimized, *i.e.,* $\mathrm{CardL1} = 2 \implies \mathcal{L}_1 = 4$. [11]

   As for DDT_1, since the minimal value that the CardD1 can take is 0, the corresponding phenomenon is quite natural, *i.e.,* $\mathrm{CardD1} = 0 \iff \mathcal{U}_1 = 0$.

   However, simultaneous minimization on both DDT_1 and LAT_1 cannot be achieved. In other words, $\{\ \mathcal{U}_1 = 0$ and $\mathrm{CardD1} = 0\ \}$ and $\{\ \mathcal{L}_1 = 4$ and $\mathrm{CardL1} = 2\ \}$ cannot hold simultaneously.

2. For all optimal 4-bit S-boxes, the characters in the set $\{\mathcal{U}_{\mathrm{Freq}},\ \mathcal{L}_{\mathrm{Freq}},\ \mathrm{Deg}_{\mathrm{Freq}},\ \#\mathrm{LS},\ \max_v(v,w)\text{-linear}\}$ are always determined simultaneously, *i.e.,* if one is fixed, the others can be known directly. Concretely:

$$\mathcal{U}_{\mathrm{Freq}} = 24 \ \Leftrightarrow\ \mathcal{L}_{\mathrm{Freq}} = 36 \ \Leftrightarrow\ \mathrm{Deg}_{\mathrm{Freq}} = 12 \ \Leftrightarrow\ \#\mathrm{LS} = 9 \ \Leftrightarrow\ \max_v(3,2)\text{-linear}.$$
$$\mathcal{U}_{\mathrm{Freq}} = 18 \ \Leftrightarrow\ \mathcal{L}_{\mathrm{Freq}} = 32 \ \Leftrightarrow\ \mathrm{Deg}_{\mathrm{Freq}} = 14 \ \Leftrightarrow\ \#\mathrm{LS} = 3 \ \Leftrightarrow\ \max_v(3,1)\text{-linear}.$$
$$\mathcal{U}_{\mathrm{Freq}} = 15 \ \Leftrightarrow\ \mathcal{L}_{\mathrm{Freq}} = 30 \ \Leftrightarrow\ \mathrm{Deg}_{\mathrm{Freq}} = 15 \ \Leftrightarrow\ \#\mathrm{LS} = 0 \ \Leftrightarrow\ \max_v(2,2)\text{-linear}.$$

   Thus, when generating S-boxes, one only needs to impose one of them as criteria. Considering that to test redundant properties wastes time, the filtering criteria should be non-reducible.

   There is a corresponding question: which one of the criteria in the set decides the others? For example, does the differential spectrum (or Walsh spectrum) imply the degree spectrum? This may be answered by using the relation between the divisibility of the Walsh spectrum and the degree of a Boolean function, *e.g.,* for a component S_b if all $\mathcal{W}_S(a,b)$ is divisible by $2^3 = 8$, then $\deg(S_b) \leq n + 1 - 3 = 2$. Thus, if there is a column in the LAT with either 0 or 8, then the component corresponding to this column is quadratic.

Exclusive Criteria

1. An optimal 4-bit S-box whose components are all with the maximal algebraic degree cannot have a large differential branch number. Precisely, $\mathrm{Deg}_{\mathrm{Freq}} = 15 \implies \mathcal{U}_1 > 0$.

   Actually, both the number of low Hamming weight differentials and the number of low Hamming weight linear approximations, cannot be very small at the same time. Precisely, $\mathrm{Deg}_{\mathrm{Freq}} = 15 \implies \mathrm{CardD1} \geq 1$ and $\mathrm{CardL1} \geq 4$.

   Furthermore, according to the above observation on inclusive criteria, all criteria in the set $\{\mathcal{U}_{\mathrm{Freq}} = 15,\ \mathcal{L}_{\mathrm{Freq}} = 30,\ \mathrm{Deg}_{\mathrm{Freq}} = 15,\ \#\mathrm{LS} = 0,\ \text{is not } (3,1)\text{-linear}\}$ conflict with the requirement $\mathrm{CardD1} = 0$ or $\mathrm{CardL1} < 4$.

[11] Note that, 2 (resp. 4) is the minimal value the CardL1 (resp. $\mathcal{L}_1$) of an optimal S-box can take.

As shown by Boura and Canteaut [BC13a] (resp. [BC16]), if the targeted cipher Hamsi (resp. PRESENT) attacked by their cube-like attack (resp. division-property-based integral attack) uses a 4-bit S-box whose components are all with maximal algebraic degree (*e.g.,* PRINCE S-box), then those attacks can be prevented (see Prop. 12 and Prop. 9). This may imply that, the criterion that all components have the maximal degree should be included in the design criteria of PRESENT S-box and Hamsi S-box (uses Serpent S2). However, $\mathcal{U}_1 = 0$ which is incompatible with this criterion is an explicit requirement for these two S-boxes, because the linear layer of PRESENT and Hamsi involves bit-permutations.

Accordingly, a careful trade-off is required between the resistance to the cube-like attack (division-property-based integral attack) and the resistance to the differential attack. One possible tradeoff is to relax the requirement on algebraic degree to be "all components but one have the maximal algebraic degree" and reserve the requirement $\mathcal{U}_1 = 0$. This tradeoff will result in some Serpent-type S-boxes [LP07] and the Golden S-boxes [Saa11]. Another possible tradeoff is to relax the requirement on $\mathcal{U}_1 = 0$ to be "with small $\mathcal{U}_1$, CardD1, $\mathcal{L}_1$, CardL1" and keep the requirement on algebraic degree. Experimental results show that the best achievable is $\mathrm{Deg}_{\mathrm{Freq}} = 15$, $\mathcal{U}_1 = 2$, CardD1 $= 6$, $\mathcal{L}_1 = 4$, CardL1 $= 4$. There are only *2* PXE-classes among optimal S-boxes possessing this sense of optimality. They belong to G_{13} (note that, G_{13} forms a CCZ-class containing only itself among the 16 optimal S-boxes).

2. A 4-bit S-box whose components all have the maximal algebraic degree cannot possess low multiplication complexity. This follows intuition. Precisely, we find all existing S-boxes with more than 3 quadratic components of $\mathtt{MC} = 4$; with 1 quadratic components of $\mathtt{MC} = 4$ or 5; with no quadratic components of $\mathtt{MC} = 5$.

3. An involution S-box cannot avoid a low hamming weight differential, *i.e.,* $S(S(x)) = x$ for $\forall x \implies \mathcal{U}_1 > 0$.

6 Conclusion and Future Work

The research on S-boxes closely interweaves with several other research lines, including attacks, Boolean functions, coding theories, and designs. The line of attacks imposes firm lower bounds on the acceptable properties; the lines of Boolean functions and coding theories provide idealistic upper bounds on the quality of being (almost) perfect; and the line of designs performs the art of difficult trade-offs. With this interweaving, the research on S-boxes has steadily progressed.

This paper provides a record on this progression. The main line of this record is following the line of attacks, in a consideration that significant attacks have almost always triggered a deep introspection on the design, especially on the design of S-boxes. Although, clear criteria can sometimes be obtained to advise new designs (*e.g.,* the differential uniformity and linearity of S-boxes), at other times, it is hard to clarify the exact property playing the key role (*e.g.,* for the division-property-based integral attack). Sometimes, the community takes a long time to fully understand how some known properties influence a striking attack (*e.g.,* the linear structures on the truncated differential and subspace trail attacks). Besides, efficient dedicated attacks generally exploit resonance properties between multiple components in a cipher. Thus, it is even harder to conclude a just right criterion for avoiding attacks. Under this situation, we tried to extract and distill known results reflected in studies on various attacks, present the proposed or implied criteria for avoiding the attacks, and form a comprehensive check-list for designers.

Besides, a platform PEIGEN is built, as a prototype in its embryonic stage, aiming to provide the community an open platform to facilitate the research and use of S-boxes. The

hope is to promote the development and further to provide feedback to these interweaving lines of research. PEIGEN is said to be in its embryonic stage, because, for larger S-boxes ($\geq$ 5-bit), it is satisfactory only for evaluating security, but not yet powerful enough for implementing and generating strong S-boxes. We believe that both heuristic and theoretical approaches exist and can be integrated into this platform. At this point, a call for further joint effort is initiated.

All the source codes of PEIGEN and generated results are available via https://github.com/peigen-sboxes/PEIGEN.

Acknowledgments

The authors would like to thank the anonymous reviewers of FSE 2019 for their detailed comments and valuable suggestions which helped improve the manuscript significantly. Special thanks go to Stefan Kölbl for his thorough proofreading and valuable comments. We would like to also thank Yongqiang Li, who helped to test which CCZ-equivalent class the 4-bit optimal S-box G_{13} belongs to. It turns out that G_{13} is not CCZ-equivalent with any other representatives of classes of optimal 4-bit S-boxes. This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Strategic Capability Research Centres Funding Initiative, Singapore Ministry of Education under Research Grants MOE2016-T2-2-014(S) and M4012049, and Nanyang Technological University under Grants M4080456 and M4082123.

References

[ARS+15] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In Oswald and Fischlin [OF15], pages 430–454.

[BBI+15] Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A Block Cipher for Low Energy. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, volume 9453 of *LNCS*, pages 411–436. Springer, 2015.

[BC13a] Christina Boura and Anne Canteaut. A New Criterion for Avoiding the Propagation of Linear Relations through an Sbox (Full version). *IACR Cryptology ePrint Archive*, 2013:211, 2013.

[BC13b] Christina Boura and Anne Canteaut. On the Influence of the Algebraic Degree of F^{-1} on the Algebraic Degree of $G \circ F$. *IEEE Trans. Information Theory*, 59(1):691–702, 2013.

[BC16] Christina Boura and Anne Canteaut. Another View of the Division Property. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, volume 9814 of *LNCS*, pages 654–682. Springer, 2016.

[BC18] Christina Boura and Anne Canteaut. On the Boomerang Uniformity of Cryptographic Sboxes. *IACR Transactions on Symmetric Cryptology*, 2018(3):290–310, Sep. 2018.

[BCBP03] Alex Biryukov, Christophe De Cannière, An Braeken, and Bart Preneel. A Toolbox for Cryptanalysis: Linear and Affine Equivalence Algorithms. In Eli Biham, editor, *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *LNCS*, pages 33–50. Springer, 2003.

[BCC10] Céline Blondeau, Anne Canteaut, and Pascale Charpin. Differential Properties of Power Functions. *IJICoT*, 1(2):149–170, 2010.

[BCC11] Christina Boura, Anne Canteaut, and Christophe De Cannière. Higher-Order Differential Properties of Keccak and *Luffa*. In Antoine Joux, editor, *Fast Software Encryption - 18th International Workshop, FSE 2011, Lyngby, Denmark, February 13-16, 2011, Revised Selected Papers*, volume 6733 of *LNCS*, pages 252–269. Springer, 2011.

[BCG+12] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçin. PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications - Extended Abstract. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2012.

[BCJS18] Christina Boura, Anne Canteaut, Jérémy Jean, and Valentin Suder. Two Notions of Differential Equivalence on Sboxes. *IACR Cryptology ePrint Archive*, 2018:617, 2018.

[BCLR17] Christof Beierle, Anne Canteaut, Gregor Leander, and Yann Rotella. Proving Resistance Against Invariant Attacks: How to Choose the Round Constants. In Katz and Shacham [KS17], pages 647–678.

[BDMW10] KA Browning, JF Dillon, MT McQuistan, and AJ Wolfe. An APN permutation in dimension six. *Finite Fields: theory and applications*, 518:33–42, 2010.

[BDPVA] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. The Keccak Reference, version 3.0 (2011). *URL: http://keccak.noekeon. org/Keccak-reference-3.0.pdf*, 4.

[Ber05] Daniel J. Bernstein. Cache-Timing Attacks on AES, 2005. https://cr.yp.to/antiforgery/cachetiming-20050414.pdf.

[Bih97] Eli Biham. A Fast New DES Implementation in Software. In Eli Biham, editor, *Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997, Proceedings*, volume 1267 of *LNCS*, pages 260–272. Springer, 1997.

[BJK+16] 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 Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 123–153. Springer, 2016.

[BKL+07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *LNCS*, pages 450–466. Springer, 2007.

[BL08] Marcus Brinkmann and Gregor Leander. On the Classification of APN Functions up to Dimension Five. *Des. Codes Cryptography*, 49(1-3):273–288, 2008.

[BLL15] Zhenzhen Bao, Peng Luo, and Dongdai Lin. Bitsliced Implementations of the PRINCE, LED and RECTANGLE Block Ciphers on AVR 8-bit Microcontrollers. *IACR Cryptology ePrint Archive*, 2015:1118, 2015.

[BMP13] Joan Boyar, Philip Matthews, and René Peralta. Logic Minimization Techniques with Applications to Cryptology. *J. Cryptology*, 26(2):280–312, 2013.

[BN13] Céline Blondeau and Kaisa Nyberg. New Links between Differential and Linear Cryptanalysis. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *LNCS*, pages 388–404. Springer, 2013.

[BNN+12] Begül Bilgin, Svetla Nikova, Ventzislav Nikov, Vincent Rijmen, and Georg Stütz. Threshold Implementations of All 3×3 and 4×4 S-Boxes. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *LNCS*, pages 76–91. Springer, 2012.

[BP10] Joan Boyar and René Peralta. A New Combinational Logic Minimization Technique with Applications to Cryptology. In Paola Festa, editor, *Experimental Algorithms, 9th International Symposium, SEA 2010, Ischia Island, Naples, Italy, May 20-22, 2010. Proceedings*, volume 6049 of *LNCS*, pages 178–189. Springer, 2010.

[BP12] Joan Boyar and René Peralta. A small depth-16 circuit for the AES s-box. In Dimitris Gritzalis, Steven Furnell, and Marianthi Theoharidou, editors, *Information Security and Privacy Research - 27th IFIP TC 11 Information Security and Privacy Conference, SEC 2012, Heraklion, Crete, Greece, June 4-6, 2012. Proceedings*, volume 376 of *IFIP Advances in Information and Communication Technology*, pages 287–298. Springer, 2012.

[BPP00] Joan Boyar, René Peralta, and Denis Pochuev. On the multiplicative complexity of boolean functions over the basis $(\wedge, \oplus, 1)$. *Theor. Comput. Sci.*, 235(1):43–57, 2000.

[BPP+17] Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT: A Small Present - Towards Reaching the Limit of Lightweight Encryption. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *LNCS*, pages 321–345. Springer, 2017.

[BPU16] Alex Biryukov, Léo Perrin, and Aleksei Udovenko. Reverse-Engineering the S-Box of Streebog, Kuznyechik and STRIBOBr1. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, volume 9665 of *LNCS*, pages 372–402. Springer, 2016.

[BS90] Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In Alfred Menezes and Scott A. Vanstone, editors, *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, volume 537 of *LNCS*, pages 2–21. Springer, 1990.

[Can16] Anne Canteaut. Lecture Notes on Cryptographic Boolean Functions. *Inria, Paris, France*, 2016. https://www.paris.inria.fr/secret/Anne.Canteaut/poly.pdf.

[Car93] Claude Carlet. Two new classes of bent functions. In Helleseth [Hel94], pages 77–101.

[Car10a] Claude Carlet. Boolean Functions for Cryptography and Error Correcting Codes. *Boolean models and methods in mathematics, computer science, and engineering*, 2:257–397, 2010.

[Car10b] Claude Carlet. Vectorial Boolean Functions for Cryptography. *Boolean models and methods in mathematics, computer science, and engineering*, 134:398–469, 2010.

[CCCF01] Anne Canteaut, Claude Carlet, Pascale Charpin, and Caroline Fontaine. On Cryptographic Properties of the Cosets of $R(1, m)$. *IEEE Trans. Information Theory*, 47(4):1494–1513, 2001.

[CCZ98] Claude Carlet, Pascale Charpin, and Victor A. Zinoviev. Codes, Bent Functions and Permutations Suitable For DES-like Cryptosystems. *Des. Codes Cryptography*, 15(2):125–156, 1998.

[CE85] David Chaum and Jan-Hendrik Evertse. Crytanalysis of DES with a Reduced Number of Rounds: Sequences of Linear Factors in Block Ciphers. In Hugh C. Williams, editor, *Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*, volume 218 of *Lecture Notes in Computer Science*, pages 192–211. Springer, 1985.

[CHM11] Nicolas Courtois, Daniel Hulme, and Theodosis Mourouzis. Solving Circuit Optimisation Problems in Cryptography and Cryptanalysis. *IACR Cryptology ePrint Archive*, 2011:475, 2011.

[Cho10] Joo Yeon Cho. Linear Cryptanalysis of Reduced-Round PRESENT. In Josef Pieprzyk, editor, *Topics in Cryptology - CT-RSA 2010, The Cryptographers' Track at the RSA Conference 2010, San Francisco, CA, USA, March 1-5, 2010. Proceedings*, volume 5985 of *LNCS*, pages 302–317. Springer, 2010.

[CHP+18] Carlos Cid, Tao Huang, Thomas Peyrin, Yu Sasaki, and Ling Song. Boomerang Connectivity Table: A New Cryptanalysis Tool. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, volume 10821 of *LNCS*, pages 683–714. Springer, 2018.

[CP18] Anne Canteaut and Léo Perrin. On CCZ-Equivalence, Extended-Affine Equivalence, and Function Twisting. *IACR Cryptology ePrint Archive*, 2018:713, 2018.

[CR15] Anne Canteaut and Joëlle Roué. On the behaviors of affine equivalent sboxes regarding differential and linear attacks. In Oswald and Fischlin [OF15], pages 45–74.

[CV94] Florent Chabaud and Serge Vaudenay. Links Between Differential and Linear Cryptanalysis. In Alfredo De Santis, editor, *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, volume 950 of *LNCS*, pages 356–365. Springer, 1994.

[CV02] Anne Canteaut and Marion Videau. Degree of Composition of Highly Nonlinear Functions and Applications to Higher Order Differential Cryptanalysis. In Lars R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, volume 2332 of *LNCS*, pages 518–533. Springer, 2002.

[DCSW08] Christophe De Canniere, Hisayoshi Sato, and Dai Watanabe. Hash Function Luffa: Specification. *Submission to NIST SHA-3 Competition*, 2008.

[DEMS16] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1. 2. *Submission to the CAESAR Competition*, 2016.

[DGV94] Joan Daemen, René Govaerts, and Joos Vandewalle. Correlation Matrices. In Preneel [Pre95], pages 275–285.

[DH18] Orr Dunkelman and Senyang Huang. Reconstructing an S-box from its Difference Distribution Table. *IACR Cryptology ePrint Archive*, 2018:811, 2018.

[Din18] Itai Dinur. An Improved Affine Equivalence Algorithm for Random Permutations. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part I*, volume 10820 of *LNCS*, pages 413–442. Springer, 2018.

[DKS10] Orr Dunkelman, Nathan Keller, and Adi Shamir. A Practical-Time Related-Key Attack on the KASUMI Cryptosystem Used in GSM and 3G Telephony. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *LNCS*, pages 393–410. Springer, 2010.

[DMP+15] Itai Dinur, Pawel Morawiecki, Josef Pieprzyk, Marian Srebrny, and Michal Straus. Cube Attacks and Cube-Attack-Like Cryptanalysis on the Round-Reduced Keccak Sponge Function. In Oswald and Fischlin [OF15], pages 733–761.

[Dob94] Hans Dobbertin. Construction of Bent Functions and Balanced Boolean Functions with High Nonlinearity. In Preneel [Pre95], pages 61–74.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[DS09] Itai Dinur and Adi Shamir. Cube Attacks on Tweakable Black Box Polynomials. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *LNCS*, pages 278–299. Springer, 2009.

[Dub01] Sylvie Dubuc. Characterization of Linear Structures. *Des. Codes Cryptography*, 22(1):33–45, 2001.

[Dun18] Orr Dunkelman. Efficient Construction of the Boomerang Connection Table. *IACR Cryptology ePrint Archive*, 2018:631, 2018.

[Eve87] Jan-Hendrik Evertse. Linear Structures in Blockciphers. In David Chaum and Wyn L. Price, editors, *Advances in Cryptology - EUROCRYPT '87, Workshop on the Theory and Application of of Cryptographic Techniques, Amsterdam, The Netherlands, April 13-15, 1987, Proceedings*, volume 304 of *LNCS*, pages 249–266. Springer, 1987.

[FJ] Jean-Pierre Flori and Jérémy Jean. libapn. https://github.com/ANSSI-FR/libapn. Latest commit on Apr 2018.

[Fuh10] Thomas Fuhr. Finding Second Preimages of Short Messages for Hamsi-256. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *LNCS*, pages 20–37. Springer, 2010.

[GJN+16] Jian Guo, Jérémy Jean, Ivica Nikolic, Kexin Qiao, Yu Sasaki, and Siang Meng Sim. Invariant Subspace Attack Against Midori64 and The Resistance Criteria for S-box Designs. *IACR Trans. Symmetric Cryptol.*, 2016(1):33–56, 2016.

[Gla] Brian Gladman. Finding Efficient Boolean Function Decompositions for the Serpent S-boxes and Their Inverses. http://brg.a2hosted.com//oldsite/cryptography_technology/serpent/anal1.cpp. Accessed: 2018-11.

[GLS16] Jian Guo, Meicheng Liu, and Ling Song. Linear Structures: Applications to Cryptanalysis of Round-Reduced Keccak. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *LNCS*, pages 249–274, 2016.

[GLSV14] Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, and Kerem Varici. LS-Designs: Bitslice Encryption for Efficient Masked Software Implementations. In Carlos Cid and Christian Rechberger, editors, *Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers*, volume 8540 of *Lecture Notes in Computer Science*, pages 18–37. Springer, 2014.

[GRR16] Lorenzo Grassi, Christian Rechberger, and Sondre Rønjom. Subspace Trail Cryptanalysis and its Applications to AES. *IACR Trans. Symmetric Cryptol.*, 2016(2):192–225, 2016.

[GRW16] Faruk Göloglu, Vincent Rijmen, and Qingju Wang. On the Division Property of S-boxes. *IACR Cryptology ePrint Archive*, 2016:188, 2016.

[Hel94] Tor Helleseth, editor. *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, volume 765 of *Lecture Notes in Computer Science*. Springer, 1994.

[HLL+00] Seokhie Hong, Sangjin Lee, Jongin Lim, Jaechul Sung, Dong Hyeon Cheon, and Inho Cho. Provable Security against Differential and Linear Cryptanalysis for the SPN Structure. In Bruce Schneier, editor, *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*, volume 1978 of *LNCS*, pages 273–283. Springer, 2000.

[JK01] Thomas Jakobsen and Lars R. Knudsen. Attacks on Block Ciphers of Low Algebraic Degree. *J. Cryptology*, 14(3):197–210, 2001.

[JPST17] Jérémy Jean, Thomas Peyrin, Siang Meng Sim, and Jade Tourteaux. Optimizing Implementations of Lightweight Building Blocks. *IACR Trans. Symmetric Cryptol.*, 2017(4):130–168, 2017.

[JS17] Anthony Journault and François-Xavier Standaert. Very High Order Masking: Efficient Implementation and Security Evaluation. In Wieland Fischer and Naofumi Homma, editors, *CHES 2017*, volume 10529 of *LNCS*, pages 623–643. Springer, 2017.

[KLPR10] Lars R. Knudsen, Gregor Leander, Axel Poschmann, and Matthew J. B. Robshaw. PRINTcipher: A Block Cipher for IC-Printing. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *LNCS*, pages 16–32. Springer, 2010.

[Knu94] Lars R. Knudsen. Truncated and Higher Order Differentials. In Preneel [Pre95], pages 196–211.

[KS17] Jonathan Katz and Hovav Shacham, editors. *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, volume 10402 of *LNCS*. Springer, 2017.

[LAAZ11] Gregor Leander, Mohamed Ahmed Abdelraheem, Hoda AlKhzaimi, and Erik Zenner. A Cryptanalysis of PRINTcipher: The Invariant Subspace Attack. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *LNCS*, pages 206–221. Springer, 2011.

[Lai94] Xuejia Lai. Additive and Linear Structures of Cryptographic Functions. In Preneel [Pre95], pages 75–85.

[LMS+15] Mario Lamberger, Florian Mendel, Martin Schläffer, Christian Rechberger, and Vincent Rijmen. The Rebound Attack and Subspace Distinguishers: Application to Whirlpool. *J. Cryptology*, 28(2):257–296, 2015.

[LP07] Gregor Leander and Axel Poschmann. On the Classification of 4 Bit S-Boxes. In Claude Carlet and Berk Sunar, editors, *Arithmetic of Finite Fields, First International Workshop, WAIFI 2007, Madrid, Spain, June 21-22, 2007, Proceedings*, volume 4547 of *LNCS*, pages 159–176. Springer, 2007.

[LR18] Yunwen Liu and Vincent Rijmen. New Observations on Invariant Subspace Attack. *Inf. Process. Lett.*, 138:27–30, 2018.

[LTW18] Gregor Leander, Cihangir Tezcan, and Friedrich Wiemer. Searching for Subspace Trails and Truncated Differentials. *IACR Trans. Symmetric Cryptol.*, 2018(1):74–100, 2018.

[Mag] Magma Computational Algebra System. http://magma.maths.usyd.edu.au. Accessed: 2018-11.

[Mat93] Mitsuru Matsui. Linear Cryptanalysis Method for DES Cipher. In Helleseth [Hel94], pages 386–397.

[MLCA] Rusydi H. Makarim, Yann Laigle-Chapuy, and Martin R. Albrecht. SageMath 8.6: sage.crypto.sbox. http://doc.sagemath.org/html/en/reference/cryptography/sage/crypto/sbox.html. Accessed: 2019-03.

[MRST09] Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl. In Orr Dunkelman, editor, *FSE 2009*, volume 5665 of *LNCS*, pages 260–276. Springer, 2009.

[MS89] Willi Meier and Othmar Staffelbach. Nonlinearity Criteria for Cryptographic Functions. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Advances in Cryptology - EUROCRYPT '89, Workshop on the Theory and Application of of Cryptographic Techniques, Houthalen, Belgium, April 10-13, 1989, Proceedings*, volume 434 of *LNCS*, pages 549–562. Springer, 1989.

[MT14] Rusydi H. Makarim and Cihangir Tezcan. Relating Undisturbed Bits to Other Properties of Substitution Boxes. In Thomas Eisenbarth and Erdinç Öztürk, editors, *Lightweight Cryptography for Security and Privacy - Third International Workshop, LightSec 2014, Istanbul, Turkey, September 1-2, 2014, Revised Selected Papers*, volume 8898 of *LNCS*, pages 109–125. Springer, 2014.

[NK92] Kaisa Nyberg and Lars R. Knudsen. Provable Security Against Differential Cryptanalysis. In Ernest F. Brickell, editor, *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, volume 740 of *LNCS*, pages 566–574. Springer, 1992.

[Nyb91] Kaisa Nyberg. Perfect Nonlinear S-Boxes. In Donald W. Davies, editor, *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*, volume 547 of *LNCS*, pages 378–386. Springer, 1991.

[Nyb93] Kaisa Nyberg. Differentially Uniform Mappings for Cryptography. In Helleseth [Hel94], pages 55–64.

[Nyb94] Kaisa Nyberg. S-boxes and Round Functions with Controllable Linearity and Differential Uniformity. In Preneel [Pre95], pages 111–130.

[OF15] Elisabeth Oswald and Marc Fischlin, editors. *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *LNCS*. Springer, 2015.

[Ohk09] Kenji Ohkuma. Weak Keys of Reduced-Round PRESENT for Linear Cryptanalysis. In Michael J. Jacobson Jr., Vincent Rijmen, and Reihaneh Safavi-Naini, editors, *Selected Areas in Cryptography, 16th Annual International Workshop, SAC 2009, Calgary, Alberta, Canada, August 13-14, 2009, Revised Selected Papers*, volume 5867 of *LNCS*, pages 249–265. Springer, 2009.

[Osv00] Dag Arne Osvik. Speeding up Serpent. In *AES Candidate Conference*, pages 317–329, 2000.

[Per17] Léo Perrin. *Cryptanalysis, Reverse-Engineering and Design of Symmetric Cryptographic Algorithms*. PhD thesis, University of Luxembourg, 2017.

[PLL+90] Bart Preneel, Werner Van Leekwijck, Luc Van Linden, René Govaerts, and Joos Vandewalle. Propagation Characteristics of Boolean Functions. In Ivan Damgård, editor, *Advances in Cryptology - EUROCRYPT '90, Workshop on the Theory and Application of of Cryptographic Techniques, Aarhus, Denmark, May 21-24, 1990, Proceedings*, volume 473 of *LNCS*, pages 161–173. Springer, 1990.

[PMA07] Lea Troels Møoller Pedersen, Carsten Valdemar Munk, and Lisbet Møoller Andersen. Cryptography - The Rise and Fall of DVD Encryption, 2007.

[PRC12] Gilles Piret, Thomas Roche, and Claude Carlet. PICARO - A Block Cipher Allowing Efficient Higher-Order Side-Channel Resistance. In Feng Bao, Pierangela Samarati, and Jianying Zhou, editors, *ACNS 2012*, volume 7341 of *LNCS*, pages 311–328. Springer, 2012.

[Pre95] Bart Preneel, editor. *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings*, volume 1008 of *LNCS*. Springer, 1995.

[PSLL03] Sangwoo Park, Soo Hak Sung, Sangjin Lee, and Jongin Lim. Improving the Upper Bound on the Maximum Differential and the Maximum Linear Hull Probability for SPN Structures and AES. In Thomas Johansson, editor, *Fast Software Encryption, 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003, Revised Papers*, volume 2887 of *LNCS*, pages 247–260. Springer, 2003.

[PW] Léo Perrin and Friedrich Wiemer. SageMath 8.6: sage.crypto.sboxes. http://doc.sagemath.org/html/en/reference/cryptography/sage/crypto/sboxes.html. Accessed: 2019-03.

[Qia] Kexin Qiao. Sbox-depth-evaluation. https://github.com/qiaokexin/Sbox-depth-evaluation. Latest commit on Jun 2016.

[QSLG17] Kexin Qiao, Ling Song, Meicheng Liu, and Jian Guo. New Collision Attacks on Round-Reduced Keccak. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part III*, volume 10212 of *LNCS*, pages 216–243, 2017.

[RA98] Lars Knudsen Ross Anderson, Eli Biham. Serpent: A Proposal for the Advanced Encryption Standard. In *First Advanced Encryption Standard (AES) Conference, Ventura, CA*, 1998.

[Rot76] O. S. Rothaus. On "Bent" Functions. *J. Comb. Theory, Ser. A*, 20(3):300–305, 1976.

[Saa11] Markku-Juhani O. Saarinen. Cryptographic Analysis of All 4×4-Bit S-Boxes. In Ali Miri and Serge Vaudenay, editors, *Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers*, volume 7118 of *LNCS*, pages 118–133. Springer, 2011.

[SLG17] Ling Song, Guohong Liao, and Jian Guo. Non-full Sbox Linearization: Applications to Collision Attacks on Round-Reduced Keccak. In Katz and Shacham [KS17], pages 428–451.

[Sto16] Ko Stoffelen. Optimizing S-Box Implementations for Several Criteria Using SAT Solvers. In Thomas Peyrin, editor, *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, volume 9783 of *LNCS*, pages 140–160. Springer, 2016.

[SWW18] Ling Sun, Wei Wang, and Meiqin Wang. More Accurate Differential Properties of LED64 and Midori64. *IACR Transactions on Symmetric Cryptology*, 2018(3):93–123, Sep. 2018.

[Tez14] Cihangir Tezcan. Improbable Differential Attacks on PRESENT Using Undisturbed Bits. *J. Computational Applied Mathematics*, 259:503–511, 2014.

[Tez16] Cihangir Tezcan. Truncated, Impossible, and Improbable Differential Analysis of ASCON. In Olivier Camp, Steven Furnell, and Paolo Mori, editors, *Proceedings of the 2nd International Conference on Information Systems Security and Privacy, ICISSP 2016, Rome, Italy, February 19-21, 2016.*, pages 325–332. SciTePress, 2016.

[TLS16] Yosuke Todo, Gregor Leander, and Yu Sasaki. Nonlinear Invariant Attack - Practical Attack on Full SCREAM, iSCREAM, and Midori64. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II*, volume 10032 of *LNCS*, pages 3–33, 2016.

[Tod15a] Yosuke Todo. Integral Cryptanalysis on Full MISTY1. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *LNCS*, pages 413–432. Springer, 2015.

[Tod15b] Yosuke Todo. Structural Evaluation by Generalized Integral Property. In Oswald and Fischlin [OF15], pages 287–314.

[TTD14] Cihangir Tezcan, Halil Kemal Taskin, and Murat Demircioglu. Improbable Differential Attacks on Serpent using Undisturbed Bits. In Ron Poet and Muttukrishnan Rajarajan, editors, *Proceedings of the 7th International Conference on Security of Information and Networks, Glasgow, Scotland, UK, September 9-11, 2014*, page 145. ACM, 2014.

[UDCI+11] Markus Ullrich, Christophe De Canniere, Sebastiaan Indesteege, Özgül Küçük, Nicky Mouha, and Bart Preneel. Finding Optimal Bitsliced Implementations of $4\times$ 4-bit S-boxes. In *SKEW 2011 Symmetric Key Encryption Workshop, Copenhagen, Denmark*, pages 16–17, 2011.

[Wag99] David A. Wagner. The Boomerang Attack. In Lars R. Knudsen, editor, *Fast Software Encryption, 6th International Workshop, FSE '99, Rome, Italy, March 24-26, 1999, Proceedings*, volume 1636 of *LNCS*, pages 156–170. Springer, 1999.

[WHYK10] Dai Watanabe, Yasuo Hatano, Tsuyoshi Yamada, and Toshinobu Kaneko. Higher Order Differential Attack on Step-Reduced Variants of *Luffa* v1. In Seokhie Hong and Tetsu Iwata, editors, *Fast Software Encryption, 17th International Workshop, FSE 2010, Seoul, Korea, February 7-10, 2010, Revised Selected Papers*, volume 6147 of *LNCS*, pages 270–285. Springer, 2010.

[WS10] Dai Watanabe and Hitachi SDL. How to Generate the S-box of Luffa. In *Early Symmetric Crypto Seminar, ESC2010*, 2010.

[ZBL+15] Wentao Zhang, Zhenzhen Bao, Dongdai Lin, Vincent Rijmen, Bohan Yang, and Ingrid Verbauwhede. RECTANGLE: A Bit-slice Lightweight Block Cipher Suitable for Multiple Platforms. *SCIENCE CHINA Information Sciences*, 58(12):1–15, 2015.

[ZBRL15] Wentao Zhang, Zhenzhen Bao, Vincent Rijmen, and Meicheng Liu. A New Classification of 4-bit Optimal S-boxes and Its Application to PRESENT, RECTANGLE and SPONGENT. In Gregor Leander, editor, *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, volume 9054 of *LNCS*, pages 494–515. Springer, 2015.

[ZZI00] Xian-Mo Zhang, Yuliang Zheng, and Hideki Imai. Relating Differential Distribution Tables to Other Properties of Substitution Boxes. *Des. Codes Cryptography*, 19(1):45–63, 2000.

A A Complete List of Notations

$\oplus, \bigoplus$ and $+, \sum$	To make a distinction, we use $\oplus, \bigoplus$ to represent *addition* and *summation* of $\mathbb{F}_2^n$, and use $+, \sum$ to represent addition and summation of $\mathbb{Z}$.	-
a	A *binary vector*, for $a \in \mathbb{F}_2^n$, $a = (a_0, a_1, \ldots, a_{n-1})$ where $a_i \in \mathbb{F}_2$ is the *coordinate* of a with index i.	-
$\mathrm{wt}(a)$	The *Hamming weight (or simply, weight)* of a binary vector $a \in \mathbb{F}_2^n$, $\mathrm{wt}(a) \triangleq \sum_{i=1}^{n} a_i$.	-
$\mathrm{supp}(a)$	The *support* of a binary vector $a \in \mathbb{F}_2^n$ is the set of all labels i such that $a_i \neq 0$.	-
$a \cdot b$	The *inner product* of two binary vectors $a, b \in \mathbb{F}_2^n$, $a \cdot b \triangleq \bigoplus_{i=1}^{n} a_i \cdot b_i$.	-
$a \preceq b$	For two binary vectors $a, b \in \mathbb{F}_2^n$, $a \preceq b$ if and only if $a_i \leq b_i$ for all $1 \leq i \leq n$.	-
$f : \mathbb{F}_2^n \to \mathbb{F}_2$	A *Boolean function* in n binary variables, mapping from $\mathbb{F}_2^n$ into $\mathbb{F}_2$. We also directly use f to denote the *value vector* of f, which is the vector corresponding to all values taken by the function when we use the lexicographical order on the inputs.	-
ANF_f	The *algebraic normal form* of $f : \mathbb{F}_2^n \to \mathbb{F}_2$. A Boolean function f can be uniquely represented by an n-variate polynomial in $\mathbb{F}_2[x_1, \cdots, x_n]/(x_1^2 \oplus x_1, \cdots, x_n^2 \oplus x_n)$: $f(x_1, \ldots, x_n) = \bigoplus_{u \in \mathbb{F}_2^n} \alpha_u \prod_{i=1}^{n} x_i^{u_i}$, where $\alpha_u \in \mathbb{F}_2$.	✔
φ_α, $\alpha \in \mathbb{F}_2^n$	The *linear Boolean function* $x \mapsto \alpha \cdot x$. The algebraic normal form of φ_α is $\varphi_\alpha(x_1, \ldots, x_n) = \bigoplus_{i=1}^{n} \alpha_i \cdot x_i$.	-
A_n	The set of all n-variable affine Boolean functions $A_n \triangleq \{\varphi_\alpha(x_1, \ldots, x_n) = \bigoplus_{i=1}^{n} \alpha_i \cdot x_i \oplus \alpha_0 \mid a_0, \ldots, a_n \in \mathbb{F}_2\}$.	-
$S : \mathbb{F}_2^n \to \mathbb{F}_2^m$	A *vectorial Boolean function* (an $n \times m$ *S-box*) mapping n bits to m bits.	-
ANF_S	The *algebraic normal form* of $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$. A vectorial Boolean function S can be uniquely represented by a n-variate polynomial in $\mathbb{F}_2^m[x_1, \cdots, x_n]/(x_1^2 \oplus x_1, \cdots, x_n^2 \oplus x_n)$: $S(x_1, \ldots, x_n) = \bigoplus_{u \in \mathbb{F}_2^n} \alpha_u \prod_{i=1}^{n} x_i^{u_i}$, where $\alpha_u \in \mathbb{F}_2^m$.	✔
$\{e_i \mid i \in \{1, \cdots, m\}\}$	The *standard basis* for $\mathbb{F}_2^m$.	-
$S_{e_i} : \mathbb{F}_2^n \to \mathbb{F}_2$ $x \mapsto e_i \cdot S(x)$	The *coordinate function (or simply, coordinate)* of a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ for index $1 \leq i \leq m$, which is the Boolean function representing the i-th output bit of S.	✔
$S_\lambda : \mathbb{F}_2^n \to \mathbb{F}_2$ $x \mapsto \lambda \cdot S(x)$	The *component function (or simply, component)* of a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ for nonzero vector $\lambda \in \mathbb{F}_2^m$, which is a Boolean function representing the linear combination $\bigoplus_{i=1}^{m} \lambda_i S_{e_i}$ of the coordinate functions of S.	✔

$D_af : \mathbb{F}_2^n \to \mathbb{F}_2$ $x \mapsto f(x) \oplus f(x \oplus a)$	The *derivative function (or simply, derivative)* of a Boolean function f. $D_af(x) \triangleq f(x) \oplus f(x \oplus a),\ a \in \mathbb{F}_2^n$.	-
$\delta_f(a,b)$, $a \in \mathbb{F}_2^n,\ b \in \mathbb{F}_2$	$\delta_f(a,b) \triangleq \#\{x \in \mathbb{F}_2^n \mid f(x) \oplus f(x \oplus a) = b\} = \lvert D_af^{-1}(b)\rvert$.	-
$r_f(a)$	The autocorrelation coefficient of a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ on $a \in \mathbb{F}_2^n$, $r_f(a) \triangleq \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x) \oplus f(x \oplus a)}$.	-
ACT_S	The autocorrelation table, in which the elements in row a and column λ is equal to autocorrelation coefficient of the component function S_λ on $a \in \mathbb{F}_2^n$, $\mathrm{ACT}_S(a,\lambda) = r_{S_\lambda}(a)$.	✔
$D_aS : \mathbb{F}_2^n \to \mathbb{F}_2^m$ $x \mapsto S(x) \oplus S(x \oplus a)$	The *derivative function (or simply, derivative)* of a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ to the direction $a \in \mathbb{F}_2^n$. $D_aS(x) \triangleq S(x) \oplus S(x \oplus a),\ a \in \mathbb{F}_2^n$.	-
$\delta_S(a,b)$, $a \in \mathbb{F}_2^n,\ b \in \mathbb{F}_2^m$	$\delta_S(a,b) \triangleq \#\{x \in \mathbb{F}_2^n \mid S(x) \oplus S(x \oplus a) = b\} = \#\{D_af^{-1}(b)\}$, where $D_af^{-1}(b)$ means the set of preimages of b under the derivative function D_af.	-
DDT	The *differential distribution table* of an S-box S (a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$). The entry (a,b) in DDT_S equals $\delta_S(a,b)$ for all $a \in \mathbb{F}_2^n$ and $b \in \mathbb{F}_2^m$.	✔
$\mathcal{U}(S)$	The *differential uniformity* of a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$, $\mathcal{U}(S) \triangleq \max_{a \in \mathbb{F}_2^n \setminus \{0\}, b \in \mathbb{F}_2^m} \delta_S(a,b)$.	✔
$\mathcal{U}_{\mathrm{Freq}}(S)$	$\mathcal{U}_{\mathrm{Freq}}(S) \triangleq \#\{(a,b) \mid \delta_S(a,b) = \mathcal{U}(S),\ a \in \mathbb{F}_2^n \setminus \{0\},\ b \in \mathbb{F}_2^m\}$.	✔
$\mathcal{D}_{\mathrm{spec}}(S)$	The *differential spectrum* of a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is the multiset $\mathcal{D}_{\mathrm{spec}}(S) \triangleq \{\delta_S(a,b) \mid a \in \mathbb{F}_2^n \setminus \{0\}, b \in \mathbb{F}_2^m\}$.	✔
BCT	The *boomerang connectivity table* of an bijective S-box S (a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^n$). The entry (a,b) equals $\beta_S(a,b) \triangleq \#\{x \in \mathbb{F}_2^n \mid S^{-1}(S(x) \oplus b) \oplus S^{-1}(S(x \oplus a) \oplus b) = a\}$ for all $a \in \mathbb{F}_2^n$ and $b \in \mathbb{F}_2^n$.	✔
$\mathcal{BU}(S)$	The *boomerang uniformity* of an S-box S (a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^n$) is the highest value in the BCT excluding the entry $(0,0)$: $\mathcal{BU}(S) \triangleq \max_{a,b \in \mathbb{F}_2^n \setminus \{0\}} \beta_S(a,b)$.	✔
$\mathcal{BD}_{\mathrm{spec}}(S)$	The *boomerang differential spectrum* of a bijective S-box S (a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^n$) is the multiset $\mathcal{BD}_{\mathrm{spec}}(S) \triangleq \{\beta_S(a,b) \mid a \in \mathbb{F}_2^n, b \in \mathbb{F}_2^n\}$.	✔
$\varepsilon_S(\alpha,\beta)$	The bias of a linear approximation (α,β), *i.e.*, $\left\lvert \frac{\#\{x \mid S_\beta(x) = \alpha \cdot x\}}{2^n} - 1/2 \right\rvert$.	-
$\mathrm{d}(f,g)$	The Hamming distance between two Boolean functions f and g is the number of function values in which they differ: $\mathrm{d}(f,g) = \mathrm{wt}(f \oplus g)$.	-

$\mathcal{F}(f)$	The discrete *Fourier transform (aka., Walsh transform) at point* 0 of the sign function $(-1)^f$ of a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$. $\mathcal{F}(f) \triangleq \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)} = 2^n - 2\,\mathrm{wt}(f)$.	-
$\mathcal{W}_f : \mathbb{F}_2^n \to \mathbb{Z}$ $\alpha \mapsto \mathcal{F}(f \oplus \varphi_\alpha)$	The *Walsh transform (aka., Fourier transform)* of a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$, $\mathcal{W}_f(\alpha) \triangleq \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x) \oplus \alpha \cdot x} = \mathcal{F}(f \oplus \varphi_\alpha) = 2^n - 2\,\mathrm{d}(f, \varphi_\alpha)$, for $\alpha \in \mathbb{F}_2^n$. The value taken by the transform at point α is called the *Walsh coefficient* of f at point α.	-
$\mathcal{W}_{\mathrm{spec}}(f)$	The *Walsh spectrum (aka. Fourier spectrum)* of a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is the multiset $\mathcal{W}_{\mathrm{spec}}(f) \triangleq \{\mathcal{W}_f(\alpha) \mid \alpha \in \mathbb{F}_2^n\}$	-
$\mathcal{L}(f)$	The *linearity* of a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$, $\mathcal{L}(f) \triangleq \max_{\alpha \in \mathbb{F}_2^n} \lvert 2^n - 2\,\mathrm{wt}(f \oplus \varphi_\alpha)\rvert = \max_{\alpha \in \mathbb{F}_2^n} \lvert \mathcal{W}_f(\alpha)\rvert$	-
$\mathcal{NL}(f)$	The *nonlinearity* of a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is the minimum Hamming distance between f and all affine functions A_n. $\mathcal{NL}(f) \triangleq \min_{g \in A_n} \mathrm{d}(f, g) = \min_{\alpha \in \mathbb{F}_2^n} \lvert \mathrm{wt}(f \oplus \varphi_\alpha)\rvert =$ $2^{n-1} - \frac{1}{2} \max_{\alpha \in \mathbb{F}_2^n} \lvert \mathcal{W}_f(\alpha)\rvert = 2^{n-1} - \frac{1}{2}\,\mathcal{L}(f)$.	-
$\mathcal{W}_S : \mathbb{F}_2^n \times \mathbb{F}_2^m \to \mathbb{Z}$ $(\alpha, \beta) \mapsto \mathcal{F}(\beta \cdot S \oplus \varphi_\alpha)$	The *Walsh transform of a vectorial Boolean function* $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$, $\mathcal{W}_S(\alpha, \beta) = \mathcal{W}_{S_\beta}(\alpha) =$ $\sum_{x \in \mathbb{F}_2^n} (-1)^{\beta \cdot S(x) \oplus \alpha \cdot x}$, $\alpha \in \mathbb{F}_2^n$, $\beta \in \mathbb{F}_2^m$. The value taken by the transform at point (α, β) is called the *Walsh coefficient* of S at point (α, β).	✔
$\mathcal{W}_{\mathrm{spec}}(S)$	The *Walsh spectrum* of a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is the multi-set $\mathcal{W}_{\mathrm{spec}}(S) \triangleq \{\mathcal{W}_S(\alpha, \beta) \mid \alpha \in \mathbb{F}_2^n, \beta \in \mathbb{F}_2^m \setminus \{0\}\}$. The *extended Walsh spectrum* of S is the multi-set of the absolute of values in $\mathcal{W}_{\mathrm{spec}}(S)$.	✔
$\mathcal{L}(S)$	The *linearity* of a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is the maximum linearity of its nontrivial components $\{S_\beta \mid \beta \in \mathbb{F}_2^m \setminus \{0\}\}$. $\mathcal{L}(S) = \max_{\beta \in \mathbb{F}_2^m \setminus \{0\}} \mathcal{L}(S_\beta) = \max_{\alpha \in \mathbb{F}_2^n, \beta \in \mathbb{F}_2^m \setminus \{0\}} \lvert \mathcal{W}_S(\alpha, \beta)\rvert$.	✔
$\mathcal{NL}(S)$	The *nonlinearity* of a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is the Hamming distance between the set of its nontrivial components $\{S_\beta \mid \beta \in \mathbb{F}_2^m \setminus \{0\}\}$ and the set of all affine functions A_n. $\mathcal{NL}(S) = \min_{\beta \in \mathbb{F}_2^m \setminus \{0\}} \mathcal{NL}(S_\beta) = 2^{n-1} - \frac{1}{2}\,\mathcal{L}(S)$.	✔
LAT	The *linear approximation table* of an S-box S (a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$). The entry (α, β) in LAT_S equals $\mathcal{W}_S(\alpha, \beta)$ for all $\alpha \in \mathbb{F}_2^n$ and $\beta \in \mathbb{F}_2^m$.	✔
$\mathcal{L}_{\mathrm{Freq}}$	$\mathcal{L}_{\mathrm{Freq}} \triangleq \#\{(\alpha, \beta) \mid \mathcal{W}_S(\alpha, \beta) = \mathcal{L}(S),\ \alpha \in \mathbb{F}_2^n,\ \beta \in \mathbb{F}_2^m \setminus \{0\}\}$, the frequency of the maximum occurs in the LAT of an S-box.	✔

$\deg(f)$	The *algebraic degree (or simply, degree)* of a Boolean function f. Let $\mathrm{ANF}_f = \bigoplus_{u\in\mathbb{F}_2^n} \alpha_u \prod_{i=0}^{n-1} x_i^{u_i}$, where $\alpha_u \in \mathbb{F}_2$. The algebraic degree of f is $\deg(f) \triangleq \max\{\mathrm{wt}(u) \mid u \in \mathbb{F}_2^n \text{ and } \alpha_u \neq 0 \in \mathbb{F}_2 \text{ in } \mathrm{ANF}_f\}$..	✔
$\mathrm{Deg}(S)$	The *algebraic degree (or simply, degree)* of a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$. Let $\mathrm{ANF}_S = \bigoplus_{u\in\mathbb{F}_2^n} \alpha_u \prod_{i=0}^{n-1} x_i^{u_i}$, where $\alpha_u \in \mathbb{F}_2^m$. $\mathrm{Deg}(S) \triangleq \max\{\mathrm{wt}(u) \mid u \in \mathbb{F}_2^n \text{ and } \alpha_u \neq 0 \in \mathbb{F}_2^m \text{ in } \mathrm{ANF}_S\}$.	✔
$\mathrm{Deg}_{\mathrm{Freq}}$	$\mathrm{Deg}_{\mathrm{Freq}} \triangleq \#\{\lambda \mid \deg(S_\lambda) = \mathrm{Deg}(S),\ \lambda \in \mathbb{F}_2^m \setminus \{0\}\}$, the number of non-trivial components with the maximal degree.	✔
$\min\deg(S)$	The *minimum algebraic degree* of a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$, $\min\deg(S) \triangleq \min_{\lambda\in\mathbb{F}_2^m\setminus\{0\}} \deg(S_\lambda)$.	✔
$\mathrm{Deg}_{\mathrm{spec}}(S)$	The *degree spectrum* of a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is a multiset $\mathrm{Deg}_{\mathrm{spec}}(S) \triangleq \{\deg(S_\lambda) \mid \lambda \in \mathbb{F}_2^m \setminus \{0\}\}$, where S_λ are component functions of S.	✔
$\mathrm{Deg}_{\mathrm{spec}_{cor}}(S)$	The *degree spectrum of the coordinates* of a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is a multiset $\mathrm{Deg}_{\mathrm{spec}_{cor}}(S) \triangleq \{\deg(S_{e_i}) \mid 1 \leq i \leq m\}$, where S_{e_i} are coordinate functions of S.	-
$d_k(S)$	$d_k(S) = \max_{K\subseteq\{1,\ldots,m\},\lvert K\rvert\leq k} \deg\left(\prod_{i\in K} S_{e_i}\right)$, maximal degree of the product of k coordinates.	✔
$\mathcal{V}_S$	A table representation of $\mathcal{V}_S(u)$ for all u indicating the appearance of monomials in the ANFs of $x \mapsto \pi_v(S(x))$ for $v \in \mathbb{F}_2^n$, where $\mathcal{V}_S(u) \triangleq \bigcup_{w\in\mathtt{Succ}(u)} V_S(w)$ and $V_S(w) \triangleq \{v \in \mathbb{F}_2^n : \pi_v(S(x))$ contains $\pi_w(x)\}$, $\mathtt{Succ}(u) = \{x \in \mathbb{F}_2^n : u \preceq x\}$ which is an affine subspace of dimension $(n - \mathrm{wt}(u))$, and $\pi_w(x) = \prod_{i=1}^n x_i^{w_i}$.	✔
$\mathrm{LS}(f)$	The *linear space* of a Boolean function f is the linear subspace of those a such that $D_a f$ is a constant function. $\mathrm{LS}(f) \triangleq \{a \in \mathbb{F}_2^n \mid D_a f = c,$ where c is constant 0 or 1$\}$. Such $a,\ a \neq 0$, is said to be a *c-linear structure* of f.	-
(λ, a, c)	A *linear structure* of a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is a triple (λ, a, c) such that a is a c-linear structure of the *component function* $S_\lambda(x)$.	✔
$\#\,\mathrm{LS}(S)$	the number of linear structures of an S-box S.	✔
$\mathrm{d}(f, \mathrm{LS}(n))$	$\mathrm{d}(f, \mathrm{LS}(n)) \triangleq \min_{\ell\in\mathrm{LS}(n)} \mathrm{d}(f, \ell)$. $\mathrm{LS}(n)$ is the subset of Boolean functions having linear structure.	-
(v, w)-linear	For $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$, exist linear subspaces $V \subset \mathbb{F}_2^n$ and $W \subset \mathbb{F}_2^m$ with $\dim V = v$ and $\dim W = w$, such that, for all $\lambda \in W,\ S_\lambda$ has degree at most 1 on all cosets of V.	✔
$\max_v(v, w)$-linear	the maximal v such that the S-box is (v, w)-linear.	✔
$\max_w(v, w)$-linear	the maximal w such that the S-box is (v, w)-linear.	✔

$\mathrm{DDT}_1(S)$	The sub-table of DDT containing entries (a, b) where $\mathrm{wt}(a) = \mathrm{wt}(b) = 1$.	✔
$\mathrm{LAT}_1(S)$	The sub-table of LAT containing entries (u, v) where $\mathrm{wt}(u) = \mathrm{wt}(v) = 1$.	✔
$\mathcal{U}_1(S)$	$\max_{a\in\mathbb{F}_2^n\setminus\{0\}, b\in\mathbb{F}_2^m}\{\lvert\delta_S(a,b)\rvert \mid \mathrm{wt}(a) = \mathrm{wt}(b) = 1\}$.	✔
$\mathcal{L}_1(S)$	$\max_{\alpha\in\mathbb{F}_2^n, \lambda\in\mathbb{F}_2^m\setminus\{0\}}\{\lvert\mathcal{W}_S(\alpha,\beta)\rvert \mid \mathrm{wt}(\alpha) = \mathrm{wt}(\beta) = 1\}$.	✔
$\mathcal{BN}_\mathrm{D}(S)$	The *differential branch number* of an S-box S (a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$), $\mathcal{BN}_\mathrm{D}(S) = \min\{\mathrm{wt}(a) + \mathrm{wt}(b) \mid \delta_S(a,b) \neq 0,\ a \in \mathbb{F}_2^n \setminus \{0\},\ b \in \mathbb{F}_2^m\}$.	-
$\mathcal{BN}_\mathrm{L}(S)$	The *linear branch number* of an S-box S (a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$), $\mathcal{BN}_\mathrm{L}(S) = \min\{\mathrm{wt}(u) + \mathrm{wt}(v) \mid \mathcal{W}_S(u,v) \neq 0,\ u \in \mathbb{F}_2^n,\ v \in \mathbb{F}_2^m \setminus \{0\}\}$.	-
$\mathrm{SetDiff1}(S)$	$\{(a, b) \mid \delta_S(a, b) \neq 0,\ \mathrm{wt}(a) = \mathrm{wt}(b) = 1\}$	✔
$\mathrm{SetLin1}(S)$	$\{(u, v) \mid \lvert\mathcal{W}_S(u, v)\rvert \neq 0,\ \mathrm{wt}(u) = \mathrm{wt}(v) = 1\}$	✔
$\mathrm{CardD1}(S)$	$\#\,\mathrm{SetDiff1}(S)$, number of non-zero entries in DDT_1 .	✔
$\mathrm{CardL1}(S)$	$\#\,\mathrm{SetLin1}(S)$, number of non-zero entries in LAT_1.	✔
$\mathrm{GI}(S)$	$\{a \mid \lvert\delta_S(a, b)\rvert = 0,\ \mathrm{wt}(a) = \mathrm{wt}(b) = 1\}$.	-
$\mathrm{GO}(S)$	$\{b \mid \lvert\delta_S(a, b)\rvert = 0,\ \mathrm{wt}(a) = \mathrm{wt}(b) = 1\}$.	-
$\mathrm{BI}(S)$	$\{a \mid \lvert\exists b, \delta_S(a, b)\rvert \neq 0,\ \mathrm{wt}(a) = \mathrm{wt}(b) = 1\}$.	-
$\mathrm{BO}(S)$	$\{b \mid \lvert\exists a, \delta_S(a, b)\rvert \neq 0,\ \mathrm{wt}(a) = \mathrm{wt}(b) = 1\}$.	-
$\mathrm{Dscore}(S)$	$\lvert\,\mathrm{GI}\,\rvert + \lvert\,\mathrm{GO}\,\rvert$ observed from DDT_1.	-
$\mathrm{Lscore}(S)$	$\lvert\,\mathrm{GI}\,\rvert + \lvert\,\mathrm{GO}\,\rvert$ observed from LAT_1.	-
XE/PE/PXE	XOR-equivalent, Permutation-equivalent, Permutation-XOR-equivalent	✔
LE	Linear-equivalent	✔
AE	Affine-equivalent	✔
EA	Extended-Affine-equivalent	-
CCZ	CCZ-equivalent	-
`BGC`	Bitslice gate complexity	✔
`GEC`	Gate equivalent complexity	✔
`MC`	Multiplicative complexity	✔
`Depth`	Depth complexity	✔

The last column indicates whether PEIGEN evaluates the corresponding criterion.

B Examples for Some Notations

Table 7: Lookup table of the first 4×4-bit S-box S_0 used in Serpent

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	3	8	F	1	A	6	5	B	E	D	4	2	7	0	9	C

$\delta_S(a,b) \triangleq \#\{x \in \mathbb{F}_2^n \mid S(x) \oplus S(x \oplus a) = b\}$

$a\backslash b$	0	1	2	4	8	3	5	6	9	A	C	7	B	D	E	F
0	16	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
1	·	·	·	·	·	4	·	·	4	·	·	4	·	4	·	·
2	·	·	·	·	·	2	4	2	·	2	2	·	·	2	2	·
4	·	·	·	·	·	·	4	2	2	2	2	2	·	·	2	·
8	·	·	·	·	·	2	·	·	2	·	·	2	4	2	·	4
3	·	2	·	2	·	2	·	4	·	2	·	2	2	·	·	·
5	·	2	·	2	·	·	·	·	2	2	4	·	2	2	·	·
6	·	·	2	·	2	·	·	2	·	·	2	·	4	·	·	4
9	·	·	2	4	2	·	·	2	·	·	2	·	·	·	4	·
A	·	·	2	·	2	2	4	·	·	2	·	·	·	2	2	·
C	·	·	2	·	2	·	4	·	2	2	·	2	·	·	2	·
7	·	4	2	·	2	·	·	2	·	·	2	·	·	·	·	4
B	·	2	·	2	4	·	·	·	2	2	·	·	2	2	·	·
D	·	2	4	2	·	2	·	·	·	2	·	2	2	·	·	·
E	·	·	2	·	2	2	·	2	2	·	2	2	·	2	·	·
F	·	4	·	4	·	·	·	·	·	·	·	·	·	·	4	4

DDT_1

$\mathcal{U} = 4$, $\mathcal{D}_{\mathrm{spec}} = \{0 : 159, 2 : 72, 4 : 24, 16 : 1\}$, $\mathcal{U}_1 = 0$, $\mathcal{D}_{\mathrm{spec}_1} = \{0 : 16\}$

Figure 2: The DDT/DDT_1 of the Serpent S-boxes S_0

Table 8: Maximal degree of the product of k coordinates of MISTY1 S-box

$$d_k(S) = \max_{K \subseteq \{1,\ldots,m\}, |K| \leq k} \deg\Big(\prod_{i \in K} S_{e_i}\Big)$$

k	1	2	3	4	5	6	7
d_k	3	5	5	6	6	6	7

$\mathcal{W}_S(a,b) \triangleq \sum_{x\in\mathbb{F}_2^n}(-1)^{b\cdot S(x)\oplus a\cdot x}$

$a\backslash b$	0	1	2	4	8	3	5	6	9	A	C	7	B	D	E	F
0	16	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
1	·	LAT$_1$	·	·	·	·	−8	·	·	·	·	−8	·	−8	·	8
2	·	·	4	−4	4	4	−4	·	−4	·	·	·	8	8	−4	4
4	·	·	−4	−4	−4	4	−4	·	−4	·	·	8	−8	·	−4	4
8	·	·	4	·	−4	−4	·	−4	4	·	−4	4	·	4	8	8
3	·	·	4	4	−4	4	−4	−8	4	−8	·	·	·	·	−4	−4
5	·	·	−4	−4	4	4	4	·	4	−8	8	·	·	·	4	4
6	·	·	·	·	·	−8	·	−8	−8	·	8	·	·	·	·	·
9	·	8	−4	·	−4	−4	·	4	−4	−8	−4	−4	·	4	·	·
A	·	·	8	4	·	·	4	4	·	·	4	−4	−8	4	−4	4
C	·	·	·	−4	8	·	−4	−4	·	·	−4	−4	−8	4	4	−4
7	·	·	·	8	·	8	·	·	−8	·	·	·	·	·	8	·
B	·	−8	·	−4	−8	·	−4	4	·	·	4	−4	·	4	4	−4
D	·	8	8	−4	·	·	−4	4	·	·	4	4	·	−4	4	−4
E	·	·	4	−8	−4	4	8	−4	−4	·	−4	−4	·	−4	·	·
F	·	8	−4	·	−4	4	·	−4	4	8	4	−4	·	4	·	·

$\mathcal{L} = 8$, $\mathcal{W}_{\text{spec}} = \{0:123, 4:96, 8:36, 16:1\}$, $\mathcal{L}_1 = 4$, $\mathcal{W}_{\text{spec}_1} = \{0:8, 4:8\}$

Figure 3: The LAT/LAT$_1$ of the Serpent S-boxes S_0

$\beta_S(a,b) \triangleq \#\{x \in \mathbb{F}_2^n \mid S^{-1}(S(x)\oplus b)\oplus S^{-1}(S(x\oplus a)\oplus b) = a\}$

$a\backslash b$	0	1	2	4	8	3	5	6	9	A	C	7	B	D	E	F
0	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
1	16	·	·	4	·	2	2	2	·	4	2	2	2	·	4	·
2	16	·	BCT$_1$	·	4	4	·	·	2	2	4	4	·	2	2	8
4	16	·	·	16	·	·	·	·	8	8	·	·	·	8	8	·
8	16	·	·	4	·	2	2	2	4	·	2	2	2	4	·	·
3	16	2	2	4	·	2	·	·	4	·	2	2	2	4	·	·
5	16	·	2	4	2	·	2	·	·	6	·	·	2	·	6	·
6	16	2	2	·	4	4	2	2	·	·	4	4	·	·	·	8
9	16	2	2	·	·	·	2	2	2	2	·	·	·	2	2	·
A	16	2	2	4	·	2	·	·	·	4	2	2	2	·	4	·
C	16	2	·	4	2	·	·	2	6	·	·	·	2	6	·	·
7	16	·	2	4	2	·	2	·	6	·	·	·	2	6	·	·
B	16	2	2	·	4	4	2	2	·	·	4	4	·	·	·	8
D	16	·	·	·	8	8	·	·	·	·	8	8	·	·	·	16
E	16	2	·	4	2	·	·	2	·	6	·	·	2	·	6	·
F	16	2	2	·	4	4	2	2	·	·	4	4	·	·	·	8

$\mathcal{BU} = 16$, $\mathcal{BD}_{\text{spec}} = \{0:107, 2:64, 4:32, 6:8, 8:12, 16:33\}$

Figure 4: The BCT of the Serpent S-boxes S_0

$\mathcal{V}_S(u) \triangleq \bigcup_{w \in \mathtt{Succ}(u)} V_S(w)$ and $V_S(w) \triangleq \{v \in \mathbb{F}_2^n : \pi_v(S(x)) \text{ contains } \pi_w(x)\}$,
where $\mathtt{Succ}(u) = \{x \in \mathbb{F}_2^n : u \preceq x\}$ and $\pi_w(x) = \prod_{i=1}^n x_i^{w_i}$

$u\backslash v$	0	1	2	4	8	3	5	6	9	A	C	7	B	D	E	F
0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
2		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
4		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
8		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
3		x	x	x		x	x	x	x	x	x	x	x	x	x	x
5		x	x	x		x		x	x	x	x	x		x	x	x
6		x	x	x		x	x	x	x	x	x	x	x	x	x	x
9		x	x		x	x	x	x	x	x	x	x	x	x	x	x
A		x	x	x		x	x	x	x	x	x	x	x	x		x
C		x	x	x		x	x	x	x	x	x	x	x	x	x	x
7		x	x	x		x				x	x	x		x	x	x
B											x	x	x			x
D		x	x					x	x			x		x	x	x
E		x	x	x		x	x	x		x			x	x		x
F																x

Figure 5: The table representation of $\mathcal{V}_S(u)$ for all $u \in \mathbb{F}_2^n$ of the Serpent S-boxes S_0

Table 9: Degree spectra and Linear structures of some S-boxes

$\mathrm{Deg}_{\mathrm{spec}}(S) = \{\deg(S_\lambda) \mid \lambda \in \mathbb{F}_2^n \setminus \{0\}\} = \{2:3, 3:12\}$, $\#\,\mathrm{LS} = 9$

Noekeon	Piccolo	PRESENT	Rectangle	LBlock_0
(0100, 0001, 1)	(0100, 0001, 0)	(0001, 0001, 1)	(0001, 0100, 1)	(0001, 0001, 1)
(0100, 1010, 1)	(0100, 1000, 1)	(0001, 1000, 1)	(0001, 1000, 1)	(0001, 0010, 1)
(0100, 1011, 0)	(0100, 1001, 1)	(0001, 1001, 0)	(0001, 1100, 0)	(0001, 0011, 0)
(1000, 0001, 1)	(1000, 0001, 1)	(1010, 0001, 1)	(0010, 0001, 1)	(0010, 0011, 1)
(1000, 1000, 0)	(1000, 0010, 0)	(1010, 1110, 1)	(0010, 0100, 1)	(0010, 1000, 1)
(1000, 1001, 1)	(1000, 0011, 1)	(1010, 1111, 0)	(0010, 0101, 0)	(0010, 1011, 0)
(1100, 0001, 0)	(1100, 0001, 1)	(1011, 0001, 0)	(0011, 0100, 0)	(0011, 0011, 1)
(1100, 0010, 1)	(1100, 1010, 1)	(1011, 0110, 1)	(0011, 1001, 1)	(0011, 1001, 0)
(1100, 0011, 1)	(1100, 1011, 0)	(1011, 0111, 1)	(0011, 1101, 1)	(0011, 1010, 1)

$\mathrm{Deg}_{\mathrm{spec}}(S) = \{\deg(S_\lambda) \mid \lambda \in \mathbb{F}_2^n \setminus \{0\}\} = \{2:1, 3:14\}$, $\#\,\mathrm{LS} = 3$

Golden_S0	Golden_S1	Golden_S2	Golden_S3	Qarma_sigma0
(1111, 0100, 0)	(0111, 0010, 0)	(1111, 0100, 0)	(0110, 0010, 1)	(0100, 0100, 0)
(1111, 1010, 1)	(0111, 1100, 1)	(1111, 1001, 1)	(0110, 0101, 1)	(0100, 1011, 1)
(1111, 1110, 1)	(0111, 1110, 1)	(1111, 1101, 1)	(0110, 0111, 0)	(0100, 1111, 1)

$\mathrm{Deg}_{\mathrm{spec}}(S) = \{\deg(S_\lambda) \mid \lambda \in \mathbb{F}_2^n \setminus \{0\}\} = \{3:15\}$, $\#\,\mathrm{LS} = 0$

PRINCE	TWINE	KLEIN	JH_0/1	Qarma_sigma1/2	Panda	Midori_Sb1
Have no linear structure						

ANF of coordinates:

$$
\begin{array}{lllllllllllllllll}
y_0 & = & +x_0 & + & +x_2 & +x_3 & +x_0x_1 & +x_0x_2 & +x_1x_2 & + & + & + & +x_0x_1x_2 & + & +x_0x_2x_3 & +x_1x_2x_3 & ;\deg = 3 \\
y_1 & = & +x_0 & + & + & + & + & +x_0x_2 & +x_1x_2 & + & +x_1x_3 & + & +x_0x_1x_2 & + & +x_0x_2x_3 & +x_1x_2x_3 & ;\deg = 3 \\
y_2 & = 1 & + & +x_1 & + & +x_3 & +x_0x_1 & +x_0x_2 & + & + & +x_1x_3 & + & +x_0x_1x_2 & + & + & +x_1x_2x_3 & ;\deg = 3 \\
y_3 & = 1 & +x_0 & +x_1 & +x_2 & +x_3 & + & + & + & +x_0x_3 & + & + & + & + & + & + & ;\deg = 2
\end{array}
$$

ANF of components:

$$
\begin{array}{lllllllllllllllll}
y_{0001b} & = y_0 & + & + & + & = & +x_0 & + & +x_2 & +x_3 & +x_0x_1 & +x_0x_2 & +x_1x_2 & + & + & + & +x_0x_1x_2 & + & +x_0x_2x_3 & +x_1x_2x_3 & ;\deg = 3 \\
y_{0010b} & = & +y_1 & + & + & = & +x_0 & + & + & + & + & +x_0x_2 & +x_1x_2 & + & +x_1x_3 & + & +x_0x_1x_2 & + & +x_0x_2x_3 & +x_1x_2x_3 & ;\deg = 3 \\
y_{0011b} & = y_0 & +y_1 & + & + & = & + & + & +x_2 & +x_3 & +x_0x_1 & + & + & + & +x_1x_3 & + & + & + & + & + & ;\deg = 2 \\
y_{0100b} & = & + & +y_2 & + & = 1 & + & +x_1 & + & +x_3 & +x_0x_1 & +x_0x_2 & + & + & +x_1x_3 & + & +x_0x_1x_2 & + & + & +x_1x_2x_3 & ;\deg = 3 \\
y_{0101b} & = y_0 & + & +y_2 & + & = 1 & +x_0 & +x_1 & +x_2 & + & + & + & +x_1x_2 & + & +x_1x_3 & + & + & + & +x_0x_2x_3 & + & ;\deg = 3 \\
y_{0110b} & = & +y_1 & +y_2 & + & = 1 & +x_0 & +x_1 & + & +x_3 & +x_0x_1 & + & +x_1x_2 & + & + & + & + & + & +x_0x_2x_3 & + & ;\deg = 3 \\
y_{0111b} & = y_0 & +y_1 & +y_2 & + & = 1 & + & +x_1 & +x_2 & + & + & +x_0x_2 & + & + & + & + & +x_0x_1x_2 & + & + & +x_1x_2x_3 & ;\deg = 3 \\
y_{1000b} & = & + & + & +y_3 & = 1 & +x_0 & +x_1 & +x_2 & +x_3 & + & + & + & +x_0x_3 & + & + & + & + & + & + & ;\deg = 2 \\
y_{1001b} & = y_0 & + & + & +y_3 & = 1 & + & +x_1 & + & + & +x_0x_1 & +x_0x_2 & +x_1x_2 & +x_0x_3 & + & + & +x_0x_1x_2 & + & +x_0x_2x_3 & +x_1x_2x_3 & ;\deg = 3 \\
y_{1010b} & = & +y_1 & + & +y_3 & = 1 & + & +x_1 & +x_2 & +x_3 & + & +x_0x_2 & +x_1x_2 & +x_0x_3 & +x_1x_3 & + & +x_0x_1x_2 & + & +x_0x_2x_3 & +x_1x_2x_3 & ;\deg = 3 \\
y_{1011b} & = y_0 & +y_1 & + & +y_3 & = 1 & +x_0 & +x_1 & + & + & +x_0x_1 & + & + & +x_0x_3 & +x_1x_3 & + & + & + & + & + & ;\deg = 2 \\
y_{1100b} & = & + & +y_2 & +y_3 & = & +x_0 & + & +x_2 & + & +x_0x_1 & +x_0x_2 & + & +x_0x_3 & +x_1x_3 & + & +x_0x_1x_2 & + & + & +x_1x_2x_3 & ;\deg = 3 \\
y_{1101b} & = y_0 & + & +y_2 & +y_3 & = & + & + & + & +x_3 & + & + & +x_1x_2 & +x_0x_3 & +x_1x_3 & + & + & + & +x_0x_2x_3 & + & ;\deg = 3 \\
y_{1110b} & = & +y_1 & +y_2 & +y_3 & = & + & + & +x_2 & + & +x_0x_1 & + & +x_1x_2 & +x_0x_3 & + & + & + & + & +x_0x_2x_3 & + & ;\deg = 3 \\
y_{1111b} & = y_0 & +y_1 & +y_2 & +y_3 & = & +x_0 & + & + & +x_3 & + & +x_0x_2 & + & +x_0x_3 & + & + & +x_0x_1x_2 & + & + & +x_1x_2x_3 & ;\deg = 3
\end{array}
$$

$\mathrm{Deg} = 3, \min \deg = 2, \mathrm{Deg}_{\mathrm{spec}} = \{2:3, 3:12\};$

Figure 6: The ANFs of the Serpent S-boxes S_0

Table 10: (v, w)-linearity of KECCAK 5-bit S-box

$v \setminus w$	1	2	3	4
1	31	31	31	31
2	155	155	155	155
3	155	155	60	5
4	20	5	0	0

(a) The number $N_{(v,w)}$ of subspaces V of dimension v for which there exists a w-dimensional W such that the S-box is (v, w)-linear with respect to (V, W).

Basis of V	W
{0x02,0x04,0x08,0x10}	{0x00, 0x02,0x04,0x06}
{0x01,0x04,0x08,0x10}	{0x00, 0x04,0x08,0x0c}
{0x01,0x02,0x08,0x10}	{0x00, 0x08,0x10,0x18}
{0x01,0x02,0x04,0x10}	{0x00, 0x01,0x10,0x11}
{0x01,0x02,0x04,0x08}	{0x00, 0x01,0x02,0x03}

(b) The 5 pairs of subspaces (V, W) where $|V| = v = 4$ and $|W| = w = 2$ with respect to which the S-box is linear.

Let $g(x) = x_1x_2 \oplus x_0 \oplus x_5$. Let S be the S-box in Scream. Then $g(x) \oplus g(S(x)) = 1, \forall x \in \mathbb{F}_2^8$. Thus, g is a nonlinear invariant for the S-box S in Scream.
Let $g(x) = x_2x_3 \oplus x_0 \oplus x_1 \oplus x_2$. Let S be the S-box in Midori64. Then $g(x) \oplus g(S(x)) = 0, \forall x \in \mathbb{F}_2^4$. Thus, g is a nonlinear invariant for the S-box in Midori64.

Figure 7: Examples of nonlinear invariants. Nonlinear invariant $g(x)$ of the S-box is $g(x) \oplus g(S(x)) = c$, where g is a non-linear Boolean function, and c is a constant.

Table 11: Representatives for all 16 classes of optimal 4 bit S-boxes [LP07]

G_0	0	1	2	D	4	7	F	6	8	B	C	9	3	E	A	5
G_1	0	1	2	D	4	7	F	6	8	B	E	3	5	9	A	C
G_2	0	1	2	D	4	7	F	6	8	B	E	3	A	C	5	9
G_3	0	1	2	D	4	7	F	6	8	C	5	3	A	E	B	9
G_4	0	1	2	D	4	7	F	6	8	C	9	B	A	E	5	3
G_5	0	1	2	D	4	7	F	6	8	C	B	9	A	E	3	5
G_6	0	1	2	D	4	7	F	6	8	C	B	9	A	E	5	3
G_7	0	1	2	D	4	7	F	6	8	C	E	B	A	9	3	5
G_8	0	1	2	D	4	7	F	6	8	E	9	5	A	B	3	C
G_9	0	1	2	D	4	7	F	6	8	E	B	3	5	9	A	C
G_{10}	0	1	2	D	4	7	F	6	8	E	B	5	A	9	3	C
G_{11}	0	1	2	D	4	7	F	6	8	E	B	A	5	9	C	3
G_{12}	0	1	2	D	4	7	F	6	8	E	B	A	9	3	C	5
G_{13}	0	1	2	D	4	7	F	6	8	E	C	9	5	B	A	3
G_{14}	0	1	2	D	4	7	F	6	8	E	C	B	3	9	5	A
G_{15}	0	1	2	D	4	7	F	6	8	E	C	B	9	3	A	5

Table 12: Representatives for all 20 classes of Serpent-type S-boxes [LP07]

R_0	0	3	5	6	7	A	B	C	D	4	E	9	8	1	2	F
R_1	0	3	5	8	6	9	A	7	B	C	E	2	1	F	D	4
R_2	0	3	5	8	6	9	B	2	D	4	E	1	A	F	7	C
R_3	0	3	5	8	6	A	F	4	E	D	9	2	1	7	C	B
R_4	0	3	5	8	6	C	B	7	9	E	A	D	F	2	1	4
R_5	0	3	5	8	6	C	B	7	A	4	9	E	F	1	2	D
R_6	0	3	5	8	6	C	B	7	A	D	9	E	F	1	2	4
R_7	0	3	5	8	6	C	B	7	D	A	E	4	1	F	2	9
R_8	0	3	5	8	6	C	F	1	A	4	9	E	D	B	2	7
R_9	0	3	5	8	6	C	F	2	E	9	B	7	D	A	4	1
R_{10}	0	3	5	8	6	D	F	1	9	C	2	B	A	7	4	E
R_{11}	0	3	5	8	6	D	F	2	7	4	E	B	A	1	9	C
R_{12}	0	3	5	8	6	D	F	2	C	9	A	4	B	E	1	7
R_{13}	0	3	5	8	6	F	A	1	7	9	E	4	B	C	D	2
R_{14}	0	3	5	8	7	4	9	E	F	6	2	B	A	D	C	1
R_{15}	0	3	5	8	7	9	B	E	A	D	F	4	C	2	6	1
R_{16}	0	3	5	8	9	C	E	7	A	D	F	4	6	B	1	2
R_{17}	0	3	5	8	A	D	9	4	F	6	2	1	C	B	7	E
R_{18}	0	3	5	8	B	C	6	F	E	9	2	7	4	A	D	1
R_{19}	0	3	5	A	7	C	B	6	D	4	2	9	E	1	8	F

Table 13: Representatives for all 4 classes of Golden S-boxes [Saa11]

K_0	0	3	5	8	6	9	C	7	D	A	E	4	1	F	B	2
K_1	0	3	5	8	6	C	B	7	9	E	A	D	F	2	1	4
K_2	0	3	5	8	6	A	F	4	E	D	9	2	1	7	C	B
K_3	0	3	5	8	6	C	B	7	A	4	9	E	F	1	2	D

Table 14: Representatives for all 10 classes of Platinum S-boxes [ZBRL15]. In the form P_{ij_k}, the index i represents CardD1 $= i$, the index j represents CardL1 $= j$, and index k represents the index of one PXE-class within one P_{ij} class.

P_{040}	0	B	C	5	6	1	9	A	3	E	F	8	D	4	2	7
P_{041}	0	C	D	A	5	B	E	7	F	6	2	1	3	8	9	4
P_{130}	0	C	9	7	6	1	F	2	3	B	4	E	D	8	A	5
P_{131}	0	C	9	7	F	2	6	1	3	B	4	E	A	5	D	8
P_{132}	0	B	8	5	F	C	3	6	E	4	7	9	2	1	D	A
P_{133}	0	D	4	B	7	E	9	2	6	A	3	5	8	1	F	C
P_{220}	0	D	8	2	E	B	7	5	F	6	3	C	4	1	9	A
P_{221}	0	B	E	1	A	7	D	4	6	C	9	F	5	8	3	2
P_{222}	0	B	6	9	C	5	3	E	D	7	8	4	2	A	F	1
P_{223}	0	E	9	5	F	8	A	7	3	B	6	C	4	1	D	2