AFS: A Family of ARX-Based Large-State S-boxes with Exceptional Properties

Zhiguang Yan[1*], Yongzhuang Wei[1,2*], René Rodríguez-Aldama[3] and Enes Pasalic[3]

[1*]Guangxi Key Laboratory of Cryptography and Information Security, Guilin University of Electronic Technology, Guilin, 541004, China.
[2]State Key Laboratory of Cryptology, Beijing, 100878, China.
[3]University of Primorska, FAMNIT, Koper, 6000, Slovenia.

*Corresponding author(s). E-mail(s): zhiguang_yan@163.com; walker_wyz@guet.edu.cn;

Abstract

Large-state ARX-based S-boxes have become a key component of modern lightweight cryptographic designs, yet deriving tight security bounds for their differential and linear properties remains challenging. In this paper, we study the security of Alzette, the 64-bit ARX-based S-box used in the SPARKLE permutation, and present a general framework for the analysis and design of large-state ARX S-boxes. We introduce SMCS, a hybrid search strategy that combines MILP-based optimization with SMT-based model checking, enabling the computation of tight bounds on maximum expected differential probabilities and linear correlations. Using SMCS, we refine existing bounds for Alzette and, for the first time, establish tight linear bounds (resp. differential bounds) for up to 15 rounds (resp. 14 rounds). Building on these results, we propose S-box configurational encoding, an automated design method for ARX-based S-boxes, and introduce a new family of S-boxes called AFS (ARX-Feistel Structure) with 32-bit and 64-bit instances. We show that selected AFS instances achieve strictly better resistance to single-trail differential and linear cryptanalysis than SPECKEY and Alzette, respectively, while preserving comparable hardware and software costs. Finally, we present the first bit-based SMT model for optimal long-trail decomposition and apply it to derive more accurate bounds for SPARX-128 and SPARKLE. Our results show that replacing the S-boxes with AFS instances yields substantial improvements in cryptanalytic security margins.

Keywords: Automatic search, differential cryptanalysis, linear cryptanalysis, cryptographic S-box, Alzette, configurational encoding

1 Introduction

The widespread promotion of the NIST lightweight authenticated encryption project [32] has rendered lightweight ciphers highly applicable in resource constrained environments, particularly within the Internet of Things (IoT). Notably, numerous large-state lightweight permutations have emerged globally, following designs that fully optimize hardware and software performance while ensuring robust security. The states of these cryptographic permutations are typically 256 bits, 320 bits, or even 512 bits, e.g., SATURNIN [10], ASCON [15], and SPARKLE [2]. Note that ASCON, the winner of the NIST lightweight authenticated encryption competition, employs a 5-bit S-box in its nonlinear layer.

In recent years, S-boxes with states of 16/32/64-bit or larger have been used to construct lightweight protocols. For instance, Piccolo [36], proposed at CHES 2011, is designed as an ultra-lightweight block cipher with a 16-bit S-box obtained by using a Substitution Permutation Network (SPN) structure of four 4-bit parallel S-boxes and a Maximum Distance Separable (MDS) matrix. Dinu et al. [14] introduced SPARX, a 32-bit S-box at Asiacrypt 2016, employing the 'long trail strategy' and they provided provable differential/linear bounds. The block cipher NBC [48], a candidate in the CACR (Chinese Association for Cryptologic Research) 2019 [9], adopts a nonlinear layer constructed by a 16-level nonlinear feedback shift register (NFSR) with four state update functions. This design provides a low hardware implementation cost, albeit it gives a notable delay (time consumption in the hardware implementation). Tian et al. [42] presented a 32-bit S-box using a ring-like cascade connection of NFSRs as the nonlinear layer of SPRING. Due to its large size and high complexity, the maximum differential probability of this S-box could only be computed using a supercomputer (Sunway Taihulight).

At CRYPTO 2020, a novel approach of designing large-state lightweight S-boxes was proposed by Beierle et al. [1]. Namely, a 64-bit lightweight S-box named Alzette, based on the ARX structure, was introduced. The authors claimed that Alzette's differential/linear properties are equivalent to AES [13] after one iteration and comparable to the super S-box of AES after two iterations. Due to its design rationale, Alzette has extremely low software implementation cost, which makes it attractive in certain resource-constrained applications. In general, large-sized S-boxes (also called super S-boxes) are expected to achieve good linear and differential properties more easily, but their precise cryptographic properties are hard (sometimes infeasible) to ascertain in a deterministic way. The rotational operation in Alzette plays an important role from the security viewpoint for it yields a better resistance to differential attacks. Therefore, a suitable selection of rotational sets not only influences the efficiency of implementation, but it also affects the differential/linear properties of the S-box.

For ARX and related primitives, Liu et al. [29] conducted rotational XOR, differential-linear, and rotational differential-linear cryptanalysis, presented at EUROCRYPT 2021, and revealed for Alzette a 4-round differential-linear distinguisher with correlation $2^{-0.27}$ and a "rotational differential-linear" distinguisher with correlation $2^{-11.37}$. Huang et al. [23] proposed a probability formula for rotational-XOR (RX) differentials through modular addition for arbitrary rotation amounts and applied it

to Alzette, but the formula was later shown to be inexact and corrected by Biryukov, Lambin, and Udovenko [5].

Notably, a technique for evaluating the (rotational) differential-linear correlation of ARX-based ciphers was derived at CRYPTO 2022 [33], yielding an 8-round differential-linear distinguisher of Alzette with a theoretical correlation of $-2^{-8.24}$ and an experimental correlation of $-2^{-5.50}$. Recently, a closed-form formula for the bias of rotational differential-linear distinguishers was obtained [28], extending the technique of Morawiecki et al. [30] for Keccak [3]. This research identified a 4-round Alzette rotational differential-linear distinguisher with theoretical correlation of $-2^{-11.37}$ and an experimental correlation of $-2^{-7.35}$.

Further differential-linear results for ARX ciphers include the MIQCP-based search of Lv et al. [27], applied to Alzette and SPECK, and the hourglass-like distinguishers of Gong et al. [19], which also improve results for Alzette.

Since differential and linear cryptanalyses play an important role in security evaluation of ciphers, the challenge in detecting the cryptographic properties of large-scale S-boxes (32-bit or 64-bit) underscores the need for further investigations. This article demonstrates that more accurate security bounds for Alzette can be determined and that there exist better large-state lightweight S-boxes in terms of performance and security compared to Alzette.

Our contributions. This paper makes the following contributions:

1. We propose **SMCS (SMT–MILP Convergence Search)**, a hybrid framework that combines MILP-based optimization with SMT-based solver to efficiently derive tight differential and linear bounds for ARX-based S-boxes, outperforming standalone MILP or SMT approaches for large numbers of rounds.
2. Using SMCS, we refine the security analysis of Alzette by confirming known bounds for small numbers of rounds and establishing, for the first time, tight bounds on the maximum expected linear correlation (resp. differential probability) for up to 15 rounds (resp. 14 rounds).
3. We introduce **S-box configurational encoding**, a general automated design methodology that encodes ARX-Feistel structures and enables systematic exploration and filtering of large-state S-boxes with respect to differential and linear security criteria.
4. We present **AFS (ARX-Feistel Structure)**, a new family of 32-bit and 64-bit S-boxes. Selected AFS instances achieve strictly better resistance to single-trail differential and linear cryptanalysis than SPECKEY and Alzette, respectively, while maintaining comparable hardware and software costs.
5. We propose **the first bit-based SMT model for optimal long-trail decomposition** and apply it to SPARX-128 and SPARKLE, obtaining significantly tighter differential and linear security bounds and demonstrating clear security gains when replacing their original S-boxes with AFS instances.

Organization. In Section 2, we describe the general design of Alzette and the automated model framework for differential/linear cryptanalysis of ARX primitives.

In Section 3, our proposed search strategy based on SMCS for differential/linear cryptanalysis is introduced. An accurate security assessment of Alzette is given. In Section 4, we introduce the concept of S-box configurational encoding and an efficient automated modeling method of designing ARX-like S-boxes. Also, we present some preliminary results related to the use of generalized Feistel structures in this context. In Section 5, we introduce the AFS family of S-boxes and demonstrate that certain members of the AFS-64 family provide much better resistance to differential/linear cryptanalysis (see Table 3). Similarly, we identify the instances within the AFS-32 family that perform significantly better than SPECKEY (see Table 7). Moreover, we provide results of large-scale exhaustive experiments on the 8-bit small-scale version of the AFS S-box family, which show interesting phenomena, and, to a certain extent, provide an explanation for the good security of our AFS S-box family. In Section 6.1, we present an SMT-based framework for optimal long-trail decomposition. Section 6.2 applies this framework to derive tighter differential and linear security bounds for SPARX-128 and SPARKLE (see Tables 10 and D2, and Tables 11 and D3). We further compare these bounds with those obtained after replacing the SPECKEY S-box in SPARX-128 and the Alzette S-box in SPARKLE by our AFS-32 and AFS-64 instances, respectively (see Tables 10 and D2, and Tables 12 and D4). The results demonstrate the advantage of the proposed AFS S-box family. Concluding remarks are given in Section 7.

2 Preliminaries

2.1 Alzette

Alzette is an ARX-based S-box that acts on a 64-bit input and is divided into two 32-bit branches (words), depicted in Fig. 1. We then express the input state as $X = X_1 \| X_0 = (x_{63}, \ldots, x_{32}) \| (x_{31}, \ldots, x_0)$. The Alzette structure is simple, including modular additions ($\boxplus$), XORs ($\oplus$) and rotation ($\ggg$) operations. The modular addition is the only nonlinear operation. More specifically, it consists of the composition of t rounds of the form $T_\ell : \mathbb{F}_2^{32} \times \mathbb{F}_2^{32} \to \mathbb{F}_2^{32} \times \mathbb{F}_2^{32}$ given in general by:

$$T_\ell(X_1, X_0) = (X_1 \boxplus (X_0 \ggg r_\ell), X_0 \oplus ((X_1 \boxplus (X_0 \ggg r_\ell)) \ggg s_\ell)) \oplus (\gamma_\ell^L, \gamma_\ell^R), \quad (1)$$

where $(r_\ell, s_\ell) \in \mathbb{Z}_{32} \times \mathbb{Z}_{32}$, $(\gamma_\ell^L, \gamma_\ell^R) \in \mathbb{F}_2^{32} \times \mathbb{F}_2^{32}$, $\boxplus$ denotes addition modulo 2^{32}, $\oplus$ denotes XOR and $x \ggg r$ denotes the cyclic rotation of a 32-bit word x to the right by the offset r. The authors of Alzette set $t = 4$. The purpose of round constants $(\gamma_\ell^L, \gamma_\ell^R)$ is to initially eliminate the statistical laws in data. For Alzette, the round constants are selected as following: $\gamma_\ell^R = 0$ and $\gamma_\ell^L = c$ for all ℓ. Moreover, the selected rotations are $(r_0, r_1, r_2, r_3, s_0, s_1, s_2, s_3) = (31, 17, 0, 24, 24, 17, 31, 16)$.

The standard instances of Alzette are specified by their round constants; in this paper, as in the original proposal, the eight constants c_i $(0 \le i \le 7)$ are defined as

$$\begin{array}{llll} c_0 = \text{b7e15162}, & c_1 = \text{bf715880}, & c_2 = \text{38b4da56}, & c_3 = \text{324e7738}, \\ c_4 = \text{bb1185eb}, & c_5 = \text{4f7c7b57}, & c_6 = \text{cfbfa1c8}, & c_7 = \text{c2b3293d}. \end{array}$$

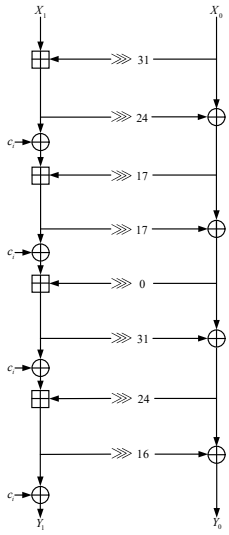


Fig. 1 An instance of Alzette.

2.2 MILP-based framework for differential/linear cryptanalysis of ARX primitives

Mixed-Integer Linear Programming (MILP) refers to a class of optimization problems, and also to the corresponding modeling framework, in which a linear objective function is optimized subject to linear constraints and integrality restrictions on some variables [11]. In our setting, a MILP model consists of an objective function, linear constraints, and continuous or integer variables.

Gurobi [20] is a widely used solver and it is the one that we use throughout this article.

MILP-based differential/linear automated search technology has been employed in the security evaluation of symmetric primitives and it is now well-established (see, for instance [7, 16, 31, 39, 40]). The first use of MILP to count the number of active S-boxes in block ciphers with bit-wise permutations was given in [38]. Then, the authors extended this model to find the actual characteristics [39] and applied their results to some block ciphers involving XOR, an S-box layer and a linear layer with a bit-wise permutation. Soon after, the first MILP-based search for ARX-based primitives was provided in [18], where the authors applied their method to the SPECK family of block ciphers.

Due to space limitations, we omit the full presentation of the standard MILP model for differential/linear cryptanalysis of ARX-based primitives (S-boxes), which has been discussed in several works e.g. in [18, 26]. Nevertheless, we recall two fundamental definitions in the context of differential/linear cryptanalysis.

Definition 2.1 [1] (**MEDCP**) Let E be a key alternating block cipher with r rounds, and round function R. A differential trail is a sequence of differences $\{\delta_0, \delta_1, \ldots, \delta_r\}$ that specifies

not only the input and output differences to the block cipher, but also the intermediate differences between the rounds such that $R(\delta_i \oplus x) \oplus R(x) = \delta_{i+1}$. The approximated probability (averaged over all round keys) is derived as the product of the probabilities of the transitions occurring in each round. The maximum probability (across all trails) computed in this way is called the Maximum Expected Differential Characteristic Propagation (MEDCP).

In the context of linear cryptanalysis a similar definition is useful, see also [1].

Definition 2.2 (**MELCC**) Let E be a key alternating block cipher with r rounds, and round function R. Let α and β denote the input and output mask so that $\beta E_k(x) = \alpha \cdot x + f(k)$, where "$\cdot$" denotes the usual scalar product over $\mathbb{F}_2^n$ and f is a function of the key bits. In practice, we look for a sequence of input, output and intermediate masks $\alpha_0, \ldots, \alpha_r$ called a linear trail/characteristic that has high absolute correlation, where $\alpha_{i+1} \cdot R(x) = \alpha \cdot x \oplus f_i(k)$. The maximum correlation (across all trails) computed in this way is defined as the Maximum Expected Linear Characteristic Correlation (MELCC).

2.3 Long Trail Strategy

The long trail strategy (LTS), introduced in the design of SPARX [14], is a method for deriving bounds against single-trail differential and linear cryptanalysis for primitives built from large S-boxes and sparse linear layers. Instead of only counting active S-boxes, LTS tracks chains of consecutive S-box evaluations that remain connected through the surrounding linear layer. Such a chain is called a long trail.

Let S be the underlying large S-box, and assume that bounds for i consecutive iterations of S are available, denoted by $\mathrm{MEDCP}(S^i)$ in the differential case and $\mathrm{MELCC}(S^i)$ in the linear case. If a trail decomposition contains t_i long trails of length i, then the corresponding differential bound is

$$\Pr(\mathcal{T}) \leq \prod_{i \geq 1} \mathrm{MEDCP}(S^i)^{t_i},$$

and the analogous linear bound replaces $\mathrm{MEDCP}(S^i)$ by $\mathrm{MELCC}(S^i)$. Equivalently, after taking negative base-2 logarithms, the problem becomes the computation of a total long-trail weight.

Therefore, tight long-trail bounds depend on two ingredients: accurate bounds for iterated applications of the S-box, and an accurate decomposition of the primitive into long trails. In Section 6.1, we refine the latter step by introducing a bit-based SMT model for optimal long-trail decomposition.

2.4 SMT-based framework for differential/linear cryptanalysis of ARX primitives

Satisfiability Modulo Theories (SMT) is a satisfiability problem that involves a range of theories and aims to determine whether a set of variable assignments can satisfy a Boolean formula containing theoretical constraints. Due to space limitations, we

present a framework (model) for differential and linear cryptanalysis of ARX primitives based on SMT in Appendix A.1. It basically covers a modeling of the XOR operation, rotations and modular addition operation. We use the CVC[37] language to describe SMT models, which is a high-level declarative language designed for encoding and solving complex SMT problems. As before, due to the duality between linear and differential approaches, the SMT models for the branching, XOR, rotation and the extra constraint for linear approximations are analogous to the differential case.

3 Novel search for differential/linear characteristics

Alzette was designed using the long-trail design strategy (LTS), recalled in Section 2.3, which forces to test the differential and linear bounds of an S-box instance over many iterations, i.e., the computation of its MEDCP and MELCC. However, the search for optimal differential characteristics/linear approximations for a large number of rounds of an S-box instance is extremely time consuming. The designers of Alzette provided only a rough range for the differential bounds over 7 rounds. Moreover, they specified tight upper bounds on the linear characteristics only for up to 8 rounds. To address this problem, we propose a new optimal differential characteristics/linear approximations search strategy. We will describe our search strategy in the context of differential cryptanalysis. However, our methods apply similarly to linear cryptanalysis without any significant difference.

Assume that the probability of an optimal differential characteristic is 2^{-p}. In this setting, the purpose of the MILP search is to find an optimal value of p, say, p_{up}, that satisfies the target constraints, i.e., MILP finds a feasible solution that minimizes p. On the other hand, the purpose of the SMT search is to find a feasible characteristic that attains the optimal value(s) within a range specified recursively from previous round differential probabilities. Thus, the search strategy SMCS, as outlined in Algorithm 1, is based on the powerful properties of these two automated search methods. Initially, we employ the MILP method to search for an optimal solution, that is, the minimum possible value of p. Since it is computationally challenging to obtain the minimum value of p for a large number of rounds within a feasible time, MILP can provide an upper bound p_{up} for p quite efficiently. Then, a range of possible optimal probability values is constructed using the optimal solution value p_{low}, obtained for the previous round as the lower bound. Thus the (integer) range of target values is $[p_{low}, \ldots, p_{up}]$. We then apply the SMT method to conduct parallel searches of feasible solutions for each value in $[p_{low}, \ldots, p_{up}]$. The value of the optimal solution is then the minimum value corresponding to a feasible solution among all the parallel SMT models. This observation allows us to reduce the search space (thus speeding up the entire process) since once an SMT model finds a feasible solution for a value of p, all the SMT models, whose targets are greater than p, will immediately terminate their search. In the worst-case scenario, the search time is determined by the longest time taken to find a feasible solution for each value in $[p_{low}, \ldots, p_{up}]$. Note however that the problem of finding optimal characteristics through MILP and/or SMT is inherently an NP problem.

Algorithm 1 A novel optimal SMCS-based search for linear/differential characteristics

Input: M_{lp}, M_{cvc}, p_{low}
▷ M_{lp}: MILP-based search model; M_{cvc}: SMT-based search model; p_{low}: optimal obj. value for $(r-1)$-round search
Output: p_{om}, C_a
▷ p_{om}: optimal objective value for the r-round search; C_a: corresponding optimal characteristic
Solve M_{lp} (with time limit) to obtain a feasible solution with objective p_{up} ▷ provides a valid upper bound for the r-round optimum
Instantiate SMT instances $\{M_{cvc}(p) \mid p = p_{low}, p_{low}+1, \ldots, p_{up}\}$ ▷ each encodes the r-round model with objective fixed to p
Run all $M_{cvc}(p)$ in parallel ▷ search for the minimal feasible p
if any $M_{cvc}(p_{fe})$ is satisfiable **then**
    Terminate all instances with $p > p_{fe}$ ▷ prune non-optimal candidates $(p > p_{fe})$ in minimization.
    $p_{om} \leftarrow p_{fe}$
    $C_a \leftarrow$ characteristic from the satisfying assignment of $M_{cvc}(p_{om})$
end if
    return p_{om}, C_a

Remark 1 The reason for combining two different strategies is as follows. In general, each tool has its own advantages in solving specific problems. For instance, MILP performs better when solving optimization problems, whereas SMT is more efficient in dealing with (un)satisfiability problems. Therefore, we employ a hybrid search strategy by taking the result of the $(r-1)-$th round as a lower boundary. Combined with the relaxed upper bound obtained by the MILP model, we can model SMT to search for parallel solutions (within a certain range of probabilities) so that the execution time is significantly reduced when the number of rounds is large. For instance, it takes more than a month to search for 7-round optimal differential characteristics of Alzette when using only MILP, and it takes 8205 seconds when doing so with SMT, but it takes only 3714 seconds via our hybrid search strategy. Moreover, our model implements a global search, whereas other heuristic methods attempting to increase the search efficiency may miss certain differential characteristics with higher probability. In [23], the Alzette's 9-round and 10-round MEDCP are estimated as 2^{-40} and 2^{-46}, whereas we could find a higher probability MEDCP of 2^{-39} and 2^{-45} as shown in Tables 1 and 2. Notably, our approach exhibits tight linear bounds (resp. differential bounds) for up to 15 rounds (resp. 14 rounds).

The source code, experimental environment and the specific differential (resp. linear) trail results can be found at https://osf.io/fbzn4/. Our analysis of differential characteristics and linear approximations of Alzette, using SMCS, is displayed in Tables 1 and 2.

Table 1 Comparison of the known differential/linear bounds of Alzette. The fourth and seventh rows show the results obtained using the SMCS search strategy. Here, ⋆ marks an improvement over [23], and † a first-reported bound.

Rounds	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Reference
MEDCP	2^{0}	2^{-1}	2^{-2}	2^{-6}	2^{-10}	2^{-18}	$\leq 2^{-24}$	$\leq 2^{-32}$	$\leq 2^{-36}$	$\leq 2^{-42}$	$\leq 2^{-46}$	$\leq 2^{-52}$	-	-	-	[1]
	2^{0}	2^{-1}	2^{-2}	2^{-6}	2^{-10}	2^{-18}	2^{-26}	2^{-34}	2^{-40}	2^{-46}	2^{-51}	$\leq 2^{-55}$	-	-	-	[23]
	2^{0}	2^{-1}	2^{-2}	2^{-6}	2^{-10}	2^{-18}	2^{-26}	2^{-34}	$2^{-39}\star$	$2^{-45}\star$	2^{-51}	$2^{-59}\dagger$	$2^{-66}\dagger$	$2^{-72}\dagger$	$\leq 2^{-77}\dagger$	This paper
MELCC	2^{0}	2^{0}	2^{-1}	2^{-2}	2^{-5}	2^{-8}	2^{-13}	2^{-17}	-	-	-	-	-	-	-	[1]
	2^{0}	2^{0}	2^{-1}	2^{-2}	2^{-5}	2^{-8}	2^{-13}	2^{-17}	2^{-19}	2^{-22}	-	-	-	-	-	[23]
	2^{0}	2^{0}	2^{-1}	2^{-2}	2^{-5}	2^{-8}	2^{-13}	2^{-17}	2^{-19}	2^{-22}	$2^{-26}\dagger$	$2^{-30}\dagger$	$2^{-34}\dagger$	$2^{-37}\dagger$	$2^{-41}\dagger$	This paper

Table 2 Optimal differential/linear characteristics of Alzette obtained via SMCS.

Characteristics	Rounds	Input	Output	Bounds
Differential characteristics	7	0080402000402000	F0107020903010C1	2^{-26}
	8	8002010000010080	C08001C0018000C3	2^{-34}
	9	210002410080C121	81E00020A0600081	2^{-39}
	10	8542000242010081	E00040A02010B000	2^{-45}
	11	8040010040000080	8010002800200079	2^{-51}
	12	8002010000010080	2090114150C11091	2^{-59}
	13	0840800180204008	0400020402020406	2^{-66}
	14	0A84000584020102	2050110028811008	2^{-72}
Linear characteristics	9	0400020181020106	8180808080808181	2^{-19}
	10	0780030181820185	0206030681838103	2^{-22}
	11	0300030181820103	0870381410603028	2^{-26}
	12	80000201C0030100	0112D89CD80C0112	2^{-30}
	13	6020204020206850	8141414041C04180	2^{-34}
	14	2C00080004040438	0C10081C040C060A	2^{-37}
	15	0400020181020106	0130000E0220001C	2^{-41}

4 Configurational encodings of ARX-like S-boxes

In this section, we propose an efficient search method that enables a flexible modeling for designing large ARX-based S-boxes. As the initial step, suitable components (modules) for constructing the S-box are selected and encoded. Then, using such an encoding, all possible configurations are generated according to the prescribed requirements. Since the encoded sequence contains the structural information about the S-box, it can also be referred to as the 'S-box configurational sequence'. To achieve a high level of automatization of the modeling process, the basic structural configurations are implemented as subroutines and invoked in the program whenever they are a part of the S-box configurational sequence. Subsequently, the process of filtering is performed by imposing certain criteria that S-boxes are required to fulfill.

4.1 Decomposing the Alzette structure

We now describe the process of MILP automated modeling applied to Alzette (a similar approach can be obtained using SMT). Note that XORing constants does not affect differential or linear propagation. Thus, for simplicity, we omit the round constants when modeling differential/linear characteristics. Observe, however, that we consider

them when evaluating implementation costs, concrete cipher instantiations, or other attack vectors (e.g. algebraic attacks).

The single-round structure of Alzette, without the addition of round constants, can be decomposed into two types of Feistel structures, as shown in Fig. 2, where $k_a, k_b \in \mathbb{Z}_{32}$ represent the rotational constants (the number of rotational shifts). By encoding the structure (a) as 1 and the structure (b) as 0, excluding the addition of round constants, Alzette consisting of 4 and 8 rounds can be represented as the following configurational encodings 10101010 and 1010101010101010, respectively. An example of using the structural sequence for MILP-based automated modeling with respect to differential cryptanalysis of Alzette is presented in Algorithm 2 in Appendix G (linear cryptanalysis can be modeled similarly). Furthermore, modeling other S-boxes that employ the structures (a) and (b) is quite flexible; it is only necessary to modify the structural sequence A_{2N}, the number of structural rounds N, and the sequence of rotational shifts K_{2N} in Algorithm 2. For convenience of the reader, our code is available at https://osf.io/fbzn4/.

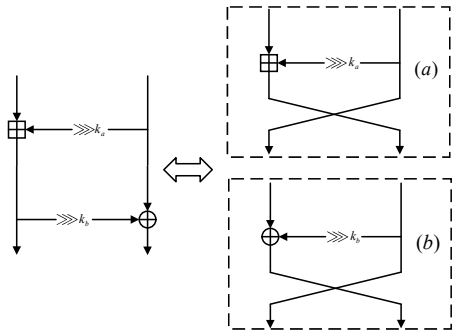


Fig. 2 Structural decomposition of 1-round Alzette without the round constant addition.

4.2 ARX S-boxes via generalized Feistel structures

A generalized Feistel structure easily admits similar decompositions and configurational encodings as discussed above. Again, for simplicity, we ignore the addition of the round constants. Fig. 3-6 show certain modules that can be combined through configurational encoding for specifying ARX-based S-boxes. In particular, the structure given in Fig. 7 is inspired by the design of the block cipher Ballet[12] (winner of the first prize in the block cipher algorithm category of the "China cryptographic algorithm design competition").

Using these generalized Feistel modules mentioned above, one can design robust S-boxes with improved resistance against differential and linear attacks, as shown in Table E5 in Appendix E. Due to a large time complexity of the evaluation process, a thorough security analysis of these S-boxes will be given in the extended version of this manuscript.

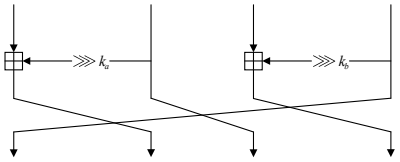


Fig. 3 Structure encoded as 2.

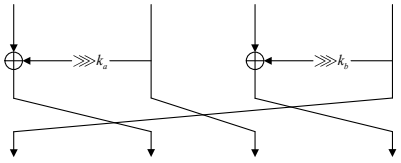


Fig. 4 Structure encoded as 3.

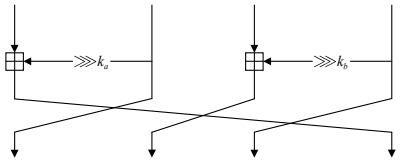


Fig. 5 Structure encoded as 4.

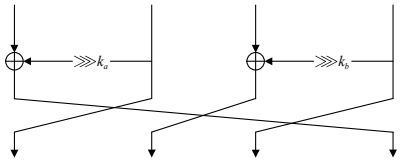


Fig. 6 Structure encoded as 5.

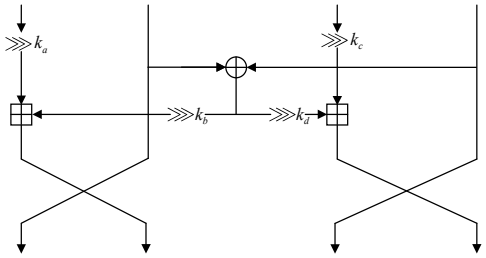


Fig. 7 Structure encoded as 6.

5 Design and analysis of the AFS family of S-Boxes

In this section, we utilize the Feistel structures (a) and (b) in Fig. 2 as elementary building blocks. The proposed search strategy may essentially be viewed as a general framework for finding (sub)optimal large cryptographic S-boxes that employ the ARX-Feistel structure. The family of S-boxes specified using our method will be named AFS, and their varying block length versions are denoted as AFS-$2n$, where n represents the branch (word) size. The main idea is to decompose the standard Feistel scheme, as in Fig 2, and to find an optimal configuration of these core components that potentially perform better than the S-boxes of Alzette. We will demonstrate that our S-boxes provide better resistance to differential/linear cryptanalysis than Alzette when 64-bit S-boxes are considered, cf. Table 3.

A similar conclusion is deduced for our 32-bit versions, where in this case the overall performance is compared to SPECKEY, cf. Table 7.

To specify large ARX-like S-boxes, we need to specify a configurational encoding (the overall structure of S-box), rotational shifts and round constants. In general, the following search strategy was adopted:

1. Selection of a suitable configuration - we preliminary fix the rotation set (for instance the same one as used in Alzette) and estimate the differential and linear probabilities for different configurations of certain Feistel-like modules and choose the one that minimizes these values.

2. Testing other rotational shifts - without exceeding the implementation cost bound, other rotations are tested in terms of the differential/linear properties.

5.1 Specification of AFS-64

The main design objective is to ensure that the software and hardware performance of AFS-64 is not inferior to Alzette. Of course, the additional goal is to potentially achieve higher resistance to differential and linear cryptanalysis than that of Alzette.

Defining a sequence of two consecutive calls to any structure as one round (representing the structures (a) and (b) in Fig. 2 as 1 and 0 respectively), there are four possible encodings for a single round: 11, 00, 01, and 10. The structures corresponding to these encodings are shown in Fig. 8-11. The addition of the round constant c aims at breaking the chain of modular additions and at eliminating the statistical properties of the data. Repeating the structure represented by the '10' encoding four times is equivalent to implementing Alzette (thus the iteration number is $N = 4$).

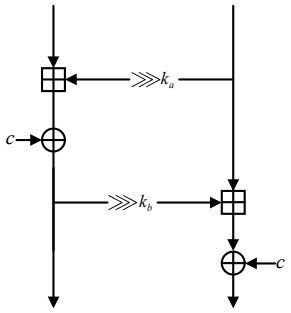


Fig. 8 Structure encoded as 11.

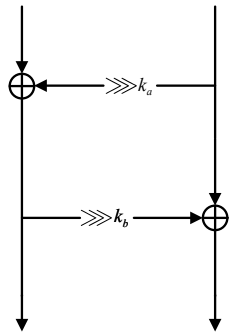


Fig. 9 Structure encoded as 00.

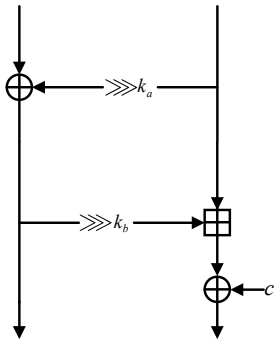


Fig. 10 Structure encoded as 01.

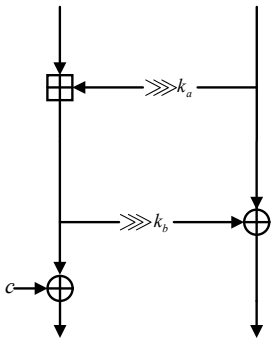


Fig. 11 Structure encoded as 10.

Notice that identifying an optimal structure for AFS-64 essentially corresponds to evaluating different configurational encodings for $N = 4$ in terms of their differential/linear properties.

It is worth pointing out that the ARX S-box considered by Fan et al. [17] fits into the framework considered here. In the notation of this paper, it corresponds to one concrete instance with structural sequence 11000011. This observation, however, is

different from having a general design framework. Our approach starts from a decomposition of Alzette-type ARX-Feistel rounds into elementary modules and uses the resulting configurational sequences as an explicit search space. This makes it possible to enumerate admissible structures, combine them with rotation choices, and evaluate the resulting candidates under the same differential and linear criteria. Furthermore, the LTS-based analysis requires bounds for several consecutive iterations. The evaluation in [17] is limited to the 4-round instance, reporting MEDCP 2^{-6} and MELCC 2^{-3}, and does not provide 8-round differential or linear bounds.

From a software performance viewpoint, we initially employ the rotational sequence $K_8 = [31, 24, 17, 17, 0, 31, 24, 16]$ which is also used in Alzette. Alternatively, other rotational shifts could have been used, but the sequence K_8 is selected due to its efficiency supported by empirical data [1]. To identify an optimal configuration, we assign the values of MEDCP and MELCC (for $N = 4$ iterations) to be at most 2^{-6} and $\leq 2^{-2}$, respectively. These values exactly correspond to the differential and linear bound of Alzette. The set of configurational encodings to be filtered is denoted by $A_8 = \{S \in \{0,1\}^8 | wt_H(S) \leq 4\}$, where wt_H represents the Hamming weight, and its cardinality is 163. The condition $wt_H(S) \leq 4$ is utilized to ensure that the number of invocations of structure (a) is at most 4, corresponding to encoding that involves the modular addition, thereby guaranteeing that the software and hardware performance is at least equivalent to that of Alzette. However, the experimental results indicate that using the rotational sequence $K_8 = [31, 24, 17, 17, 0, 31, 24, 16]$, the Hamming weight of any configurational encoding that meets the differential/linear criteria is at least 3, see also Table 3. In this context, a sequence and its reverse are considered as the same type of a structural sequence. For instance, 10100011 and 11000101 represent the same structural sequence.

Table 3 Differential and linear bounds for selected configurational sequences A_8, with $wt_H(A_8) \leq 4$, and rotational shifts $K_8 = [31, 24, 17, 17, 0, 31, 24, 16]$.

Cryptographic S-box	A_8	MEDCP	MELCC	$wt_H(A_8)$
Alzette	10101010	2^{-6}	2^{-2}	4
AFS-64	11000011	2^{-9}	2^{-4}	4
	10100011	2^{-9}	2^{-3}	4
	11100001	2^{-9}	2^{-3}	4
	10000011	2^{-8}	2^{-3}	3
	10001011	2^{-8}	2^{-3}	4
	10010011	2^{-8}	2^{-3}	4

There are 22 structural sequences that meet the given criteria (excluding '10101010' which corresponds to Alzette), though only the six best are listed in Table 3. For self-completeness, the structure of AFS-64 that employs the structural sequence '11000011' is given in Appendix C. To perform a comparison with Alzette, we refer to the cost metric for different rotation shifts used by Beierle et al. in [1], as shown in Table 4.

The rotational sequence [31,24,17,17,0,31,24,16] used by Alzette corresponds to a cost sequence of [2.66, 2.00, 2.66, 2.66, 0.00, 2.66, 2.00, 0.00], with a total cost of

Table 4 Average number of clock cycles to implement rotations for three platforms when the operand is 32 bits.

Rotational shifts	8-bit AVR	16-bit MSP	32-bit ARM	Cost
(0, 16)	0	0	0	0.00
(8, 24)	0	6	0	2.00
(1, 15, 17, 31)	5	3	0	2.66

14.66. Therefore, in the second stage of filtering, also aimed at ensuring software performance at least equivalent to Alzette's, only those rotational shifts whose total cost is at most 14.66 are considered. For these rotational sequences, we can estimate the differential and linear probabilities of a 4-round (also for 8 rounds) S-box whose encoding is 11000011.

In addition, when the number of iterations is $N = 8$ then (relative to Alzette) we consider MELCC $\leq 2^{-17}$ and MEDCP $\leq 2^{-32}$.

Partial results of this filtering process are presented in Table 5. It should be noted that our aim here is merely to demonstrate the effectiveness of our method, and therefore only a single rotational sequence in Table 3 is considered.

Table 5 Comparison of AFS-64 with Alzette, using the structure 11000011 and different K_8 with a total cost of 14.66. The first row specifies MEDCP and the second row MELCC.

Cryptographic S-box	A_8	K_8	$N = 4$	$N = 8$	Reference
Alzette	10101010	[31,24,17,17,0,31,24,16]	2^{-6} 2^{-2}	$\leq 2^{-32}$ 2^{-17}	[1]
			2^{-6} 2^{-2}	2^{-34} 2^{-17}	This paper
AFS-64	11000011	[1,1,16,24,31,31,8,16]	2^{-11} 2^{-6}	2^{-36} 2^{-18}	This paper
		[1,1,16,8,31,31,24,16]	2^{-11} 2^{-6}	2^{-36} 2^{-18}	
		[17,24,1,1,16,31,24,0]	2^{-10} 2^{-6}	2^{-37} 2^{-20}	
		[17,8,1,1,16,31,8,0]	2^{-10} 2^{-6}	2^{-37} 2^{-20}	

5.2 Specification of AFS-32

The challenge of providing provable security against single-trail differential and linear cryptanalysis in the design phase of ARX ciphers was addressed in [14], where the lightweight block cipher Sparx was introduced. Sparx employs a 32-bit ARX-based S-box named SPECKEY.

In this case, one can also design better 32-bit S-boxes. To illustrate this, we introduce the AFS-32 family,

whose software/hardware performance and resistance to differential/linear cryptanalysis are not inferior to SPECKEY. The round structure of SPECKEY is shown in Fig. 12. Moreover, four iterations of SPECKEY are used in Sparx as its S-box, thus $N = 4$. Clearly, the branch word size of AFS-32 is 16 bits.

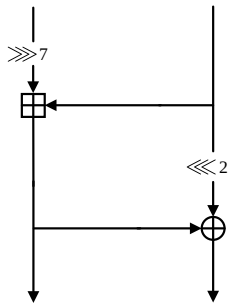


Fig. 12 SPECKEY round structure.

Similarly to the AFS-64 family, we employ the structures (a) and (b) in the context of structural encoding. The possible structures for one iteration/round are the same as in the previous section, where the only difference being the size of the branch word. For efficiency reasons, we only consider two types of structural encodings, namely 11000011 and 10100011, whereas the rotational shifts are selected using imposed bounds on differential and linear probabilities.

The rotational shifts are selected by measuring the average number of clock cycles to implement different rotational shifts, see Table 6.

Table 6 Average cost of implementing a rotational shift for 16-bit vectors.

Rotational shift	8-bit AVR	16-bit MSP	32-bit ARM	Cost
0	0	0	0	0.00
8	3	1	1	1.66
15	3	2	2	2.33
1	4	2	2	2.66
7	6	3	2	3.66
9	7	3	2	4.00
14	6	4	2	4.00
2	8	4	2	4.66
6	9	5	2	5.33
4	7	8	2	5.66
12	7	8	2	5.66
13	9	6	2	5.66
3	10	6	2	6.00
10	11	5	2	6.00
11	10	7	2	6.33
5	11	7	2	6.66

For the considered structure 10100011, the rotational sequence was selected from $8! = 40,320$ permutations of $(0, 1, 2, 7, 8, 9, 14, 15)$. However, for the structure 11000011, we considered the full rotation space so that permutations utilize 8 distinct values chosen from the set $\{0, 1, \ldots, 15\}$. To provide a fair comparison to Sparx, our search was further constrained to sequences whose total implementation cost C satisfied $20.00 \leq C < 24.00$.

For instance, considering four iterations of SPECKEY in Sparx with the rotational sequence $[7, 14, 7, 14, 7, 14, 7, 14]$, the total implementation cost is $(3.66 + 4.00) \times 4 = 30.64$. In contrast, the total implementation cost of an instance of AFS-32, using the rotational sequence $[0, 1, 2, 7, 8, 9, 14, 15]$, is $0.00 + 2.66 + 4.66 + 3.66 + 1.66 + 4.00 + 4.00 + 2.33 = 22.97$, which represents a significant improvement in terms of implementation cost compared to SPECKEY. Additionally, the selected configurational encoding ensures that the total software and hardware cost of implementing the relevant operations (modular addition, XOR, and round constant addition) in AFS-32 is at least as good as the implementation of 4-rounds SPECKEY. Table 7 provides a comparison of MEDCP and MELCC for different choices of rotational constants, where the selected parameters ensure that AFS-32 instances achieve a better software and roughly equal hardware performance. For convenience, the structure of AFS-32 using the encoding 10100011 and the encoding 11000011 are given in Appendix C.

Table 7 Partial comparison between AFS-32 and SPECKEY having the rotational sequences K_8 with a total cost in [20,24). The first row specifies MEDCP and the second row stands for MELCC.

Cryptographic S-box	A_8	K_8	4-Round	8-Round	Reference
SPECKEY	-	[7,14,7,14,7,14,7,14]	2^{-5} 2^{-3}	2^{-24} 2^{-12}	[14]
AFS-32	11000011	[1,8,14,0,12,7,15,9]	2^{-10} 2^{-5}	2^{-31} 2^{-15}	This paper
		[7,14,8,1,13,0,9,15]	2^{-10} 2^{-5}	2^{-31} 2^{-15}	
	10100011	[0,7,14,15,8,1,2,9]	2^{-9} 2^{-5}	2^{-30} 2^{-14}	
		[8,0,2,1,9,14,7,15]	2^{-9} 2^{-5}	2^{-29} 2^{-14}	

The implementation-cost claims above are supported by the concrete software and hardware benchmarks reported in Appendix F. In brief, the tested AFS-64 instances with the same number of modular additions as Alzette have essentially the same software cost as Alzette on AVR ATmega128 and ARM Cortex-M3 when comparable rotation sequences are used. For the 32-bit case, the selected AFS-32 instances reduce the software cost compared with SPECKEY, e.g., from 87/151 to 59/81 AVR cycles (core-only/core+load-store) and from 40 to 36 ARM cycles, while also reducing code size. The ASIC synthesis results show that the selected AFS instances remain competitive in area and delay.

5.3 Evaluation of cryptographic properties

In this section, we provide a rather extensive security analysis (based on computer-based simulations) in the fixed-key scenario. In virtually all block cipher designs, the security arguments with respect to differential/linear cryptanalysis are commonly presented using the Markov chain hypothesis, thus the results are interpreted as being averaged over the key space, see also [1]. However, it is theoretically possible (violating the Markov hypothesis) that the cryptographic properties diverge from the expected (averaged) values for certain fixed keys. Due to the employment of MILP and SMT tools (with respect to the round constants) this averaging process is immanently embedded in our algorithms. Furthermore, in our case the secret key essentially corresponds to the round constants and therefore the expected differential/linear probabilities are deduced by randomly sampling the round constants and using a randomly selected pool of plaintexts or plaintext-ciphertext pairs.

Differential probabilities in the fixed-key scenario. In accordance to the above comments, we conducted experiments concerning all the AFS instances given in Table 5 and 7 to verify whether the expected maximum probabilities of differential trails (determined by our automated search tools) agree with the actual differential probabilities of the fixed ARX-boxes.

Using our automated tools, we have identified differential trails for four different rotational sequence instances of AFS-64 (respectively AFS-32) corresponding to the maximum expected differential trail probability. Specifically, following the order of instances listed in our table from top to bottom, we found 17, 17, 3, and 3 (respectively 2, 4, 6, and 4) distinct trails. These values represent the complete set of differential characteristics achieving the 4-round MEDCP. We have experimentally verified the actual probability for all of these highest-probability differential trails. The precise specifications of these trails are provided in Appendix B. On the other hand, denoting by c_i the round constants, for any AFS instance AFS_{c_i} and any such differential trail (where the input-output difference is the tuple (α, β)), we experimentally computed the actual probability of the differential $\alpha \to \beta$ by using:

$$\frac{|\{x \in S | AFS_{c_i}(x) \oplus AFS_{c_i}(x \oplus \alpha) = \beta\}|}{|S|},$$

where S is a set of 2^{28} randomly sampled inputs. The experimental results show that the expected probabilities of the differential trails, given in Appendix B, approximate the actual differential probabilities of AFS quite accurately. More precisely, the experimentally computed probabilities deviate from the theoretical values by at most $\pm 10^{-5}$, for the selected cardinality $\#S = 2^{28}$. Most notably, the experimental results by using randomly selected round constants confirm that all AFS instances are insensitive to the choice of round constants, a similar observation was also reported in [1].

Linear correlations in the fixed-key scenario. As for the case of differential trails, for the AFS instances given in Tables 5 and 7, similar simulations have been performed for the purpose of comparing the expected linear correlations (derived

by our automated search tools) with the actual values obtained using a randomly sampled subset of plaintexts, denoted by S.

For different AFS-64 (resp. AFS-32) instances, corresponding to the rows in Table 5 (resp. Table 7), the following was observed. Using our automated search tools, we have identified 496, 496, 960, and 960 (resp. 56, 88, 352, and 464) linear trails that correspond to the maximum expected correlations $2^{-6}, 2^{-6}, 2^{-6}$ and 2^{-6} (resp. $2^{-5}, 2^{-5}, 2^{-5}$ and 2^{-5}). On the other hand, our experimental results indicate again a slight clustering phenomena. The worst-case linear correlations deviate slightly from the expected values and are given as $2^{-5.01}, 2^{-5.12}, 2^{-5.22}$, and $2^{-4.57}$ (resp. $2^{-4.55}, 2^{-4.58}, 2^{-3.55}$, and $2^{-3.58}$). Our experimental results were derived using a randomly sampled subset of plaintexts S of cardinality 2^{24} (resp. 2^{28}, 2^{28}, 2^{24}, 2^{24}).

Algebraic properties of AFS. Integral cryptanalysis [4] exploits low algebraic degree of a more fine-grained algebraic degeneracy of the cryptographic primitive. An integral distinguisher defines an input set X such that the encryption function sums to zero over this set (or a subset of bits), for any value of the secret key involved. In the case of keyless permutations, such as an ARX-box, such distinguishers are trivial to find but they do not pose a real security threat per se. However, an analysis of the growth of the algebraic degree [22, 24] (and the evolution of the algebraic structure in general) provides a useful information about the permutation. When such a permutation is employed in a block cipher, this information then becomes helpful in specifying integral distinguishers of the cipher.

Division property is a technique introduced by Todo [43] for finding integral characteristics [6]. Originally, it was applied to substitution-permutation networks and block ciphers using Feistel-like structure. The so-called bit-based division property was proposed by Todo and Morii [44] and applied to the block cipher Simon that uses 32-bit blocks. Due to the high computation complexity of the search algorithm, it is infeasible to apply this technique to ciphers with larger block sizes. However, Xiang et al. [47] discovered that the bit-based division property propagation can be efficiently modeled as an MILP instance, which, surprisingly, can be solved in practice using modern optimization software (Gurobi Optimizer [20]) for practically all known block ciphers [21, 46].

Modular addition is often the most complex component of ARX ciphers, and the division property is the most effective tool for detecting integral distinguishers of this operation. Thus, having a precise division property model for modular addition is crucial in the search for integral distinguishers in ARX ciphers. The most frequently used models (among many others) for handling modular addition were specified in [41], and, more recently, in [1]. However, both approaches employ the two-subset bit-based division property (2BDP), which has certain disadvantages and may overlook certain balanced bits. Recently, Hu and Yap [25] developed another approach by applying Braeken and Semaev's theorem [8], which invokes the so-called "perfect MP (monomial prediction) model" for modular addition.

Using this MP model, we have evaluated the upper bounds on the minimum and maximum algebraic degree of the coordinate functions for certain ARX-box designs (including their inverses computed using the bit-division property in [1]), namely

for Alzette and the proposed AFS instances. These upper bounds (obtained without a time-consuming operation of removing monomials appearing an even number of times), displayed in Table 8, are expected to be close to the actual values, since it was demonstrated that the MP model is exact [25].

Remark 2 The mentioned upper bounds are not correct only if the monomials of the largest degree are all canceled due to their even parity of appearances. The main problem is that when modeling MP, using e.g. MILP or SMT tools, there are multiple propagation paths that all give rise to the presence of a certain fixed output monomial. Instead of embedding complicated multiple path conditions as the optimization goal of MILP, we rather impose the conditions on the algebraic degree of the inputs and then the optimization goal of the MILP model is to maximize the output degree.

Consequently, certain AFS structures may even achieve the maximum algebraic degree in all their coordinates. Notice also that the inverse of AFS S-box in general attain larger algebraic degree than the inverse of Alzette.

Table 8 Upper bounds on the minimum and maximum degree of the coordinate functions of Alzette and AFS. CS and RS denote the choice of configurational and rotational sequence, respectively. LB represents the lower bound on algebraic degree of all coordinate functions for a 4-round S-box.

ARX-box	CS	RS	Rounds	1	2	3	4	LB	Inverse rounds	1	2	3	4
Alzette	10101010	[31,24,17,17,0,31,24,16]	min	1	10	42	63	63	min	1	1	32	46
			max	32	62	63	63		max	32	62	63	63
AFS-64	11000011	[1,1,16,24,31,31,8,16]	min	1	17	35	63	63	min	1	9	26	61
			max	63	63	63	63		max	61	61	61	63
		[1,1,16,8,31,31,24,16]	min	1	17	31	63	63	min	1	25	40	61
			max	63	63	63	63		max	61	61	61	63
		[17,24,1,1,16,31,24,0]	min	1	26	49	58	58	min	1	25	48	61
			max	63	63	63	63		max	61	61	61	63
		[17,8,1,1,16,31,8,0]	min	1	10	42	57	57	min	1	9	41	61
			max	62	62	62	63		max	61	61	61	63
AFS-32	11000011	[1,8,14,0,12,7,15,9]	min	1	9	16	31	31	min	1	16	23	28
			max	31	31	31	31		max	30	30	30	31
		[7,14,8,1,13,0,9,15]	min	1	15	23	31	31	min	1	10	15	29
			max	31	31	31	31		max	30	30	30	31
	10100011	[0,7,14,15,8,1,2,9]	min	1	6	18	24	24	min	1	4	20	31
			max	16	30	30	31		max	31	31	31	31
		[8,0,2,1,9,14,7,15]	min	1	3	15	30	29	min	1	8	25	29
			max	16	30	30	31		max	31	31	31	31

Moreover, employing the bit-based division property, we have performed a computer-based search for integral characteristics that cover maximal number of rounds, for all AFS versions. The results given in Table 9 show that both Alzette and AFS instances admit integral characteristics, when they are composed of up to six rounds. On the other hand, for two chained AFS instances no integral characteristics could be found with state-of-the-art algorithms. It is worth mentioning that certain instances of AFS perform better than Alzette in this context, meaning that the number of rounds for which an integral characteristic may exist is smaller.

Table 9 The number of rounds of the longest integral characteristic of AFS and its inverse searched using bit-based division property, where CS represents the configurational sequence and RS represents the rotational sequences

ARX-box	CS	RS	# Rounds	# Rounds - Inverse
Alzette	10101010	[31,24,17,17,0,31,24,16]	6	5
AFS-64	11000011	[1,1,16,24,31,31,8,16]	6	6
		[1,1,16,8,31,31,24,16]	6	5
		[17,24,1,1,16,31,24,0]	5	5
		[17,8,1,1,16,31,8,0]	6	6
AFS-32	11000011	[1,8,14,0,12,7,15,9]	7	5
		[7,14,8,1,13,0,9,15]	5	4
	10100011	[0,7,14,15,8,1,2,9]	6	6
		[8,0,2,1,9,14,7,15]	6	5

5.4 Small-Scale Experimental Validation

Precisely analyzing the differential and linear properties of large S-boxes (e.g. 32-bit and 64-bit versions in the AFS family) is computationally infeasible due to the sheer size of their input/output spaces—exhaustively computing the Difference Distribution Table (DDT) or Linear Approximation Table (LAT) for such dimensions is obviously impractical. To provide concrete evidence supporting the cryptographic strength inherent in the AFS design, and to better understand why these S-boxes perform well, we have conducted a small-scale experimental validation. This involves scaling down the core design structure of the AFS S-boxes to an 8-bit variant.

The AFS design consists of a structure (defined by a configurational sequence) and a rotation sequence. For the 8-bit experiment, we explored a constrained space of rotational sequences defined as $\{0,1,2,3\}^8$, which gives $4^8 = 65,536$ distinct rotational sequences. Moreover, we examined all possible 8-bit configurational sequences containing at least one modular addition, represented by the set $\{0,1\}^8 \setminus \{0\}^8$. This gives us $2^8 - 1 = 255$ different structural configurations. For each of the 255 configurational sequences, we exhaustively iterated through all 65,536 rotational sequences. For every resulting 8-bit S-box instance, we computed its complete DDT and LAT to thoroughly investigate its differential and linear properties. To eliminate the influence of round constants, we uniformly set all round constants to zero. The computation of LATs was parallelized using GPUs (NVIDIA RTX 3090), while DDT computation was parallelized on CPUs (Intel(R) Xeon(R) Platinum 8582C). For these 8-bit instances, DDT computation is inexpensive on CPUs, whereas LAT evaluation consists of independent Walsh-type inner products and benefits from GPU parallelism. The entire experiment took approximately one day to complete.

To evaluate the overall cryptographic strength of each configurational sequence across all rotational sequences, we have considered the following averaging approach that we call *composite score*. For each S-box instance, we calculated its maximum differential probability (P_d) and maximum linear probability (P_l) (derived from the squared maximum linear correlation). We then averaged these maximum probabilities over the 65,536 rotational sequences for a given configurational sequence, obtaining $\overline{P_d}$

and $\overline{P_l}$. The composite score was calculated as the Euclidean norm (L2 norm) of these average probabilities: $\sqrt{\overline{P_d}^2 + \overline{P_l}^2}$. A lower score indicates better overall resistance.

Our statistical analysis reveals an interesting phenomenon regarding the number of modular additions used in the structure. Intuitively, one might expect that more modular additions (non-linear operations) would lead to better scores, similarly to how using more S-boxes in a cipher generally enhances security. However, as shown in Fig. 13, our results indicate that the improvement in the composite score starts to plateau and, even, slightly degrade when the number of additions exceeds 6. The structures using exactly 6 modular additions achieved the best average scores for this rescaled version. Furthermore, we observed that the configurational sequences achieving the best scores for a given number of additions often exhibit structural patterns, such as near-palindromic forms. The best sequences found for 1 to 8 additions are: 01000000, 10000001, 11000001, 11000011, 11010011, 11011011, 11110111, and 11111111, respectively. Notably, the sequence 11000011, which we previously identified as a high-performing structure for the AFS64 S-box, is the best-scoring configuration among those using 4 additions in the 8-bit experiments.

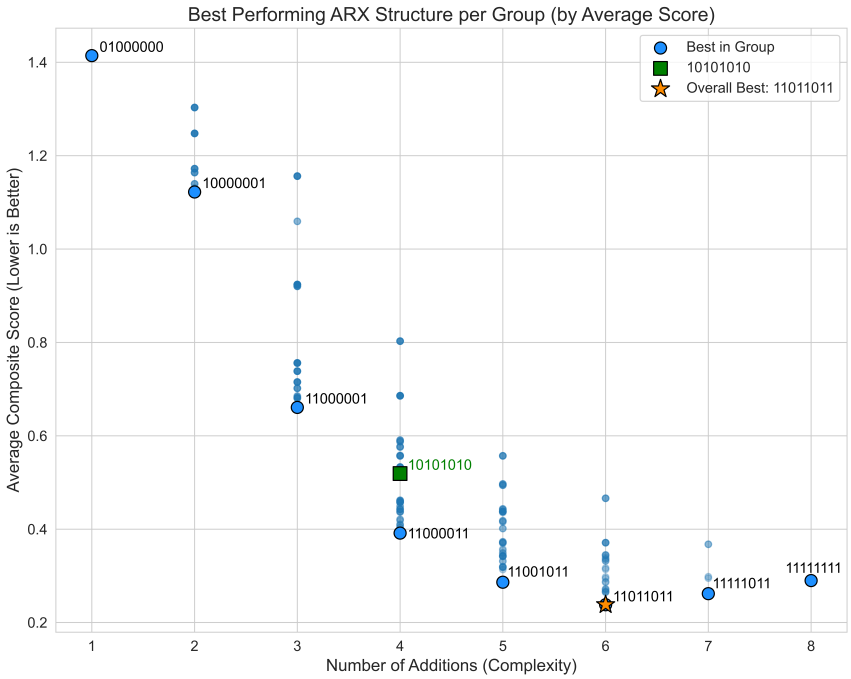


Fig. 13 Impact of the number of modular additions on the composite security score for 8-bit S-boxes.

We also compared our results with the Alzette S-box structure, which uses the configurational sequence 10101010. On average (across the 65,536 rotational sequences), 10101010 yielded a significantly worse composite score than 11000011. This suggests that the differential and linear properties of our chosen AFS structure are less sensitive to variations in the rotational sequence compared to the Alzette structure, meaning our structure maintains good properties over a wider range of rotations.

6 An Application of AFS S-boxes

To demonstrate the practical advantages of our AFS S-box family, we integrate them into the SPARX-128 cipher [14] and the SPARKLE permutation [2] as replacements for their native S-boxes (SPECKEY and Alzette, respectively).

Both primitives employ the Long Trail Design Strategy (LTS)—a dual of the classical Wide Trail Strategy (WTS) that is applicable to ARX constructions. While WTS prescribes small S-boxes with heavy linear layers, LTS advocates large ARX-based S-boxes with sparse linear layers. Using the so-called Long Trail Argument, one can bound the maximum differential and linear probabilities by decomposing trails into sequences of active S-boxes (long trails), where the total probability is bounded by the product of individual long trail probabilities.

Proving security for LTS-based primitives requires finding the optimal Long Trail Decomposition (LTD) to bound differential/linear probabilities. However, existing methods rely on truncated word-level analysis that ignores bit-level propagation within S-boxes and linear layers, retaining impossible trails and yielding loose bounds. To address this shortfall, we introduce the Bit-Based Long Trail Decomposition (BLTD), which is a framework that precisely captures differential and linear propagation through both linear layers and within S-boxes at bit level. Unlike previous truncated-trail methods, our approach yields tighter bounds by modeling exact bit-level dependencies rather than word-level approximations.

6.1 A Bit-Based SMT Modeling Approach for LTS

Building on the LTS definition in Section 2.3, this section introduces a bit-based SMT model for computing an optimal long-trail decomposition.

We now present the LTS-specific formulation for our bit-based model. The model is presented using SMT, however, it can readily be extended to a MILP-based model. In contrast to other methods, our method does not require any prior study of the linear layer's properties.

Weight Reformulation. To obtain a proper SMT encoding, let us consider additive weights via $-\log_2$. Namely, define $p(S^i) := -\log_2(\mathrm{MEDCP}(S^i))$ and incremental weights $P_I(j) := p(S^j) - p(S^{j-1})$ with $p(S^0) := 0$. The total weight for decomposition $\{t_i\}_{i\geq 1}$ becomes:

$$W := \sum_{i\geq 1} t_i \sum_{j=1}^{i} P_I(j). \tag{2}$$

This additive structure assigns precise costs per active S-box based on its trail position, which allows an efficient SMT encoding.

6.1.1 SMT Model for Optimal Differential LTD

Our differential SMT model consists of a differential propagation model with three key components:

1) Cipher propagation constraints which ensure valid differential trails;

2) Long trail constraints to track where the trail starts and how to extend it;
3) Objective Function: minimizing total weight $\sum P_I[LT_i]$.

Cipher Propagation Constraints

Variables: For each S-box: $AS \in \{0,1\}$ (active status); $LT \in \mathbb{Z}_{\geq 0}$ (position in trail, $LT = 0$ iff inactive); constant array $P_I[j] = p(S^j) - p(S^{j-1})$ storing incremental weights, where $p(S^i) = -\log_2(\text{MEDCP}(S^i))$.
General constraints: The active status is determined by the m-bit input differential α:

IF $(\alpha = 0\text{bin}\mathbf{0}^m)$ THEN $(AS = 0\text{bin}0)$ ELSE $(AS = 0\text{bin}1)$ ENDIF.

Long Trail Constraints

Initialization (round $r = 0$):

IF $(AS = 0\text{bin}1)$ THEN $(LT = 0\text{hex}01)$ ELSE $(LT = 0\text{hex}00)$ ENDIF.

Propagation (round $r > 0$): For LTS-specific linear layers with at most one direct copy per row/column [14], we distinguish between the cases: direct connection and a conditionally extended connection, shown in Fig. 14.

In the former case, the input comes exclusively from a single copied S-box, which is modeled as

IF $(AS_1 = 0\text{bin}1)$ THEN $(LT_1 = \text{BVPLUS}(8, LT_0, 0\text{hex}01))$ ELSE $(LT_1 = 0\text{hex}00)$ ENDIF.

In the latter case, we have $\gamma = \alpha \oplus \beta$, where α is copied and β passes through the linear transformation L, thus

IF $((AS_1 = 0\text{bin}1)$ AND $(\beta = 0\text{bin}\mathbf{0}^m))$ THEN $(LT_1 = \text{BVPLUS}(8, LT_0, 0\text{hex}01))$
ELSE IF $((AS_1 = 0\text{bin}1)$ AND $(\beta \neq 0\text{bin}\mathbf{0}^m))$ THEN $(LT_1 = 0\text{hex}01)$
ELSE $(LT_1 = 0\text{hex}00)$ ENDIF ENDIF.

Objective Function

The objective function is given by minimizing the total weight

$$W_{\min} = \min \sum_{i=0}^{t-1} P_I[LT_i].$$



Fig. 14 Scenarios for long trail propagation through a specific linear layer.

Alternatively, decide $\sum P_I[LT_i] \leq P_{\text{threshold}}$ for a threshold-based verification. This model is significantly faster than general optimal characteristic search, for instance, it yields optimal decompositions instantly for any round count of the SPARX-128 and SPARKLE variants.

6.1.2 SMT Model for Optimal Linear LTD

The linear model follows the same structure as the differential case, with two key adaptations. First, an S-box is active if its output linear mask is non-zero (rather than input differential). Second, due to the duality between linear and differential propagation, cases for the linear layer propagation are swapped relative to the differential model, where the duality can be described as *branching* $\leftrightarrow$ XOR. The initialization and objective function remain unchanged.

6.1.3 Accelerating Optimal Characteristic Search

To further improve the search for optimal differential/linear characteristics on LTS-based ciphers, we propose two pruning strategies which allow our long trail decomposition model to provide tight bounds efficiently when integrated with the SMCS strategy (Section 3).

Static Pruning. Pre-calculate W_{LTDmin} (minimum LTD weight) and use it as a fixed threshold in SMCS:

$$W_{\text{True}} \geq W_{\text{LTDmin}}.$$

Dynamic Pruning. Integrate LTD constraints directly into the SMCS search enforcing:

$$W_{\text{True}} \geq W_{\text{LTD}},$$

where W_{LTD} is the LTD weight of the current characteristic. Since BLTD's optimal solution is always a lower bound for the true optimal characteristic (as per the Long Trail Argument), SMCS can use the current BLTD weight to prune branches dynamically. Note that when BLTD finds its optimum, SMCS may not yet have found the true optimum, but the bound relationship always holds.

6.2 Security Evaluation with AFS S-boxes

We evaluate SPARX-128 and SPARKLE using our BLTD model described in Section 6.1 to derive tight differential/linear bounds, when replacing their native S-boxes with

AFS S-boxes presented in Section 5.2. The results are displayed in the context of differential cryptanalysis. Corresponding results on linear cryptanalysis are displayed in Appendix D.

6.2.1 Evaluation of SPARX-128

First, we consider the minimum weights of the differential and linear long trails of SPARX-128 using the original methods [14], and compare them with the minimum weights of trails obtained using BLTD. Then, we replace the SPECKEY S-box by the AFS-32 variants attaining optimal LTD (AFS-32 instances $[1, 8, 14, 0, 12, 7, 15, 9]$ and $[7, 14, 8, 1, 13, 0, 9, 15]$). Table 10 and Table D2 present these results for differential and linear bounds, respectively. In these tables, each step comprises 4 S-box rounds plus linear layer.

Table 10 Optimal Differential Long Trail Weights ($W_{\min}$) for SPARX-128.

Step	SPECKEY		AFS-32 (11000011)	
	Original [14]	BLTD	$[1, 8, 14, 0, 12, 7, 15, 9]$	$[7, 14, 8, 1, 13, 0, 9, 15]$
1	–	5	10	10
2	–	24	31	31
3	–	52	69	67
4	–	78	106	105
5	138	106	142	140
6	–	130	160	156
7	–	154	197	194
8	–	178	230	224
9	–	202	267	262
10	–	226	288	283

Remark 3 From the data presented in Table 10 and Table D2, we draw two primary conclusions. First, we observe that the MEDCP and MELCC bounds for 5 steps, as derived from the (original) optimal truncated decomposition by the designers of SPARX-128, were underestimated (i.e., the weights were lower than what our model found). This implies that the designers overestimated the security strength of the SPARX-128 algorithm at 5 steps against differential and linear cryptanalysis. Second, after instantiating two of the AFS-32 S-boxes within SPARX-128, the MEDCP and MELCC bounds (the W_{min} values) at each step are significantly improved.

6.2.2 Evaluation of SPARKLE

Note that improvements in the MEDCP and MELCC values directly impact the final differential and linear security bounds derived from the optimal long trail decomposition. To isolate this impact and showcase the efficiency of our model, our evaluation proceeds in two stages. First, we use the original Alzette MEDCP/MELCC values into BLTD and compare it with previous methods. Next, we consider the bounds for

Alzette presented in Table 1. We then run two BLTD models for SPARKLE, whose S-box is either the Alzette S-box (with the updated bounds) or an instance of the AFS-64 family, namely, AFS-64(11000011) with rotation values $[17, 24, 1, 1, 16, 31, 24, 0]$.

These two stages are summarized in Tables 11 and D3, and Tables 12 and D4, respectively. For an n-bit SPARKLE instance, the MEDCP target is to reach a probability at most 2^{-n}, i.e., an optimal long-trail weight $W_{\min} \geq n$, in as few steps as possible. For MELCC, the target is to reach a correlation bound at most $2^{-n/2}$, i.e., $W_{\min} \geq n/2$. Thus, the differential target weights are 256, 384, and 512 for SPARKLE-256, -384, and -512, respectively, while the linear target weights are 128, 192, and 256. In the first stage, we note that our bit-level SMT model matches or exceeds prior methods [2, 14], with notable improvements meaning that truncated methods underestimate security (e.g., SPARKLE-512 steps 5–13). Upon looking at Table 12 and Table D4, we note that SPARKLE attains substantially tighter bounds when used with AFS-64 than with Alzette. For instance, SPARKLE-256 reaches 2^{-266} security at 8 steps vs. 2^{-228} for Alzette; SPARKLE-512 achieves 2^{-554} at 11 steps vs. 2^{-491}. Fig. 15 visualizes these gains.

Table 11 Comparison of Optimal Differential Long Trail Weights (W_{min}) for SPARKLE versions for original Alzette MEDCP parameters.

Perm. / Steps	1	2	3	4	5	6	7	8	9	10	11	12	13	Method
SPARKLE-256	6	32	64	88	140	168	192	216	≥256	≥256	≥256	≥256	≥256	[2]
	6	32	64	88	140	168	192	216	256	280	320	344	384	[14]
	6	32	64	88	140	168	192	216	256	280	320	344	384	BLTD
SPARKLE-384	6	32	70	100	178	200	230	260	326	356	≥384	≥384	≥384	[2]
	6	32	70	100	166	200	230	260	308	336	374	412	440	[14]
	6	32	70	100	178	200	230	260	326	356	390	420	486	BLTD
SPARKLE-512	6	32	76	112	210	232	268	276	295	424	433	496	≥512	[2]
	6	32	76	112	168	208	232	272	312	336	376	416	440	[14]
	6	32	76	112	210	232	268	304	396	424	460	496	588	BLTD

Table 12 Comparison of Optimal Differential Long Trail Weights (W_{min}) for SPARKLE versions via BLTD, using updated Alzette and AFS-64(11000011)[17,24,1,1,16,31,24,0].

Permutation / Steps	1	2	3	4	5	6	7	8	9	10	11	12	13	S-box
SPARKLE-256	6	34	71	92	148	182	207	228	275	296	343	364	411	Alzette
	10	37	86	116	168	205	236	266	314	344	386	416	464	AFS-64
SPARKLE-384	6	34	77	104	188	220	247	274	349	376	417	444	519	Alzette
	10	37	96	136	214	243	283	323	400	430	470	510	587	AFS-64
SPARKLE-512	6	34	83	116	222	254	287	320	423	456	491	524	627	Alzette
	10	37	106	156	251	280	330	380	475	504	554	604	699	AFS-64

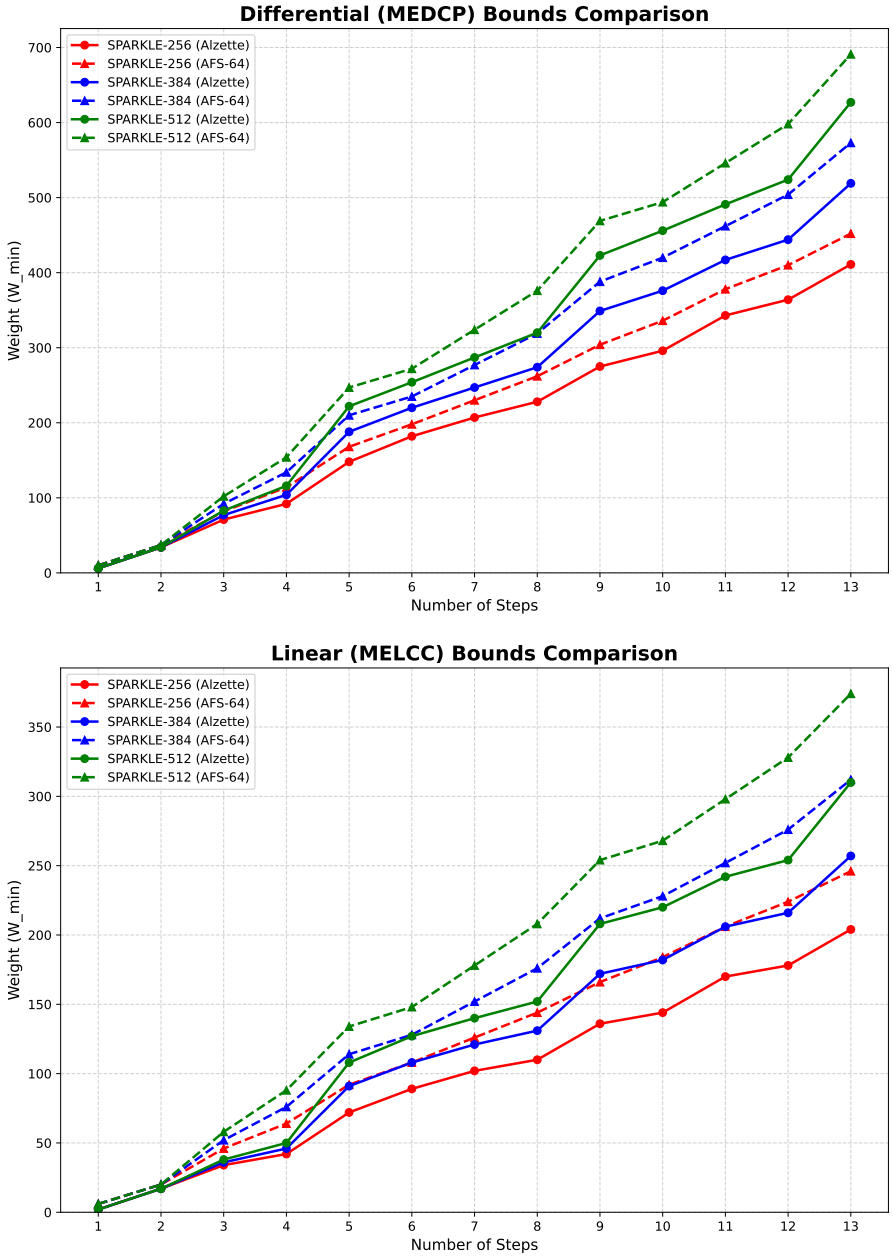


Fig. 15 Visual comparison of optimal long trail weights for SPARKLE variants using updated Alzette and our AFS-64 S-box.

Remark 4 From the data in Tables 11, 12, D3 and D4 and from Fig. 15, it is clear that replacing the Alzette S-box with our AFS-64 instance results in a very significant improvement of the security bounds against differential and linear cryptanalysis for all versions of SPARKLE at each step. Overall, using our AFS-64 S-box allows for a reduction in the number of steps required to achieve a target security level, or conversely, provides a significantly higher security margin for the same number of steps.

7 Conclusions

An efficient search strategy for finding differential/linear characteristics for ARX-based S-boxes of large size (named SMCS), combining the MILP and SMT approach, has been proposed. As a proof of concept, by employing this strategy, we could establish tight bounds for the maximum expected differential characteristic probabilities (resp. the maximum expected linear correlations) of Alzette for up to 15 rounds (resp. 14 rounds). From the design perspective, our method based on the so-called configurational encoding has been efficiently employed in specifying the AFS family of S-boxes that use the ARX-Feistel structure. Notably, the 64-bit version AFS-64 offers improved resistance to differential and linear cryptanalysis while preserving the same software and hardware costs as Alzette. Similarly, the 32-bit version AFS-32 provides significantly better resistance to differential/linear cryptanalysis than the S-box SPECKEY used in SPARX-128. These S-boxes have similar hardware performance, whereas certain instances within AFS-32 offer better software efficiency. More importantly, generalized Feistel structures are quite suitable for providing constituent modules for the design of ARX S-boxes. The preliminary results are quite promising, even though an extended security analysis is required. Finally, we proposed the first bit-based SMT model for optimal long trail decomposition, which can also be easily extended to other automated analysis techniques like MILP. Using our model, we provided more precise differential and linear security bounds for SPARX-128 and SPARKLE. Furthermore, by replacing the SPECKEY S-box in SPARX-128 and the Alzette S-box in SPARKLE with our AFS-32 and AFS-64 instances, respectively, we demonstrated the significant advantages of the new primitives in resisting differential and linear cryptanalysis.

Appendix A SMT models for differential/linear cryptanalysis of ARX-based S-boxes

A.1 SMT model for differential cryptanalysis of ARX primitives

Differential SMT Model 1 (XOR). Assume that n-bit input and output differences are α, β and γ, respectively. The SMT model for the XOR operation where $\alpha \oplus \beta = \gamma$ is as follows:

$$\begin{aligned} &\alpha, \beta, \gamma : \text{BITVECTOR}(n); \\ &\text{ASSERT}(\gamma = \text{BVXOR}(\alpha, \beta)). \end{aligned} \tag{A1}$$

Differential SMT Model 2 (Branching). Assume that n-bit input and output differences are α, β and γ respectively. The SMT model for the branching operation is as follows:

$$\begin{aligned} &\alpha, \beta, \gamma : \text{BITVECTOR}(n); \\ &\text{ASSERT}((\alpha = \beta)\text{AND}(\beta = \gamma)). \end{aligned} \tag{A2}$$

Differential SMT Model 3 (Rotation). For an n-bit difference α, the SMT model for the rotation of α to the right by k-bits $k \in \{1, \ldots, n-1\}$ is as follows:

$$\begin{aligned} &\alpha : \text{BITVECTOR}(n); \\ &\alpha[k-1:0]@\alpha[n-1:k]. \end{aligned} \tag{A3}$$

Lipmaa and Moriai [26] showed that, given n-bit input differences α, β and γ, if $(\alpha, \beta \mapsto \gamma)$ is an effective differential, then they must satisfy the following:

$$eq(\alpha \ll 1, \beta \ll 1, \gamma \ll 1) \wedge (\alpha \oplus \beta \oplus \gamma \oplus (\beta \ll 1)) = 0; \tag{A4}$$

$$eq(\alpha, \beta, \gamma) = (\neg\alpha \oplus \beta) \wedge (\neg\alpha \oplus \gamma). \tag{A5}$$

The differential probability $xdp^{\boxplus}(\alpha, \beta \mapsto \gamma)$ can then be computed using the following formula:

$$xdp^{\boxplus}(\alpha, \beta \mapsto \gamma) = 2^{-w^*(\neg eq(\alpha,\beta,\gamma))}, \tag{A6}$$

where $w^*(x)$ denotes the Hamming weight of x, excluding the most significant bit $x[n-1]$.

Differential SMT Model 4 (Modular addition). The set of above conditions can be turned into an SMT model as follows:

$$\begin{aligned} &\alpha, \beta, \gamma, p, DP : \text{BITVECTOR}(n); \\ &\text{ASSERT}(p = \text{BVXOR}(\sim(\alpha[n-2:0]@\alpha[n-1:n-1]), \\ &\beta[n-2:0]@\beta[n-1:n-1]), \\ &\&\ \text{BVXOR}(\sim(\alpha[n-2:0]@\alpha[n-1:n-1]), \\ &\gamma[n-2:0]@\gamma[n-1:n-1])); \\ &\text{ASSERT}(p\ \&\ \text{BVXOR}(\text{BVXOR}(\text{BVXOR}(\alpha, \beta), \gamma), \\ &\beta[n-2:0]@\beta[n-1:n-1]) = 0\text{bin}0\cdots0); \\ &\text{ASSERT}(DP = \text{BVPLUS}(n, 0\text{bin}0\cdots0@\sim(p[n-1:n-1]), \\ &\cdots, 0\text{bin}0\cdots0@\sim(p[1:1]))). \end{aligned} \tag{A7}$$

Differential SMT Model 5 (Extra constraint). To ensure that the initial m-bit input difference X_0 is not entirely zero, the following additional constraints are introduced:

$$\begin{array}{l} X_0 : \text{BITVECTOR}(m); \\ \text{ASSERT}(\text{NOT}(X_0 = 0\text{bin}0 \cdots 0)). \end{array} \tag{A8}$$

A.2 SMT model for linear cryptanalysis of modular addition

In [34, 45], Wallén proposed an algorithm for computing linear approximations after modulo 2^n addition. Schulte-Geers [35] also provided a simple explicit method for computing the linear correlation after the modulo 2^n addition. For n-bit input and output masks Λ_α, Λ_β and Λ_γ, if $(\Lambda_\alpha, \Lambda_\beta \mapsto \Lambda_\gamma)$ constitutes an effective linear approximation, it must satisfy:

$$\begin{array}{l} (\Lambda_\gamma \oplus \Lambda_\alpha)\&\Gamma = (\Lambda_\gamma \oplus \Lambda_\alpha); \\ (\Lambda_\gamma \oplus \Lambda_\beta)\&\Gamma = (\Lambda_\gamma \oplus \Lambda_\beta); \\ \Gamma = \Psi(\Lambda_\gamma \oplus \Lambda_\alpha \oplus \Lambda_\beta), \end{array} \tag{A9}$$

where $\Psi(x) = (0, x[n-1], x[n-1] \oplus x[n-2], \ldots, x[n-1] \oplus \cdots \oplus x[1])$. The absolute value of the linear approximation correlation $|cor^{\boxplus}(\Lambda_\alpha, \Lambda_\beta, \Lambda_\gamma)|$ can be calculated using the following formula:

$$|cor^{\boxplus}(\Lambda_\alpha, \Lambda_\beta, \Lambda_\gamma)| = 2^{-w(\Gamma)}, \tag{A10}$$

where $w(x)$ denotes the Hamming weight of x.

Linear SMT Model 4 (Modular addition). The above set of conditions can be turned into an SMT model as follows:

$$\begin{array}{l} \Lambda_\alpha, \Lambda_\beta, \Lambda_\gamma, p, \Gamma, LP : \text{BITVECTOR}(n); \\ \text{ASSERT}(p = \text{BVXOR}(\text{BVXOR}(\Lambda_\gamma, \Lambda_\alpha), \Lambda_\beta)); \\ \text{ASSERT}(\Gamma = 0\text{bin}0@p[n-1:n-1]@\text{BVXOR}(\\ p[n-1:n-1], p[n-2:n-2])@\cdots@\text{BVXOR}(\\ \cdots \text{BVXOR}(p[n-1], p[n-2]), \cdots, p[1]); \\ \text{ASSERT}(\text{BVXOR}(\Lambda_\gamma, \Lambda_\alpha) = \text{BVXOR}(\Lambda_\gamma, \Lambda_\alpha)\ \&\ \Gamma); \\ \text{ASSERT}(\text{BVXOR}(\Lambda_\gamma, \Lambda_\beta) = \text{BVXOR}(\Lambda_\gamma, \Lambda_\beta)\ \&\ \Gamma); \\ \text{ASSERT}(LP = \text{BVPLUS}(n, 0\text{bin}0 \cdots 0@\Gamma[n-1:n-1], \\ \cdots, 0\text{bin}0 \cdots 0@\Gamma[0:0])). \end{array} \tag{A11}$$

Appendix B Differential Trails in AFS

In this appendix, we provide the differential trails of some instances of AFS together with their maximum expected differential probability. The results for AFS-64 and AFS-32 are displayed in Table B1.

Table B1 The input and output differences α, β (in hex.) of all differential trails of AFS corresponding to the max. expected differential trail probability p for 4 rounds.

ARX-box	CS	RS	α	β	$-\log_2(p)$
AFS-64	11000011	[1,1,16,24,31,31,8,16]	40A0810180410000	8000000000008080	11
			0000005000000020	80008100800C080	
			0000005000000020	80008100810C080	
			0000005000000020	8000810081004080	
			0000005000000020	8000810081014080	
			0140A08100804100	0080000080000080	
			4000A00080004000	0280000180010282	
			0050000000200000	8100800040808101	
			0050000000200000	81008000C0808101	
			810140A000008041	0000800080800000	
			A000000040000000	0100010201020081	
			A000000040000000	0100010201020181	
			A000000040000000	0100010201020281	
			A000000040000000	0100010201020381	
			A0A0004040400080	8100000000008101	
			C000A00080004000	0280000180010282	
			C0A0810180410000	8000000000008080	
		[1,1,16,18,31,31,24,16]	0000005000000020	8000810041008081	11
			0000005000000020	80008100C1008081	
			0080A14101004080	0000008080800000	
			80A1410000408001	0000800080000080	
			4000A00080004000	0000028182810002	
			410080A180010040	8000000000808000	
			0050000000200000	81008000808C100	
			0050000000200000	810080008081C100	
			0050000000200000	8100800080804100	
			0050000000200000	8100800080814100	
			A000000040000000	0100010200820101	
			A000000040000000	0100010201820101	
			A000000040000000	0100010202820101	
			A000000040000000	0100010203820101	
			A04000A040800040	8000010001008001	
			C000A00080004000	0000028182810002	
			C10080A180010040	8000000000808000	
		[17,24,1,1,16,31,24,0]	8040808001000081	8080000080808000	10
			8041808001000081	8080000080808000	
			A0502000400000A0	8000010080000101	
		[17,8,1,1,16,31,8,0]	8080804000800101	8000008080008080	10
			A000205040A00000	8100000081000001	
			8081804000800101	8000008080008080	
AFS-32	11000011	[1,8,14,0,12,7,15,9]	41428280	82000141	10
			C1428280	82000141	
		[7,14,8,1,13,0,9,15]	80301040	80489091	10
			84424102	80408081	
			84C24102	80408081	
			80101040	80489091	
	10100011	[0,7,14,15,8,1,2,9]	80840184	41008021	9
			81840184	41008021	
			82840184	41008021	
			83840184	41008021	
			89050901	90004048	
			89050901	9000C048	
		[8,0,2,1,9,14,7,15]	90D0D000	84000801	9
			90D0D000	84000803	
			9050D000	84000801	
			9050D000	84000803	

Appendix C Optimal structural configurations for the AFS family

The instances of AFS family, with $N = 4$ iterations of Feistel-like modules, exhibiting the best performance are: AFS-64 with structural encoding '11000011' and AFS-32 with structural encoding '10100011', shown in Fig. C1 and Fig. C2. Note that the diagram of AFS-32(11000011) can be equally depicted by Fig. C1. In both cases, the standard round constants (same as in Alzette and SPECKEY) are used. Namely,

$$c_0 = 86\text{cd}, \quad c_1 = 7\text{f}03, \quad c_2 = 71\text{eb}, \quad c_3 = \text{bb}12,$$
$$c_4 = \text{ced}4, \quad c_5 = \text{c}3\text{f}6, \quad c_6 = 3\text{add}, \quad c_7 = 8\text{e}5\text{c}.$$

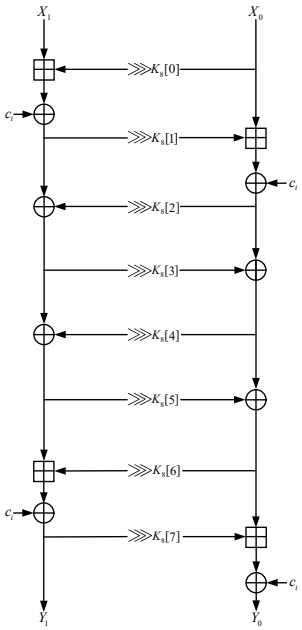


Fig. C1 AFS-64(11000011).

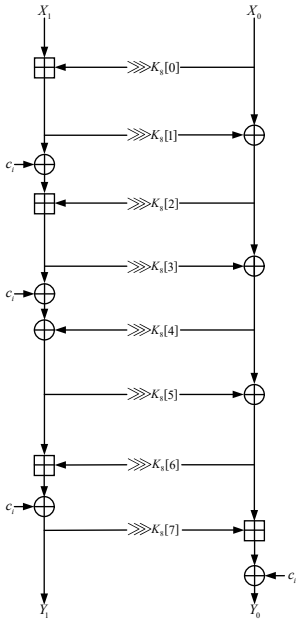


Fig. C2 AFS-32(10100011).

Appendix D Linear Cryptanalysis Results for SPARX-128 and SPARKLE

Table D2 Optimal Linear Long Trail Weights ($W_{\min}$) for SPARX-128.

Step	SPECKEY		AFS-32 (11000011)	
	Original [14]	BLTD		
			$[1, 8, 14, 0, 12, 7, 15, 9]$	$[7, 14, 8, 1, 13, 0, 9, 15]$
1	–	3	5	5
2	–	12	15	15
3	–	26	34	34
4	–	40	53	53
5	68	54	69	69
6	–	64	78	78
7	–	78	97	97
8	–	90	114	114
9	–	102	130	130
10	–	114	141	141

Table D3 Comparison of Optimal Linear Long Trail Weights (W_{min}) for SPARKLE versions, using original Alzette MELCC parameters.

Permutation / Steps	1	2	3	4	5	6	7	8	9	10	11	12	13	Method
SPARKLE-256	2	17	23	42	57	72	91	106	125	140	159	174	193	[2]
	2	17	23	42	57	72	91	106	125	≥128	≥128	≥128	≥128	[14]
	2	17	23	42	57	72	91	106	125	140	159	174	193	BLTD
SPARKLE-384	2	17	25	46	76	89	110	131	161	174	≥192	≥192	≥192	[2]
	2	17	25	46	72	89	106	125	140	159	174	193	208	[14]
	2	17	25	46	76	89	110	131	161	174	195	216	246	BLTD
SPARKLE-512	2	17	27	50	93	106	129	152	195	208	231	254	≥256	[2]
	2	17	27	50	72	93	106	127	140	161	174	195	208	[14]
	2	17	27	50	93	106	129	152	195	208	231	254	297	BLTD

Table D4 Comparison of Optimal Linear Long Trail Weights (W_{min}) for SPARKLE versions, comparing updated Alzette with AFS-64(11000011)[17,24,1,1,16,31,24,0]

Permutation / Steps	1	2	3	4	5	6	7	8	9	10	11	12	13	S-box
SPARKLE-256	2	17	34	42	72	89	102	110	136	144	170	178	204	Alzette
	6	20	48	66	92	112	130	148	172	192	212	230	252	AFS-64
SPARKLE-384	2	17	36	46	91	108	121	131	172	182	206	216	257	Alzette
	6	20	54	78	116	132	156	180	218	234	258	282	320	AFS-64
SPARKLE-512	2	17	38	50	108	127	140	152	208	220	242	254	310	Alzette
	6	20	60	90	136	152	182	212	258	274	304	334	380	AFS-64

Appendix E ARX boxes based on generalized Feistel modules

Table E5 Differential and linear properties of 64-bit S-boxes with some generalized ARX-Feistel modules, where CS and RS represent the configurational and rotational sequences, respectively; the first row specifies MEDCP and the second row MELCC.

CS	RS	4-Round	5-Round	8-Round
22333322	[0,0,8,8,9,9,2,2,1,1,14,14,15,15,7,7]	2^{-10} 2^{-6}	2^{-19} 2^{-11}	$\leq 2^{-35}$ 2^{-21}
	[0,0,2,2,1,1,9,9,8,8,7,7,15,15,14,14]	2^{-10} 2^{-6}	2^{-19} 2^{-11}	- $\leq 2^{-19}$
	[0,0,8,8,1,1,2,2,9,9,14,14,7,7,15,15]	2^{-10} 2^{-6}	2^{-19} 2^{-11}	- $\leq 2^{-19}$
	[0,0,9,9,8,8,2,2,1,1,14,14,15,15,7,7]	2^{-10} 2^{-6}	2^{-19} 2^{-11}	- $\leq 2^{-19}$
	[9,9,2,2,1,1,8,8,7,7,15,15,14,14,0,0]	2^{-10} 2^{-6}	2^{-19} 2^{-11}	- $\leq 2^{-18}$
	[0,0,9,9,15,15,14,14,8,8,7,7,2,2,1,1]	2^{-10} 2^{-6}	2^{-19} 2^{-11}	- 2^{-18}
	[1,1,7,7,8,8,14,14,15,15,2,2,9,9,0,0]	2^{-10} 2^{-6}	2^{-19} 2^{-11}	- -
	[0,0,2,2,1,1,9,9,15,15,14,14,8,8,7,7]	2^{-10} 2^{-6}	2^{-19} 2^{-11}	- -
	[7,7,14,14,15,15,8,8,9,9,1,1,2,2,0,0]	2^{-10} 2^{-6}	2^{-19} 2^{-11}	- -
	[0,0,14,14,15,15,7,7,1,1,2,2,8,8,9,9]	2^{-10} 2^{-6}	2^{-19} 2^{-11}	- -
	[0,0,9,9,8,8,2,2,1,1,7,7,15,15,14,14]	2^{-10} 2^{-6}	2^{-19} 2^{-11}	- -
	[0,0,7,7,1,1,2,2,8,8,9,9,14,14,15,15]	2^{-10} 2^{-6}	2^{-19} 2^{-11}	- -
44555544	[0,0,8,8,9,9,2,2,1,1,14,14,15,15,7,7]	2^{-10} 2^{-6}	2^{-19} 2^{-11}	- -
	[0,0,2,2,1,1,9,9,8,8,7,7,15,15,14,14]	2^{-10} 2^{-6}	2^{-19} 2^{-11}	- $\leq 2^{-19}$
	[0,0,8,8,1,1,2,2,9,9,14,14,7,7,15,15]	2^{-10} 2^{-6}	2^{-19} 2^{-11}	- -
	[0,0,9,9,8,8,2,2,1,1,14,14,15,15,7,7]	2^{-10} 2^{-6}	2^{-19} 2^{-11}	- $\leq 2^{-19}$
	[9,9,2,2,1,1,8,8,7,7,15,15,14,14,0,0]	2^{-10} 2^{-6}	2^{-19} 2^{-11}	- -
	[0,0,9,9,15,15,14,14,8,8,7,7,2,2,1,1]	2^{-10} 2^{-6}	2^{-19} 2^{-11}	- -
	[1,1,7,7,8,8,14,14,15,15,2,2,9,9,0,0]	2^{-10} 2^{-6}	2^{-19} 2^{-11}	- -
	[0,0,2,2,1,1,9,9,15,15,14,14,8,8,7,7]	2^{-10} 2^{-6}	2^{-19} 2^{-11}	- -
	[7,7,14,14,15,15,8,8,9,9,1,1,2,2,0,0]	2^{-10} 2^{-6}	2^{-19} 2^{-11}	- -
	[0,0,14,14,15,15,7,7,1,1,2,2,8,8,9,9]	2^{-10} 2^{-6}	2^{-19} 2^{-11}	- -
	[0,0,9,9,8,8,2,2,1,1,7,7,15,15,14,14]	2^{-10} 2^{-6}	2^{-19} 2^{-11}	- -
	[0,0,7,7,1,1,2,2,8,8,9,9,14,14,15,15]	2^{-10} 2^{-6}	2^{-19} 2^{-11}	- -
6	[7,1,14,12]	2^{-6} 2^{-2}	2^{-10} 2^{-4}	2^{-34} 2^{-16}

Appendix F Implementation Benchmarks

This appendix provides the software and hardware measurements used to support the implementation-cost discussion in Section 5. AVR results were measured on ATmega128 using Microchip Studio 7.0 with hand-written AVR assembly. We report both core-only cycles, which count only the S-box computation itself, and core+L/S cycles, which additionally include the load/store overhead for moving inputs and outputs between registers and memory. ARM results were measured on Cortex-M3 using Keil uVision v5.43.1 with hand-written ARM assembly. ASIC results were synthesized from Verilog HDL using Synopsys Design Compiler 2018 with the NanGate 45 nm standard-cell library.

Table F6 AVR benchmark results on ATmega128.

S-box	Rotation sequence	Cycles (C/C+L/S)	Bytes (C/C+L/S)
Alzette (10101010)	[31,24,17,17,0,31,24,16]	77/121	156/196
AFS-64 (11000011)	[1,1,16,24,31,31,8,16]	77/121	156/196
AFS-64 (11000011)	[1,1,16,8,31,31,24,16]	77/121	156/196
AFS-64 (11000011)	[17,24,1,1,16,31,24,0]	78/122	158/198
AFS-64 (11000011)	[17,8,1,1,16,31,8,0]	78/122	158/198
AFS-64 (11000011)	[31,24,17,17,0,31,24,16]	77/121	156/196
AFS-64 (10100011)	[31,24,17,17,0,31,24,16]	77/121	156/196
AFS-64 (11100001)	[31,24,17,17,0,31,24,16]	77/121	156/196
AFS-64 (10000011)	[31,24,17,17,0,31,24,16]	73/117	148/188
AFS-64 (10001011)	[31,24,17,17,0,31,24,16]	77/121	156/196
AFS-64 (10010011)	[31,24,17,17,0,31,24,16]	77/121	156/196
SPECKEY	[7,14,7,14,7,14,7,14]	87/151	176/224
AFS-32 (11000011)	[1,8,14,0,12,7,15,9]	64/86	130/150
AFS-32 (11000011)	[7,14,8,1,13,0,9,15]	60/82	122/142
AFS-32 (10100011)	[0,7,14,15,8,1,2,9]	59/81	120/140
AFS-32 (10100011)	[8,0,2,1,9,14,7,15]	59/81	120/140

Table F7 ARM benchmark results on Cortex-M3.

S-box	Rotation sequence	ARM cycles	ARM bytes
Alzette (10101010)	[31,24,17,17,0,31,24,16]	12	48
AFS-64 (11000011)	[1,1,16,24,31,31,8,16]	12	48
AFS-64 (11000011)	[1,1,16,8,31,31,24,16]	12	48
AFS-64 (11000011)	[17,24,1,1,16,31,24,0]	12	48
AFS-64 (11000011)	[17,8,1,1,16,31,8,0]	12	48
AFS-64 (11000011)	[31,24,17,17,0,31,24,16]	12	48
AFS-64 (10100011)	[31,24,17,17,0,31,24,16]	12	48
AFS-64 (11100001)	[31,24,17,17,0,31,24,16]	12	48
AFS-64 (10000011)	[31,24,17,17,0,31,24,16]	11	44
AFS-64 (10001011)	[31,24,17,17,0,31,24,16]	12	46
AFS-64 (10010011)	[31,24,17,17,0,31,24,16]	12	48
SPECKEY	[7,14,7,14,7,14,7,14]	40	136
AFS-32 (11000011)	[1,8,14,0,12,7,15,9]	36	112
AFS-32 (11000011)	[7,14,8,1,13,0,9,15]	36	112
AFS-32 (10100011)	[0,7,14,15,8,1,2,9]	36	112
AFS-32 (10100011)	[8,0,2,1,9,14,7,15]	36	112

Table F8 ASIC synthesis results.

S-box	Rotation sequence	ASIC delay (ns)	ASIC area (μm^2)
Alzette (10101010)	[31,24,17,17,0,31,24,16]	1.95	1426.55
AFS-64 (11000011)	[1,1,16,24,31,31,8,16]	1.95	1310.00
AFS-64 (11000011)	[1,1,16,8,31,31,24,16]	1.95	1310.00
AFS-64 (11000011)	[17,24,1,1,16,31,24,0]	1.93	1351.01
AFS-64 (11000011)	[17,8,1,1,16,31,8,0]	1.93	1351.01
AFS-64 (11000011)	[31,24,17,17,0,31,24,16]	1.98	1368.03
AFS-64 (10100011)	[31,24,17,17,0,31,24,16]	1.99	1351.81
AFS-64 (11100001)	[31,24,17,17,0,31,24,16]	1.99	1354.47
AFS-64 (10000011)	[31,24,17,17,0,31,24,16]	1.58	1157.89
AFS-64 (10001011)	[31,24,17,17,0,31,24,16]	1.95	1356.86
AFS-64 (10010011)	[31,24,17,17,0,31,24,16]	1.98	1413.78
SPECKEY	[7,14,7,14,7,14,7,14]	1.86	830.71
AFS-32 (11000011)	[1,8,14,0,12,7,15,9]	1.61	643.45
AFS-32 (11000011)	[7,14,8,1,13,0,9,15]	1.52	680.42
AFS-32 (10100011)	[0,7,14,15,8,1,2,9]	1.64	659.14
AFS-32 (10100011)	[8,0,2,1,9,14,7,15]	1.55	641.32

Appendix G Detailed Configurational Encoding Algorithm

For completeness, we give the detailed MILP-generation procedure used for the configurational encoding of Alzette.

Algorithm 2 Differential properties of Alzette using MILP and configurational encodings

Input: $r = t$, $N = 4$, $A_{2N} = \{10\}^N$, $K_{2N} = [31, 24, 17, 17, 0, 31, 24, 16]$
▷ r: number of rounds; N: number of structural rounds; A_{2N}: configurational sequence; K_{2N}: rotational shift sequence
Output: M'
▷ M': generated MILP model for r-round differential analysis
Initialize $M \leftarrow \emptyset$ ▷ empty MILP model
for $i \leftarrow 0$ **to** $2t - 1$ **do**
    if $A_{2N}[i \bmod 2N] = 0$ **then**
        $M \leftarrow M_0(i)$ ▷ XOR-based round modeling
    else
        $M \leftarrow M_1(i)$ ▷ modular-addition-based round modeling
    end if
end for
Set objective function in M ▷ minimize weight p of the differential characteristic ($\Pr = 2^{-p}$)
    return M

procedure $M_0(i)$
▷ Function: XOR round constraints
    Initialize local model $M_0 \leftarrow \emptyset$
    Add constraint $X_i^R = X_{i+1}^L$ to M_0 ▷ Feistel structure
    Add constraint $X_i^R = X_i^R \ggg K_{2N}[i \bmod 2N]$ to M_0 ▷ rotational shift
    for $j \leftarrow 0$ **to** 31 **do**
        Add XOR difference constraint for bit j to M_0
    end forreturn M_0
end procedure

procedure $M_1(i)$
▷ Function: modular addition round constraints
    Initialize local model $M_1 \leftarrow \emptyset$
    Add constraint $X_i^R = X_{i+1}^L$ to M_1 ▷ Feistel structure
    Add constraint $X_i^R = X_i^R \ggg K_{2N}[i \bmod 2N]$ to M_1 ▷ rotational shift
    Add MSB carry constraint for modular addition to M_1
    for $j \leftarrow 0$ **to** 30 **do**
        Add differential constraint for modular addition at bit j to M_1
    end forreturn M_1
end procedure

Acknowledgements This work was partially supported by National Cyber Security-National Science and Technology Major Project (2026ZD1500600, 2026ZD1500604), by the National Natural Science Foundation of China (62402132), and by the Guangxi Natural Science Foundation (2026GXNSFBA00640189). Enes Pasalic and René Rodríguez-Aldama are partially supported by the Slovenian Research and Innovation Agency (ARIS) through the project J1-60012 and the program P1-0404. We thank Baptiste Lambin for helpful discussions.

Declarations

Conflict of interest The authors declare no conflict of interest.

References

[1] Beierle, C., Biryukov, A., Cardoso dos Santos, L., Großschädl, J., Perrin, L., Udovenko, A., Velichkov, V., Wang, Q.: Alzette: a 64-bit ARX-box(feat. CRAX and TRAX). In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology – CRYPTO 2020. LNCS, vol. 12172, pp. 419–448. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56877-1_15

[2] Beierle, C., Biryukov, A., Cardoso dos Santos, L., Großschädl, J., Perrin, L., Udovenko, A., Velichkov, V., Wang, Q.: Lightweight AEAD and hashing using the Sparkle permutation family. IACR Trans. Symmetric Cryptol. **2020**(S1), 208–261 (2020). https://doi.org/10.13154/tosc.v2020.iS1.208-261

[3] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Keccak. In: Johansson, T., Nguyen, P. (eds.) Advances in Cryptology – EUROCRYPT 2013. LNCS, vol. 7881, pp. 313–314. Springer, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_19

[4] Beyne, T., Verbauwhede, M.: Integral cryptanalysis using algebraic transition matrices. IACR Trans. Symmetric Cryptol. **2023**(S4), 244–269 (2023). https://doi.org/10.46586/tosc.v2023.i4.244-269

[5] Biryukov, A., Lambin, B., Udovenko, A.: Exact formula for RX-differential probability through modular addition for all rotations. IACR Trans. Symmetric Cryptol. **2025**(1), 542–591 (2025). https://doi.org/10.46586/tosc.v2025.i1.542-591

[6] Boura, C., Canteaut, A.: Another view of the division property. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology – CRYPTO 2016. LNCS, vol. 9814, pp. 654–682. Springer, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_24

[7] Boura, C., Coggia, D.: Efficient MILP modelings for sboxes and linear layers of SPN ciphers. IACR Trans. Symmetric Cryptol. **2020**(3), 327–361 (2020). https://doi.org/10.46586/tosc.v2020.i3.327-361

[8] Breaken, A., Semaev, I.: The ANF of the composition of addition and multiplication mod 2^n with a Boolean function. In: Gilbert, H., Handschuh, H. (eds.) Fast Software Encryption – FSE 2005. LNCS, vol. 3557, pp. 112–125. Springer, Berlin, Heidelberg (2005). https://doi.org/10.1007/11502760_8

[9] CACR: National cryptographic algorithm design competition. Available at https://sfjs.cacrnet.org.cn/ (2018)

[10] Canteaut, A., Duval, S., Leurent, G., Naya-Plasencia, M., Perrin, L., Pornin, T., Schrottenloher, A.: Saturnin: a suite of lightweight symmetric algorithms for post-quantum security. IACR Trans. Symmetric Cryptol. **2020**(S1), 160–207 (2020). https://doi.org/10.13154/tosc.v2020.iS1.160-207

[11] Conforti, M., Cornuéjols, G., Zambelli, G.: Integer programming models. In: Integer Programming. Graduate Texts in Mathematics, vol. 271, pp. 45–84. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11008-0_2

[12] Cui, T., Wang, M., Fan, Y., Hu, K., Huang, L.: Ballet: a software-friendly block cipher. Journal of Cryptologic Research. **2019**(6), 704–712 (2019). https://doi.org/10.13868/j.cnki.jcr.000335

[13] Daemen, J., Rijmen, V.: The design of Rijndael - the Advanced Encryption Standard (AES). Information Security and Cryptography, Springer, Heidelberg (2020). https://doi.org/10.1007/978-3-662-60769-5

[14] Dinu, D., Perrin, L., Udovenko, A., Velichkov, V., Großschädl, J., Biryukov, A.: Design strategies for ARX with provable bounds: SPARX and LAX. In: Cheon, J., Takagi, T. (eds.) Advances in Cryptology – ASIACRYPT 2016. LNCS, vol. 10031, pp. 484–513. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_18

[15] Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon v1.2: lightweight authenticated encryption and hashing. Journal of Cryptology. **34**(3), 33 (2021). https://doi.org/10.1007/s00145-021-09398-9

[16] Erlacher, J., Mendel, F., Eichlseder, M.: Bounds for the security of Ascon against differential and linear cryptanalysis. IACR Trans. Symmetric Cryptol. **2022**(1), 64–87 (2022). https://doi.org/10.46586/tosc.v2022.i1.64-87

[17] Fan, T., Feng, W., Wei, Y.: The design and cryptanalysis of a large state lightweight cryptographic S-box. Journal of Xidian University. **50**(4), 170–179 (2023). https://doi.org/10.19665/j.issn1001-2400.2023.04.017

[18] Fu, K., Wang, M., Guo, Y., Sun, S., Hu, L.: MILP-based automatic search algorithms for differential and linear trails for SPECK. In: Peyrin, T. (ed.) Fast Software Encryption – FSE 2016. LNCS, vol. 9783, pp. 268–288. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-52993-5_14

[19] Gong, X., Wang, Q., Hao, Y., Jiao, L., Hu, X.: Persistence of hourglass(-like) structure: improved differential-linear distinguishers for several ARX ciphers. In: Advances in Cryptology – ASIACRYPT 2025. LNCS, vol. 16245, pp. 317–351. Springer, Singapore (2025). https://doi.org/10.1007/978-981-95-5018-0_11

[20] GUROBI: Gurobi optimizer 11.0. Available at http://www.gurobi.cn/ (2023)

[21] Hao, Y., Leander, G., Meier, W., Todo, Y., Wang, Q.: Modeling for three-subset division property without unknown subset. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology – EUROCRYPT 2020. LNCS, vol. 12105, pp. 466–495. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_17

[22] Hebborn, P., Lambin, B., Leander, G., Todo, Y.: Lower Bounds on the Degree of Block Ciphers. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2020. LNCS, vol. 12491, pp. 537–566. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64837-4_18

[23] Huang, M., Xu, Z., Wang, L.: On the probability and automatic search of Rotational-XOR cryptanalysis on ARX ciphers. The Computer Journal. **65**(12), 3062–3080 (2022). https://doi.org/10.1093/comjnl/bxab126

[24] Hu, K., Sun, S., Wang, M., Wang, Q.: An algebraic formulation of the division property: revisitiing degree evaluations, cube attacks, and key-independent sums. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2020. LNCS, vol. 12491, pp. 446–476. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64837-4_15

[25] Hu, K., Yap, T.: Perfect monomial prediction for modular addition. IACR Trans. Symmetric Cryptol. **2024**(3), 177–199 (2024). https://doi.org/10.46586/tosc.v2024.i3.177-199

[26] Lipmaa, H., Moriai, S.: Efficient algorithms for computing differential properties of addition. In: Matsui, M. (ed.) Fast Software Encryption – FSE 2001. LNCS, vol. 2355, pp. 336–350. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45473-X_28

[27] Lv, G., Jin, C., Cui, T.: A MIQCP-based automatic search algorithm for differential-linear trails of ARX ciphers. Cryptology ePrint Archive, Paper 2023/259 (2023). https://eprint.iacr.org/2023/259

[28] Liu, Y., Niu, Z., Sun, S., Li, C., Hu, L.: Rotational differential-linear cryptanalysis revisited. Journal of Cryptology. **36**(3), 1–45 (2023). https://doi.org/10.1007/s00145-022-09440-4

[29] Liu, Y., Sun, S., Li, C.: Rotational cryptanalysis from a differential-linear perspective: practical distinguishers for round-reduced FRIET, Xoodoo, and Alzette. In: Canteaut, A., Standaert, FX. (eds.) Advances in Cryptology –

EUROCRYPT 2021. LNCS, vol. 12696, pp. 741–770. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77870-5_26

[30] Morawiecki, P., Pieprzyk, J., Srebrny, M.: Rotational cryptanalysis of round-reduced Keccak. In: Moriai, S. (ed.) Fast Software Encryption – FSE 2013. LNCS, vol. 8424, pp. 241–262. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43933-3_13

[31] Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: Wu, CK., Yung, M., Lin, D. (eds) Information Security and Cryptology. Inscrypt 2011. Lecture Notes in Computer Science, vol 7537, (2012). https://doi.org/10.1007/978-3-642-34704-7_5

[32] NIST: Lightweight authenticated encryption project. Available at https://csrc.nist.gov/projects/lightweight-cryptography (2018)

[33] Niu, Z., Sun, S., Liu, Y., Li, C.: Rotational differential-linear distinguishers of ARX ciphers with arbitrary output linear masks. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology – CRYPTO 2022. LNCS, vol. 13507, pp. 3–32. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15802-5_1

[34] Nyberg, K., Wallén, J.: Improved linear distinguishers for SNOW 2.0. In: Robshaw, M. (eds.) Fast Software Encryption – FSE 2006. LNCS, vol. 4047, pp. 144–162. Springer, Berlin, Heidelberg (2006). https://doi.org/10.1007/11799313_10

[35] Schulte-Geers, E.: On CCZ-equivalence of addition mod 2^n. Designs, Codes and Cryptography. **66**, 111–127 (2013). https://doi.org/10.1007/s10623-012-9668-4

[36] Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T.: Piccolo: an ultra-lightweight blockcipher. In: Preneel, B., Takagi, T. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2011. LNCS, vol. 6917, pp. 342–357. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23951-9_23

[37] Stump, A., Barrett, C.W., Dill, D.L.: CVC: a cooperating validity checker. In: Brinksma, E., Larsen, K. (eds.) Computer Aided Verification. LNCS, vol. 2404, pp. 500–504. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45657-0_40

[38] Sun, S., Hu, L., Song, L., Xie, Y., Wang, P.: Automatic security evaluation of block ciphers with S-bP Structures against related-key differential attacks. In: Lin, D., Xu, S., Yung, M. (eds.) Information Security and Cryptology. Inscrypt 2013. LNCS, vol 8567, (2014). https://doi.org/10.1007/978-3-319-12087-4_3

[39] Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (related-key) differential characteristic search: application on SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In: Sarkar, P.,

Iwata, T. (eds.) Advances in Cryptology – ASIACRYPT 2014. LNCS, vol. 8873, pp. 158–178. Springer, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-45611-8_9

[40] Sun, L., Wang, M.: Sok: modeling for large s-boxes oriented to differential probabilities and linear correlations. IACR Trans. Symmetric Cryptol. **2023**(1), 111–151 (2023). https://doi.org/10.46586/tosc.v2023.i1.111-151

[41] Sun, L., Wang, W., Wang, M.: Automatic search of bit-based division property for ARX ciphers and word-based division property. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology – ASIACRYPT 2017. LNCS, vol. 10624, pp. 128–157. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_5

[42] Tian, T., Qi, W., Ye, C., Xie, X.: SPRING: A family of small hardware-oriented block ciphers based on NFSRs. Journal of Cryptologic Research. **6**(6), 815–834 (2019). https://doi.org/10.13868/j.cnki.jcr.000343

[43] Todo, Y.: Structural evaluation by generalized integral property. In: Oswald, E., Fischlim, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 287–314. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_12

[44] Todo, Y., Morii, M.: Bit-based division property and application to simon family. In: Peyrin, T. (eds.) Fast Software Encryption. FSE2016. LNCS, vol. 9783. Springer, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-52993-5_18

[45] Wallén, J.: Linear approximations of addition modulo 2^n. In: Johansson, T. (ed.) Fast Software Encryption – FSE 2003. LNCS, vol. 2887, pp. 261–273. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39887-5_20

[46] Wang, S., Hu, B., Guan, J., Zhang, K., Shi, T.: Milp-aided method of searching division property using three subsets and applications. In: Galbraith, S., Moriai, S. (eds.) Advances in Cryptology – ASIACRYPT 2019. LNCS, vol. 11923, pp. 398–427. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34618-8_14

[47] Xiang, Z., Zhang, W., Bao, Z., Lin, D.: Applying MILP methods to searching integral distinguishers based on division property for 6 lightweight block ciphers. In: Cheon, J., Takagi, T. (eds.) Advances in Cryptology – ASIACRYPT 2016. LNCS, vol. 10031, pp. 648–678. Springer, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_24

[48] Xu, H., Duan, M., Tan, L., Qi, W., Wang, Z.: On the NBC algorithm. Journal of Cryptologic Research. **6**(6), 760–767 (2019). https://doi.org/10.13868/j.cnki.jcr.000339