IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2024, No. 3, pp. 200–237. DOI:10.46586/tosc.v2024.i3.200-237

Reconstructing S-Boxes from Cryptographic Tables with Milp

Raghvendra Rohit[1] and Sumanta Sarkar[2]

[1] Cryptography Research Centre, Technology Innovation Institute, Abu Dhabi, UAE
raghvendra.rohit@tii.ae
[2] University of Warwick, Coventry, United Kingdom
sumanta.sarkar@warwick.ac.uk

Abstract. Reconstructing an S-box from a cryptographic table such as difference distribution table (DDT), linear approximation table (LAT), differential-linear connectivity table (DLCT) or boomerang connectivity table (BCT) is one of the fundamental problems in symmetric-key cryptography. Till now, there are only very few known methods which can reconstruct an S-box from a given table: guess-and-determine algorithms of Boura et al. (DCC 2019) and Tian et al. (DCC 2020), sign determination algorithm of Dunkelman et al. (ToSC 2019) and STP based approach of Lu et al. (DCC 2022). In this paper we consider the reconstruction problem in an even more challenging setup where one needs to reconstruct S-boxes from a partial cryptographic table. We are able to reconstruct S-boxes when only a few number of rows of a cryptographic table is given. This problem has never been studied in the literature. We apply mixed integer linear programming (MILP) as the key tool for solving this problem. Needless to say that we can solve the reconstruction problem when the full table is given and this is the first ever application of MILP tool in solving such fundamental problems. As a further application of our method, we provide the generic MILP models which can search for S-boxes with a given cryptographic property such as differential uniformity, linearity, differential-linear uniformity or boomerang uniformity. Additionally, our method can recover a Boolean function from a given Walsh spectrum or a Boolean function with a given nonlinearity. We also introduce a new heuristic called Optimistic MILP objective that guides the model towards obtaining multiple S-boxes or Boolean functions with the same cryptographic property. We give detailed experimental results for up to 6-bit S-boxes showing the effectiveness of our technique.

Keywords: Substitution box · Difference Distribution Table (DDT) · Linear Approximation Table (LAT) · Differential-Linear Connectivity Table (DLCT) · Boomerang Connectivity Table (BCT) · Mixed Integer Linear Programming (MILP)

1 Introduction

The substitution box, commonly known as S-box, is one of the key components of a symmetric cipher. Typically, an S-box is a nonlinear function that takes n-bits as input and outputs m-bits. For majority of symmetric ciphers, we have n equals m and an S-box is a permutation.

S-box plays a crucial role in providing security to a symmetric cipher and from a designer's perspective, an S-box should have some strong cryptographic properties. Some of these properties, namely differential uniformity (1), linearity (3), differential-linear uniformity (5) and boomerang uniformity (9) are well investigated in literature. These properties help in measuring the resilience of a cipher against the so-called differential [BS91], linear cryptanalysis [Mat94], differential-linear attacks [LH94], and boomerang

Received: 2024-06-01 Accepted: 2024-08-01 Published: 2024-09-06

attacks [Wag99], respectively. Lower the values of these metrics, higher the cipher's resistance against these attacks.

The aforementioned cryptographic properties can simply be derived from the description of an S-box. There exist numerous tools for the same [Rus, mag, FJ, BGLS19]. However, finding S-boxes with a given property is a challenging task as search space of S-boxes grows exponentially with the dimensions. For instance, the so-called Big APN problem, which is to find an APN permutation on even dimensions, has been open for many years having the only success in dimension 6 [BDMW10]. The available tools for generating S-boxes with given cryptographic properties only work for some limited dimensions. For instance, the libapn by Flori et al. [FJ], a C++ library, can only search for APN permutations up to dimension 5. Another tool PEIGEN by Bao et al. [BGLS19], also implemented in C++, can generate S-boxes up to dimension 4. For $n \geq 5$, this tool often becomes impractical.

Another method to derive the cryptographic properties of an S-box is by forming a table. For example, the differential uniformity of an S-box is obtained by first computing the difference distribution table (Def. 1) and then taking the maximum entry (except the first entry) in that table as the differential uniformity. In the same way, we derive the linearity, differential-linear uniformity and boomerang uniformity of an S-box from their respective cryptographic tables, namely linear approximation table (Def. 2), differential-linear connectivity table (Def. 3) and boomerang connectivity table (Def. 5). While it is easy to map an S-box to any of its cryptographic tables, however, it is not obvious to solve the converse problem:

"*Given a cryptographic table* T*, where* T *is either of* DDT*,* LAT*,* DLCT *and* BCT*, how to reconstruct an S-box that maps to* T *?*"

This is a fundamental problem in symmetric key cryptography which has received very little attention. Solution to this problem has a far-reaching impact, though. For instance, a designer could fill these tables in a way so that the interaction between S-boxes and mixing layer becomes strong, and then from the tables, she can derive the description of the S-box. This will also help in getting new theoretical insights on the properties of these tables [BCJS19]. Moreover, cryptanalytic attacks on ciphers based on secret S-boxes can benefit by learning these tables and then finding the corresponding secret S-box. A concrete example of the latter is Bar-On et al.'s slide attack on GOST [BBDK18] where the attacker can learn the DDT and aims to recover the underlying secret S-box. While this problem of reconstructing S-boxes from cryptographic tables such as DDT, LAT, DLCT and BCT is known, there is another relevant question which has never been investigated in the literature.

Problem P1. "*Given an* $l \times 2^n$*-dimensional table* T*, where* T *contains* l *rows of either of full* $2^n \times 2^n$*-dimensional* DDT*,* LAT*,* DLCT *and* BCT*, how to reconstruct an S-box that maps to this partial table* T *?*"

Here partial table refers to fraction of a full table of an S-box. For example, suppose S is an S-box and D_S is its DDT. We take some l-rows of D_S, which we call a partial DDT. Then the above question states if it is possible to reconstruct an S-box from this partial DDT that is similar to S. It is equally interesting if any S-box could be reconstructed from an arbitrary partial table.

In [BCJS19], Boura et al. proposed a guess-and-determine algorithm which takes as input a DDT and returns all n-bit to n-bit functions whose difference distribution table has the same support as the given DDT. Later, Dunkelman and Huang [DH19] solved the problem of reconstructing an S-box from DDT by introducing the sign determination problem based on the well known relations between DDT and squared LAT [CV94, DGV94, BN13] and then recovering S-box from the latter one. Lu et al. [LMC+22] modeled the

relationship between an S-box and its DDT and LAT in terms of a satisfiability modulo theories (SMT) problem and applied an SMT solver STP (Simple Theorem Prover) to solve the model and get the related S-boxes. Tian et al. [TBP20] gave a solution that could reconstruct S-boxes from a given BCT.

The above solutions are elegant on their own; however, they have certain limitations:

- The works of [BCJS19, TBP20] are not capable of solving Problem P1 due to their tree-based search approach. More precisely, given the depth d (number of rows of a table), their algorithm can only find d values of the S-box. So, given d rows, the remaining $2^n - d$ values of S-box have to be searched exhaustively while ensuring that entries of given rows are not affected. Moreover, their approach is not applicable for finding an S-box or Boolean function with a given cryptographic property.[1] Similarly, [DH19] can not also solve Problem P1 and find S-box/Boolean function with a given property.

- The approach in [LMC+22] for DDT and LAT works for up to 5-bit S-boxes and considers only differential uniformity, linearity, fixed points and BIBO properties. They never considered beyond 5-bit S-boxes. Additionally, they did not account for the reconstruction of the S-boxes from full DLCT and BCT, nor did they address Problem P1.

There has been a similar research direction on partial tables taken up by Biryukov et al. [BV14], though. It is well known that for practical ARX ciphers it is infeasible to compute the full DDT, so they considered a partial DDT – computing differentials that have probability above a fixed threshold. This is possible as the probabilities of XOR (respectively modular addition) differentials through the modular addition (respectively XOR) operation are monotonously decreasing with the bit size of the word. Their effort is focused to recover the key utilising these partial differentials, and it is not clear how such a tool could be employed to solve the reconstruction problem of S-boxes from a partial DDT, LAT, DLCT or BCT.

1.1 Our Contributions

When we look back at the existing tools that have been used to solve the S-box reconstruction problem, we find that applicability of Mixed Integer Linear Programming (MILP) technique has been overlooked despite the fact that MILP has shown a significant impact on cryptanalysis. In this work, we give a new perspective and an unified approach to the S-box reconstruction problem based on MILP. We present the novel applications of Mixed Integer Linear Programming and Mixed Integer Quadratic Constraint Programming (MIQCP) to solve Problem P1 and to find S-boxes with a given cryptographic property such as differential uniformity, linearity, differential-linear uniformity or boomerang uniformity. We now describe our contributions in detail.

1. **Modeling an S-box with permutation matrix.** For the reconstruction of S-boxes from a cryptographic table using MILP technique, one first needs to model the S-boxes. In this regard, we represent an n-bit S-box as an array of 2^n integer variables. To model this array as a permutation of $[0, \ldots, 2^n - 1]$ in an MILP setting, a naive approach then would be to choose constraints defining the pairwise inequalities of these integer variables. However, this way one needs to introduce $\binom{2^n}{2}$ constraints. In fact, this has been done in [LMC+22] in their STP modeling. We work out a much better and efficient approach – we utilize the well-known permutation matrices of order $2^n \times 2^n$. We call it *(target) S-box* modeling.

[1] Confirmed with one of the authors of [BCJS19, TBP20]

2. **Solving Problem P1:** Having modeled the n-bit S-box, we solve the reconstruction problem from partial tables. We assume that $l \leq 2^n$ rows are given.

   (i) Partial DDT to S-box. In our MILP modeling, we first enumerate solution pairs for a given input difference and model XOR of their S-box output values. We then map solutions with entries of the *partial* DDT. When we are given a full DDT, we need to consider each entries from the table. In case, a partial table is given, available entries from the partial DDT is considered. Finally, we check the feasibility of the entire model. An infeasible model means the given (partial) DDT is not valid, i.e., it does not correspond to any S-box. On the other hand, a feasible model will output an S-box. When a full DDT is given, MILP outputs the corresponding S-box. For partial DDT, the output S-box's full DDT includes partial DDT, however, depending on the number of rows in the partial DDT, the function varies and may differ in their differential uniformities.

   (ii) Partial LAT to S-box. In LAT, each pair of input and output masks defines a linear equation. Hence we first model these linear equations which involves modeling the scalar products and XOR operations. We then model the count of the occurrences when the linear relation holds followed by mapping the respective counts to entries in the LAT. As done in case of DDT, we can reconstruct S-boxes from partial or full LAT.

   (iii) Partial DLCT to S-box. We combine *partial* DDT and *partial* LAT approach to solve this problem. In particular, we enumerate solutions for a given input difference, model XOR of their S-box output values followed by modeling the scalar product of output mask with output differences and then map it to partial DLCT or full DLCT.

   Furthermore, as differential-linear connectivity and autocorrelation tables are connected, our model can also recover an S-box from the autocorrelation table.

   (iv) Partial BCT to S-box. BCT involves nonlinear constraints and this makes the modeling even more challenging, so, it is completely different from modeling of DDT or LAT. Due to the complex nature of boomerang condition (8), we use MIQCP within MILP to solve this problem. First, we show the modeling of (8) in terms of quadratic constraints. Next, we map the number of solutions for the pairs of input-output difference to entries of partial BCT. Similar to DDT, MILP outputs the S-box corresponding to the partial BCT, and when full BCT is given, it outputs the corresponding S-box.

3. **Finding S-boxes and Boolean functions with a given cryptographic property.** We show how to tweak our MILP models of DDT, LAT, DLCT and BCT to solve this problem. Notably, we present the first novel and generic MILP-based algorithm to search for S-boxes with fixed differential uniformity, linearity, differential-linear uniformity and boomerang uniformity. We present how the same approach could be extended to reconstruct S-boxes from a given Walsh spectrum and consequently find Boolean function with a given nonlinearity.

4. **Optimistic MILP objective and CCZ-inequivalent S-boxes.** We introduce *Optimistic MILP objective* as a new heuristic to search for multiple S-boxes while solving Problem P1. Our heuristic starts with a given S-box, adds a random objective function in the model and then search for other S-boxes. We demonstrate its application to the 6-bit inverse S-box. Consequently, within a matter of seconds,

we find four S-boxes which are CCZ-inequivalent to the inverse S-box, and each of the S-boxes has differential uniformity 4.

We validate all our proposed models with the cryptographic tables (both full and partial) of several S-boxes. In particular, we consider the inverse S-box ($x \mapsto x^{-1}$) in dimensions 3, 4 and 5, PRESENT, GIFT, SKINNY, ASCON and Keccak S-boxes. Moreover, we run our models with several given values of differential uniformity, linearity, differential-linear uniformity and boomerang uniformity. We are able to successfully recover the respective S-boxes. We evaluate the performance of our models with Gurobi and OR-Tools in Tables 4 to 11, and also compare with [LMC+22] in Table 14.

Our codes are available at https://github.com/sumantasarkar/supplementary_material_tosc_2024_v3.

1.2 Outline of the Paper

The rest of the paper is organized as follows. In Section 2, we first recall several cryptographic tables associated with S-boxes and then discuss equivalence properties of S-boxes and briefly explain the idea of MILP. In Section 3, we discuss the modeling of an S-box using MILP. Sections 4, 5, 6 and 7 present the detailed and step-by-step MILP-based approach of reconstructing an S-box from DDT, LAT, DLCT and BCT along with experimental results, respectively. In Section 8, we first give MILP models to search for S-boxes with a given cryptographic property. We then introduce the Optimistic MILP objective heuristic and give experimental results. We also discuss the reconstruction of a Boolean function from a Walsh spectrum and find Boolean function with a given nonlinearity. In Section 9, we discuss the advantages and limitations of our method, and provide comparison with the existing works. Finally, we conclude in Section 10 with future research directions.

2 Preliminaries

In this section, we recall different cryptographic tables of an S-box and their role in the security analysis of the cipher. We also discuss various equivalence relations for S-boxes and briefly explain the mixed integer linear programming (MILP) method. We first begin with the mathematical notations that will be used throughout the paper.

2.1 Notation

Let $\mathbb{F}_{2^n}$ denote the finite field of characteristic 2 of order 2^n and $\mathbb{F}_2^n$ be the vector space with 2^n binary n-tuples. An n-bit S-box is a mapping from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$. When S-boxes are used in symmetric ciphers it is considered that they are permutations. For $u, v \in \mathbb{F}_2^n$, we denote its scalar product by $u \cdot v$. We sometimes use '$\cdot$' to denote the integer product or matrix multiplication if the meaning is clear from the context.

Let $\mathsf{P} = [p_{i,j}]$ be a $2^n \times 2^n$ binary matrix, where $p_{i,j} \in \{0, 1\}$. We say P is a permutation matrix if every row and every column of it contains only single 1 and 0's at all other places.

2.2 Cryptographic Tables

We now describe the cryptographic tables such as Difference Distribution Table (DDT), Linear Approximation Table (LAT), Differential-linear Connectivity Table (DLCT), Autocorrelation Table (ACT) and Boomerang Connectivity Table (BCT) of the S-box $S : \mathbb{F}_2^n \to \mathbb{F}_2^n$.

Definition 1 (Difference Distribution Table (DDT))**.** For input difference $\alpha \in \mathbb{F}_2^n$ and the output difference $\beta \in \mathbb{F}_2^n$, we define the differential of S as $S(x) \oplus S(x \oplus \alpha) = \beta$. The

difference distribution table is a $2^n \times 2^n$ array where the (α, β)-th element is

$$\mathsf{DDT}_S(\alpha, \beta) = \#\{x \in \mathbb{F}_2^n : S(x) \oplus S(x \oplus \alpha) = \beta\}.$$

The differential uniformity of the S-box S is defined as

$$\mathsf{DU}_S = \max_{\alpha \neq 0} \mathsf{DDT}_S(\alpha, \beta). \tag{1}$$

A δ-differential uniform S-box is the one that has differential uniformity equals to δ. Lower the differential uniformity better the resistance against differential attack (Biham et al. [BS91]). The best differential uniformity that an S-box can have is 2, and such an S-box is called *almost perfect nonlinear (APN)*.

Definition 2 (Linear Approximation Table (LAT))**.** For input mask $\lambda \in \mathbb{F}_2^n$ and output mask $\gamma \in \mathbb{F}_2^n$, the linear approximation of S is given by the relation $\lambda \cdot x = \gamma \cdot S(x)$. The linear approximation table of S is a $2^n \times 2^n$ array where the (λ, γ)-th element is

$$\mathsf{LAT}_S(\lambda, \gamma) = \#\{x \in \mathbb{F}_2^n : \lambda \cdot x = \gamma \cdot S(x)\} - 2^{n-1}.$$

Dividing $\mathsf{LAT}_S(\lambda, \gamma)$ by 2^n, we get the probability bias of the linear approximation for the masks (λ, γ).

LAT is utilized to launch the linear approximation attack (Matsui [Mat94]), essentially the attacker looks for the pair (λ, γ) with higher biases.

The *correlation* of S-box with input mask λ and output mask γ is defined as

$$C_S(\lambda, \gamma) = \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} (-1)^{\gamma \cdot S(x) \oplus \lambda \cdot x}. \tag{2}$$

We also have *linearity* of S which is defined as

$$\mathcal{L}_S = 2^n \max_{\lambda \in \mathbb{F}_2^n, \gamma \in \mathbb{F}_2^n, \gamma \neq 0} |C_S(\lambda, \gamma)|. \tag{3}$$

It is easy to derive that

$$\begin{cases} \mathsf{LAT}_S(\lambda, \gamma) + 2^{n-1} & \leq (1 + \frac{\mathcal{L}_S}{2^n}) \cdot 2^{n-1}, \quad \text{if } C_S(\lambda, \gamma) \geq 0 \\ \mathsf{LAT}_S(\lambda, \gamma) + 2^{n-1} & \geq (1 - \frac{\mathcal{L}_S}{2^n}) \cdot 2^{n-1}, \quad \text{o.w.} \end{cases} \tag{4}$$

From the designer's perspective, S-box should have low linearity, that is low absolute correlation values.

Differential-Linear (DL) attack was introduced by Langford et al. [LH94], based on this attack Bar-On et al. [BDKW19] presented Differential-Linear Connectivity Table (DLCT) for an S-box.

Definition 3 (Differential-Linear Connectivity Table (DLCT))**.** For input difference $\alpha \in \mathbb{F}_2^n$ and mask $\lambda \in \mathbb{F}_2^n$, the differential-linear approximation of S is given by the relation $\lambda \cdot S(x) = \lambda \cdot S(x \oplus \alpha)$. The differential-linear connectivity table of S is a $2^n \times 2^n$ array where the (α, λ)-th element is

$$\mathsf{DLCT}_S(\alpha, \lambda) = \#\{x \in \mathbb{F}_2^n : \lambda \cdot S(x) = \lambda \cdot S(x \oplus \alpha)\} - 2^{n-1}.$$

The *differential-linear uniformity* of S is defined as

$$\mathsf{DLU}_S = \max_{\alpha \in \mathbb{F}_2^n, \lambda \in \mathbb{F}_2^n, \alpha \neq 0, \lambda \neq 0} |\mathsf{DLCT}_S(\alpha, \lambda)|. \tag{5}$$

A d-differential-linear uniform S-box is the one that has differential-linear uniformity equals to d.

Definition 4 (Autocorrelation Table (ACT))**.** For $a \in \mathbb{F}_2^n$ and $b \in \mathbb{F}_2^n$, the autocorrelation of S is defined as

$$\mathsf{ACT}_S(a,b) = \sum_{x \in \mathbb{F}_2^n} (-1)^{b \cdot (S(x) \oplus S(x \oplus a))}.$$

The autocorrelation table is a $2^n \times 2^n$ array where the (a,b)-th element is $\mathsf{ACT}_S(a,b)$.

In [CKL+19], Canteaut et al. provided the link between DLCT and ACT which is given by

$$\mathsf{DLCT}(a,b) \quad = \quad \frac{1}{2}\mathsf{ACT}(a,b). \tag{6}$$

Definition 5 (Boomerang Connectivity Table (BCT))**.** For $u \in \mathbb{F}_2^n$ and $v \in \mathbb{F}_2^n$, the boomerang of S is given by the relation

$$S^{-1}(S(x) \oplus v) \oplus S^{-1}(S(x \oplus u) \oplus v) = u. \tag{7}$$

The boomerang connectivity table of S is a $2^n \times 2^n$ array where the (u,v)-th element is

$$\mathsf{BCT}_S(u,v) = \#\{x \in \mathbb{F}_2^n : S^{-1}(S(x) \oplus v) \oplus S^{-1}(S(x \oplus u) \oplus v) = u\}.$$

The boomerang can be equivalently defined for the pair (u,v) by the two relations

$$S(x) \oplus S(y) = v \quad \text{and} \quad S(x \oplus u) \oplus S(y \oplus u) = v, \tag{8}$$

and accordingly BCT is given by

$$\mathsf{BCT}_S(u,v) = \#\{(x,y) \in \mathbb{F}_2^n \times \mathbb{F}_2^n \text{ such that (8) holds}\}.$$

BCT was introduced by Cid et al. [CHP+18] that gave better insights to Boomerang attack (Wagner [Wag99]). *Boomerang uniformity* of S is defined as

$$\mathsf{BU}_S = \max_{u \in \mathbb{F}_2^n, v \in \mathbb{F}_2^n, u \neq 0, v \neq 0} \mathsf{BCT}_S(u,v). \tag{9}$$

2.3 Equivalence of S-boxes

One important classification of S-boxes is partitioning them into equivalence classes as some of the cryptographic properties in these classes remain invariant. Therefore, one member S-box of an equivalence class describes such cryptographic properties of the others.

Let S, S' be two n-bit S-boxes, the equivalence relations between S and S' are defined as follows.

Definition 6 (XOR Equivalence)**.** S is XOR-equivalent to S' if there exist $c, d \in \mathbb{F}_2^n$ with $(c,d) \neq (0,0)$ such that

$$S'(x) = S(x \oplus c) \oplus d, \qquad \text{for all } x \in \mathbb{F}_2^n. \tag{10}$$

Definition 7 (Affine Equivalence)**.** S is affine equivalent to S' if there exist binary nonsingular matrices A and B, and $c, d \in \mathbb{F}_2^n$ such that

$$S'(x) = B \cdot S(A \cdot x \oplus c) \oplus d, \qquad \text{for all } x \in \mathbb{F}_2^n. \tag{11}$$

Definition 8 (Extended Affine Equivalence)**.** S is extended affine equivalent to S' if there exist binary nonsingular matrices A and B, $c, d \in \mathbb{F}_2^n$ and a binary matrix L such that

$$S'(x) = B \cdot S(A \cdot x \oplus c) \oplus d + L \cdot x, \qquad \text{for all } x \in \mathbb{F}_2^n. \tag{12}$$

Definition 9 (CCZ-Equivalence [CCZ98])**.** S is CCZ-equivalent to S' if there exists an affine permutation of $\mathcal{A}$ of $\mathbb{F}_2^n \times \mathbb{F}_2^n$ such that

$$\{(x, S'(x)), x \in \mathbb{F}_2^n\} = \mathcal{A}(\{(x, S(x)), x \in \mathbb{F}_2^n\}).$$

That means the graph of $S'(x)$ can be obtained by applying an affine permutation on the graph of $S(x)$.

Some cryptographic properties that are preserved under these three equivalence relations are differential uniformity and linearity property. The algebraic degree of a function is invariant under the extended affine equivalence (if the function is not affine), however, it is not invariant under CCZ-equivalence.

2.4 Mixed Integer Linear Programming

Integer linear programming is a class of mathematical problems which seek to maximize (or minimize or check feasibility of) a linear function $f(x_1, x_2, \ldots, x_n)$ given some linear constraints and bounds in terms of decision variables $\{x_i\}_{i=1}^n$. When only some and not all decision variables are integers, the problem is called Mixed Integer Linear Programming or MILP in short.

The high level idea is to convert a cryptographic problem as a MILP model, pass that model to the optimizer solver which then returns either the desired solution or an infeasible model meaning no solution exists. Examples of such solvers include Gurobi optimizer [gur], SCIP [GFG+16], CPLEX Optimization Studio [IBM] and OR-Tools [ort].

Applications of MILP and MIQCP. The use of MILP in symmetric key cryptography has been amplified over the last decade since the seminal work of Mouha et al. [MWGP11] and Wu and Wang [WW11]. Later, there has been a series of works targeting search of differential/linear trails and related attacks via MILP [SHW+14, FWG+16, CJF+16, ST17, AST+17, CHP+17, BC20, SSS+20, MR22, NPE23]. Concurrently, the usage of MILP was extended to boomerang and rectangle attacks [CHP+17, DDV20, HNE22, LMR22], meet-in-the-middle attacks [BDG+21, DHS+21, HDS+22, SSS+23], and integral attacks, cube and division-property based attacks [XZBL16, SWLW16, SG18, LDB+19, RHSS21, HST+21, DL22]. Very recently, the works in [BGG+23, LJC23] employed MIQCP for the cryptanalysis of ARX ciphers.

3 Modeling S-box Output with a Permutation Matrix

In a usual setup an n-bit S-box is represented as an integer array $[S(0), \ldots, S(2^n - 1)]$ which is a permutation of $[0, 1, \ldots, 2^n - 1]$, where $S : i \mapsto S(i)$. Therefore, in the MILP setting, if we model an n-bit S-box output array by an array of variables $[y_0, \ldots, y_{2^n-1}]$, we must ensure that 1) each y_i can assume only an integer value from $\{0, \ldots, 2^n - 1\}$ and 2) the array $[y_0, \ldots, y_{2^n-1}]$ is a permutation of $[0, 1, \ldots, 2^n - 1]$. In what follows, we describe how to model S-box in terms of a permutation matrix.

Consider the $2^n \times 2^n$ permutation matrix $\mathsf{P} = [p_{i,j}]$. Then

$$\mathsf{P} \cdot [0, 1, \ldots, 2^n - 1]^T$$

is a permutation of $[0, 1, \ldots, 2^n - 1]$. Thus, P uniquely identifies an S-box by its binary entries $p_{i,j}$ as $S : i \mapsto j$ if and only if $p_{i,j} = 1$ (see Ex. 1). So we have

$$y_i = j \quad \text{if and only if} \quad p_{i,j} = 1.$$

Further, as per the property of a permutation matrix, for each row and each column of P, the row-sum and column-sum should be 1. Then we have the following constraints to model the S-box.

$$\begin{aligned} \text{S-box values: } & y_i = \sum_{j=0}^{2^n-1} j \cdot p_{i,j}, \text{ for } i = 0, \ldots, 2^n - 1 \\ \text{Row-sum: } & \sum_{j=0}^{2^n-1} p_{i,j} = 1, \text{ for } i = 0, \ldots, 2^n - 1 \\ \text{Column-sum: } & \sum_{i=0}^{2^n-1} p_{i,j} = 1, \text{ for } j = 0, \ldots, 2^n - 1 \end{aligned} \tag{13}$$

We denote the above procedure by model_sbox_with_permutation_matrix and present it in Algo. 1. For an n-bit S-box, this takes n and MILP model $\mathcal{M}$ as inputs and updates $\mathcal{M}$ by adding variables and constraints of (13).

Algorithm 1: model_sbox_with_permutation_matrix

Input: n and $\mathcal{M}$
Output: $\mathcal{M}$

1 $p_{i,j} \in \{0,1\}$, for $i = 0, \ldots, 2^n - 1$ and $j = 0, \ldots, 2^n - 1$ ▷ Variables
2 $y_i \in \{0, 1, \ldots, 2^n - 1\}$, for $i = 0, \ldots, 2^n - 1$ ▷ Variables
3 $y_i = \sum_{j=0}^{2^n-1} j \cdot p_{i,j}$, for $i = 0, \ldots, 2^n - 1$ ▷ S-box values constraints
4 $\sum_{j=0}^{2^n-1} p_{i,j} = 1$, for $i = 0, \ldots, 2^n - 1$ ▷ Row-sum constraints
5 $\sum_{i=0}^{2^n-1} p_{i,j} = 1$, for $j = 0, \ldots, 2^n - 1$ ▷ Column-sum constraints
6 **return** $\mathcal{M}$

Example 1 (Permutation matrix for an S-box)**.** The S-box $[7, 6, 5, 3, 4, 1, 0, 2]$ can be obtained from the 8×8 permutation matrix

$$\mathsf{P} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Note that if we simply run Algo. 1 with Gurobi [gur], it will return an arbitrary S-box.

4 Recovering S-boxes from (Partial) Difference Distribution Table

In this section, we first describe our method for recovering an S-box from a given DDT using MILP. Then, we demonstrate that MILP is also effective when only a fraction of the DDT rows are provided.

When we determine the DDT of an S-box, we do the following steps: 1) take the integer array of S-box output, 2) find all output differences for an input difference α, 3) count the

number of output differences that match to a β, 4) make this count an entry corresponding to row α and column β and 5) complete the row for α by varying β.

Similarly, in a MILP setup, we formulate the problem of finding an n-bit S-box from a given DDT by considering the S-box output as an array of integer variables $\{y_i\}_{i=0}^{2^n-1}$ ensuring that this array represents a permutation of $[0, \ldots, 2^n - 1]$, then model the output differences and finally obtain the array of variables $[y_0, \ldots, y_{2^n-1}]$ that maps to the given DDT. The last step occurs when MILP solver picks a solution that satisfies all the constraints. Two questions remains now. First, how to ensure $[y_i]_{i=0}^{2^n-1}$ to represent a permutation of $[0, \ldots, 2^n - 1]$. We solve this by mapping $[y_i]_{i=0}^{2^n-1}$ to a permutation matrix of order $2^n \times 2^n$. Second, how to model the output differences of variables $\{y_i\}$ and connect them to DDT. We solve this by enumerating solutions corresponding to a fixed input difference.

Therefore, in summary, our method is divided into three steps, namely 1) model (target) S-box using the permutation matrix, 2) enumerate solution pairs for a given input difference and model XOR of their S-box output values and 3) map Step-2 with entries of the DDT. We discuss these three steps along with examples. In the end, we give experimental evidences to show the correctness of our approach.

4.1 Modeling the XOR of S-box Output Values

Let $i \neq j$. Our goal is to model S-box output difference $y_i \oplus y_j$ given that y_i and y_j are integer variables. We tackle this problem by working with solution pairs for a given input difference.

Consider an input difference α from $\{1, \ldots, 2^n - 1\}$. Define

$$\mathcal{X}_n(\alpha) \coloneqq \{(i, j) \mid i < j \text{ and } i \oplus j = \alpha\}.$$

Note that if the pair of integers (i, j) is such that $i \oplus j = \alpha$, then so is (j, i). Thus, we add the condition $i < j$ in the definition of $\mathcal{X}_n(\alpha)$ to break this symmetry.

We now define $\mathcal{B}_n(\alpha) \coloneqq \{y_i \oplus y_j \mid (i, j) \in \mathcal{X}_n(\alpha)\}$ as the set consisting of all possible output differences corresponding to α. The set $\mathcal{B}_n(\alpha)$ actually captures the notion of the set of differentials $\{S(i) \oplus S(j = i \oplus \alpha)\}$ as we would do in case we have a defined S-box S. Note that there might be multiple pairs having the same output difference $S(i) \oplus S(j)$, for $j = i \oplus \alpha$, so we have to consider that in $\mathcal{B}_n(\alpha)$, there are some duplicate entries. In other words, $\mathcal{B}_n(\alpha)$ is a multiset and its cardinality is 2^{n-1}. So it is now left to model XOR of $y_i \oplus y_j$ for each nonzero integer α.

Suppose, a and b are two binary variables, if we bring a new binary variable c, where $c = a \oplus b$, then the XOR operation can be modeled by the following relation [FWG+16]:

$$a + b + c = 2 \cdot d, \text{ where } d \in \{0, 1\}. \tag{14}$$

Suppose $y_i \oplus y_j = c_{i,j}$, then using (14) it is now possible to model this XOR operation. Let $y_i = (y_{i,0}, \ldots, y_{i,n-1})$, $y_j = (y_{j,0}, \ldots, y_{j,n-1})$ and $c_{i,j} = (c_{i,j,0}, \ldots, c_{i,j,n-1})$ be the binary decomposition of the variables y_i, y_j and $c_{i,j}$, respectively. Then $y_i \oplus y_j = c_{i,j}$ can be modeled by the following set of constraints:

$$\begin{aligned} y_{i,0} + y_{j,0} + c_{i,j,0} &= 2 \cdot d_{i,j,0} \\ y_{i,1} + y_{j,1} + c_{i,j,1} &= 2 \cdot d_{i,j,1} \\ &\vdots \\ y_{i,n-1} + y_{j,n-1} + c_{i,j,n-1} &= 2 \cdot d_{i,j,n-1} \end{aligned} \tag{15}$$

where $d_{i,j,0}, \ldots, d_{i,j,n-1}$ are binary variables.

We now have all the steps that can be combined to model the output difference of the S-box $[y_0, y_1, \ldots, y_{2^n-1}]$. We denote this modeling procedure by model_xor_solution (see Algo. 2) and do it in two steps as follows.

1. Compute the solution sets $\mathcal{X}_n(\alpha)$ for $\alpha = 1, \ldots, 2^n - 1$ (Lines 1 to 3).
2. For each α and for each solution pair $(i, j) \in \mathcal{X}_n(\alpha)$, we model the XOR operation $y_i \oplus y_j = c_{i,j}$. This is shown in Lines 8 to 15.

Algorithm 2: model_xor_solution

Input: n and $\mathcal{M}$
Output: $\mathcal{M}$

```
1  for α = 1 to 2^n − 1 do
2  |  Compute X_n(α)
3  end
4  for i = 0 to 2^n − 1 do
5  |  y_{i,k} ∈ {0,1}, for k = 0, ..., n − 1                ▷ Variables
6  |  y_i = Σ_{k=0}^{n−1} 2^k · y_{i,k}                     ▷ Binary decomposition
7  end
8  for α = 1 to 2^n − 1 do
9  |  for (i, j) ∈ X_n(α) do
10 |  |  for k = 0 to n − 1 do
11 |  |  |  c_{i,j,k}, d_{i,j,k} ∈ {0,1}                    ▷ Variables
12 |  |  |  y_{i,k} + y_{j,k} + c_{i,j,k} = 2 · d_{i,j,k}   ▷ Binary XOR constraint
13 |  |  end
14 |  end
15 end
16 return M
```

To make this part easier to follow, we provide an example below.

Example 2. Let $n = 3$ and $\alpha = 1$. Then we have $\mathcal{X}_3(1) = \{(0,1),(2,3),(4,5),(6,7)\}$ and $\mathcal{B}_3(1) = \{y_0 \oplus y_1, y_2 \oplus y_3, y_4 \oplus y_5, y_6 \oplus y_7\}$. To model $\mathcal{B}_3(1)$, we have the following constraints as given in Table 1.

Table 1: Constraints to model $\mathcal{B}_3(1)$ in Ex. 2

Binary decomposition	
$y_0 = y_{0,0} + 2 \cdot y_{0,1} + 4 \cdot y_{0,2}$	$y_1 = y_{1,0} + 2 \cdot y_{1,1} + 4 \cdot y_{1,2}$
$y_2 = y_{2,0} + 2 \cdot y_{2,1} + 4 \cdot y_{2,2}$	$y_3 = y_{3,0} + 2 \cdot y_{3,1} + 4 \cdot y_{3,2}$
$y_4 = y_{4,0} + 2 \cdot y_{4,1} + 4 \cdot y_{4,2}$	$y_5 = y_{5,0} + 2 \cdot y_{5,1} + 4 \cdot y_{5,2}$
$y_6 = y_{6,0} + 2 \cdot y_{6,1} + 4 \cdot y_{6,2}$	$y_7 = y_{7,0} + 2 \cdot y_{7,1} + 4 \cdot y_{7,2}$
$y_0 \oplus y_1$	$y_2 \oplus y_3$
$y_{0,0} + y_{1,0} + c_{0,1,0} = 2 \cdot d_{0,1,0}$	$y_{2,0} + y_{3,0} + c_{2,3,0} = 2 \cdot d_{2,3,0}$
$y_{0,1} + y_{1,1} + c_{0,1,1} = 2 \cdot d_{0,1,1}$	$y_{2,1} + y_{3,1} + c_{2,3,1} = 2 \cdot d_{2,3,1}$
$y_{0,2} + y_{1,2} + c_{0,1,2} = 2 \cdot d_{0,1,2}$	$y_{2,2} + y_{3,2} + c_{2,3,2} = 2 \cdot d_{2,3,2}$
$y_4 \oplus y_5$	$y_6 \oplus y_7$
$y_{4,0} + y_{5,0} + c_{4,5,0} = 2 \cdot d_{4,5,0}$	$y_{6,0} + y_{7,0} + c_{6,7,0} = 2 \cdot d_{6,7,0}$
$y_{4,1} + y_{5,1} + c_{4,5,1} = 2 \cdot d_{4,5,1}$	$y_{6,1} + y_{7,1} + c_{6,7,1} = 2 \cdot d_{6,7,1}$
$y_{4,2} + y_{5,2} + c_{4,5,2} = 2 \cdot d_{4,5,2}$	$y_{6,2} + y_{7,2} + c_{6,7,2} = 2 \cdot d_{6,7,2}$

Note that the newly introduced variables $c_{i,j,k}$ in (15) correspond to output differences $y_i \oplus y_j$ and so they are related to input difference $\alpha = i \oplus j$. Next, we show how to link these variables to the entries of DDT that will establish the connection between the variables $\{y_i\}$ and DDT.

4.2 Mapping Output Differences to Entries of DDT

Note that in the DDT for all S-boxes, row 0 (input difference $\alpha = 0$) and column 0 (output difference $\beta = 0$) are fixed: $((0,0)$-th entry is 2^n and rest of the entries are 0). This happens as the S-box is a permutation. As our modeled S-box $[y_0, y_1, \ldots, y_{2^n-1}]$ is ensured to be a permutation, therefore, we do not consider the row 0 and column 0 in the modeling of the given DDT.

To connect binary variables $c_{i,j,k}$ with DDT entries we proceed row by row. Before giving a general description of this step, we first illustrate the idea in Ex. 3.

Example 3. Consider the first row (input difference $\alpha = 1$) from the DDT in Table 2. From Ex. 2, we have $\mathcal{B}_3(1) = \{y_0 \oplus y_1, y_2 \oplus y_3, y_4 \oplus y_5, y_6 \oplus y_7\}$. Each element in $\mathcal{B}_3(1)$

Table 2: DDT of a 3-bit S-box

$\alpha \setminus \beta$	0	1	2	3	4	5	6	7
0	8	0	0	0	0	0	0	0
1	0	2	2	0	0	2	2	0
2	0	0	2	2	2	2	0	0
3	0	2	0	2	2	0	2	0
4	0	2	0	2	0	2	0	2
5	0	0	2	2	0	0	2	2
6	0	2	2	0	2	0	0	2
7	0	0	0	0	2	2	2	2

can take values from 1 to 7. Thus, we introduce a binary matrix $\mathsf{W}_1 = [w_{i,j,k}]$, where $i \oplus j = \alpha = 1$, and $k = 1, \ldots, 7$ and

$$w_{i,j,k} = 1 \quad \text{if and only if} \quad y_i \oplus y_j = k. \tag{16}$$

For instance, if $y_0 \oplus y_1 = 2$, then $w_{0,1,2} = 1$. In Table 3, we show the relation between $y_i \oplus y_j$ and the $w_{i,j,k}$ entries of W_1.

We have the following important observations related to matrix W_1.

1. **The row-sum must be 1.** This is due to the property as given in (16). Therefore for the row corresponding to $y_i \oplus y_j$,

$$\sum_{k=1}^{7} w_{i,j,k} = 1. \tag{17}$$

Table 3: Relating $y_i \oplus y_j$ with the matrix W_1

$\alpha = 1 \setminus \beta$	1	2	3	4	5	6	7
$y_0 \oplus y_1$	$w_{0,1,1}$	$w_{0,1,2}$	$w_{0,1,3}$	$w_{0,1,4}$	$w_{0,1,5}$	$w_{0,1,6}$	$w_{0,1,7}$
$y_2 \oplus y_3$	$w_{2,3,1}$	$w_{2,3,2}$	$w_{2,3,3}$	$w_{2,3,4}$	$w_{2,3,5}$	$w_{2,3,6}$	$w_{2,3,7}$
$y_4 \oplus y_5$	$w_{4,5,1}$	$w_{4,5,2}$	$w_{4,5,3}$	$w_{4,5,4}$	$w_{4,5,5}$	$w_{4,5,6}$	$w_{4,5,7}$
$y_6 \oplus y_7$	$w_{6,7,1}$	$w_{6,7,2}$	$w_{6,7,3}$	$w_{6,7,4}$	$w_{6,7,5}$	$w_{6,7,6}$	$w_{6,7,7}$

Note that the binary decomposition of $y_i \oplus y_j$ is $(c_{i,j,0}, c_{i,j,1}, c_{i,j,2})$ (see (15)). Thus, we have the following constraint that describes $y_i \oplus y_j$ in terms of $w_{i,j,k}$ via $(c_{i,j,0}, c_{i,j,1}, c_{i,j,2})$.

$$y_i \oplus y_j = \sum_{k=1}^{7} k \cdot w_{i,j,k} = c_{i,j,0} + 2 \cdot c_{i,j,1} + 4 \cdot c_{i,j,2} \tag{18}$$

2. **For each column, the column-sum equals $\frac{\mathsf{DDT}[1][\mathbf{column}]}{2}$.** This is because, in Table 3, $y_i \oplus y_j$ and $y_{i'} \oplus y_{j'}$ can assume the same value k. Also, we have to divide the values of DDT by 2 as we already have excluded the symmetric solutions from $\mathcal{X}_3(1)$ by forcing $(i < j)$. Thus, we have the following constraint for each $k = 1, \ldots, 7$ that relates the variables $w_{i,j,k}$ and the given DDT values.

$$\sum_{(i,j) \in \mathcal{X}_3(1)} w_{i,j,k} = \frac{\mathsf{DDT}[1][k]}{2} \tag{19}$$

Example 3 can now be easily extended for a DDT of a general n-bit S-box. We derive all the constraints for the n-bit S-box $[y_0, \ldots, y_{2^n-1}]$ as per (17), (18) and (19) by varying $\alpha \in \{1, \ldots, 2^n - 1\}$. These steps are accumulated in model_map_output_difference_to_ddt and presented in Algo. 3.

Algorithm 3: model_map_output_difference_to_ddt

Input: n, $\mathcal{M}$ and DDT
Output: $\mathcal{M}$

for $\alpha = 1$ *to* $2^n - 1$ **do**
  $w_{i,j,k} \in \{0,1\}$ for $(i,j) \in \mathcal{X}_n(\alpha)$ and $k = 1, \ldots, 2^n - 1$ ▷ Variables
  for $(i,j) \in \mathcal{X}_n(\alpha)$ **do**
    $\sum_{k=1}^{2^n-1} w_{i,j,k} = 1$ ▷ Row-sum constraint
    $\sum_{k=1}^{2^n-1} k \cdot w_{i,j,k} = \sum_{\ell=0}^{n-1} 2^\ell \cdot c_{i,j,\ell}$ ▷ Mapping variables
  end
  for $k = 1$ *to* $2^n - 1$ **do**
    $\sum_{(i,j) \in \mathcal{X}_n(\alpha)} w_{i,j,k} = \frac{\mathsf{DDT}[\alpha][k]}{2}$ ▷ DDT constraint
  end
end
return $\mathcal{M}$

4.3 Modeling (partial) DDT to S-box

Recovering S-box from a full DDT. At this point we have all the steps that complete the modeling of an S-box from a DDT. So our final step is to combine the three functions:

model_sbox_with_permutation_matrix,
model_xor_solution,
model_map_output_difference_to_ddt,

and create the final model. We present the complete algorithm to recover an S-box from a given DDT in Algo. 4. Consequently, an infeasible model means the given DDT is not valid, i.e., it does not correspond to any S-box. On the other hand, a feasible model will output an S-box.

Algorithm 4: model_ddt_to_sbox

Input: n, DDT and empty $\mathcal{M}$
Output: S-box $= [y_0, y_1, \ldots, y_{2^n-1}]$ or given DDT is invalid

1 $\mathcal{M} \leftarrow$ model_sbox_with_permutation_matrix$(n, \mathcal{M})$
2 $\mathcal{M} \leftarrow$ model_xor_solution$(n, \mathcal{M})$
3 $\mathcal{M} \leftarrow$ model_map_output_difference_to_ddt$(n, \mathcal{M}, \text{DDT})$
4 **if** $\mathcal{M}$ *is feasible* **then**
5 | **return** $[y_0, y_1, \ldots, y_{2^n-1}]$
6 **end**
7 **else**
8 | **return** DDT is invalid
9 **end**

Recovering S-box from a partial DDT. Let $\{\alpha_1, \alpha_2, \ldots, \alpha_l\} \subset \{1, 2, \ldots, 2^{n-1}\}$ be l distinct input differences with $1 \leq l < 2^n$. Suppose a partial DDT with rows corresponding to these input differences are given. Then, in Line 2 and Line 3 of Algo. 4, we model these l rows only. Solving the modified model will either return a S-box having this partial DDT or an infeasible model.

4.4 Experimental Validation and Discussion

We run our experiments on AMD EPYC 7763 64-Core Processor using 8 threads. We use Gurobi 10 [gur] and OR-Tools [ort] as the underlying MILP solvers to discuss the efficiency. Note that for a fair comparison among two solvers, we did not use any internal optimization tricks of these solvers. Furthermore, we emphasize that we will be using the same experimental setup throughout the paper.

Though the model size (number of variables and constraints) can be easily obtained from Algo. 4, we report the exact model details and timings of reconstructing S-boxes from DDT in Table 4. Notice that we obtain XOR-equivalent S-boxes as the output. From Table 4, we observe that running the exact model with two different solvers give different outcomes. For instance, we set a time limit of one hour, but within that time, Gurobi does not produce an S-box corresponding to the DDT of 5 and 6-bit inverse S-boxes and APN6. On the other hand, OR-Tools give those S-boxes in seconds.

Table 4: Model size and timings for DDT to S-box. The symbol '-' denotes that no solution is obtained with a time limit of one hour.

DDT of	Model details			Time (sec.)	
	Variables		Linear Constraints	Gurobi 10	OR-Tools
	Integer	Binary			
Example 3	8	452	221	0.01	0.03
4-bit Inverse	16	3080	1009	0.4	0.21
PRESENT	16	3080	1009	0.17	0.23
GIFT	16	3080	1009	0.3	0.23
SKINNY	16	3080	1009	0.03	0.3
ASCON	32	21520	4561	0.19	2.87
Keccak	32	21520	4561	0.22	2.73
5-bit Inverse	32	21520	4561	-	1.23
6-bit Inverse	64	155680	20353	-	14.75
APN6	64	155680	20353	-	10.31

In Table 5, we demonstrate the power of our method for partial DDT's by taking APN6 and 5-bit inverse S-boxes as examples. Notice that for at least 17 (resp. 9) rows we

could recover a 2-differential uniform S-box while we obtain a 4-differential uniform S-box with 16 (resp. 8) rows of DDT of APN6 (resp. 5-bit inverse). This finding gives a strong evidence that 17 properly filled rows (this includes the first row as well) are sufficient to construct an APN in dimension 6.

Table 5: Reconstructing S-box from first l rows of DDT with OR-Tools. The minimum number of rows up to which the properties of obtained S-boxes remain identical is highlighted in blue. First row of each function corresponds to the full DDT.

DDT of	# rows (l)	Time (s)	Diff. unif.	Linearity	Differential-linear unif.	Boomerang unif.
APN6	64	10.31	2	16	32	2
	49	13.82	2	16	32	2
	33	17.24	2	16	32	2
	25	20.76	2	16	32	2
	17	22.23	2	16	32	2
	16	22.95	4	24	32	14
5-bit Inverse	32	0.73	2	12	4	2
	28	0.92	2	12	4	2
	24	0.91	2	12	4	2
	20	1.07	2	12	4	2
	16	0.77	2	12	4	2
	12	0.64	2	12	4	2
	9	1.13	2	12	4	2
	8	1.02	4	16	8	10

Table 6: 6-bit S-boxes with differential uniformity 2 and 4 from Table 5.

Rows	Diff. unif.	S-box
17	2	0, 63, 35, 46, 47, 43, 37, 25, 24, 18, 21, 57, 6, 13, 22, 17, 15, 59, 16, 3, 34, 58, 53, 33, 40, 9, 50, 32, 30, 41, 56, 55, 1, 39, 10, 19, 42, 61, 31, 23, 8, 62, 51, 14, 44, 49, 29, 11, 45, 2, 48, 20, 36, 12, 38, 4, 26, 5, 28, 60, 54, 52, 27, 7
16	4	0, 63, 35, 46, 47, 43, 37, 25, 24, 18, 21, 57, 6, 13, 22, 17, 1, 39, 10, 19, 42, 61, 31, 23, 8, 62, 51, 14, 44, 49, 29, 11, 2, 45, 20, 48, 12, 36, 4, 38, 5, 26, 60, 28, 52, 54, 7, 27, 3, 16, 59, 15, 33, 53, 58, 34, 32, 50, 9, 40, 55, 56, 41, 30

We list the S-boxes corresponding to first 17 and 16 rows of APN6's DDT in Table 6. Notice that the two S-boxes in this table are closely related and one can be obtained from another by flipping specific outputs. Also, the obtained 2-differential uniform S-box is affine equivalent to APN6 with the parameters, i.e, $\mathsf{APN6} = A \circ \mathsf{S} \circ B(x \oplus a) \oplus b$ where $a = 44$, $b = 7$ and the matrices are

$$A = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

5 Recovering S-boxes from (Partial) Linear Approximation Table

In this section, first we explain our method to recover an S-box from a given LAT using MILP and next we show that it is also effective for a partial LAT where a fraction of rows are given. Our method consists of two steps, namely (1) model (target) S-box via a permutation matrix and (2) model the associated linear equation corresponding to each pair of input mask and output mask which involves modeling the scalar product as well XOR operations, and 3) model the count of the occurrences when the linear relation holds.

As done in Sec. 3, we model the S-box as an array of integer variables $[y_0, \ldots, y_{2^n-1}]$ using a permutation matrix, so we skip this part. We move on to the second step in which we model the linear relation corresponding to a pair of input and output masks. Note that here we use the sign " $\cdot$ " interchangeably for both integer products or scalar products.

5.1 Modeling the Linear Equation Corresponding to a pair of Input Output Masks

Let λ and γ be the (integer) input and (integer) output masks, respectively. Recall from Def. 2 that

$$\mathsf{LAT}(\lambda, \gamma) = \#\{\vec{i} \in \mathbb{F}_2^n : \vec{\lambda} \cdot \vec{i} = \vec{\gamma} \cdot \vec{y_i}\} - 2^{n-1}, \tag{20}$$

where $\vec{i}, \vec{\lambda}, \vec{\gamma}$ and $\vec{y_i}$ are the binary vector representation of i, λ, γ and y_i respectively. Note that (20) uses the vector notation explicitly in contrast to Def. 2. This is for the better understanding of our modeling and this will be used throughout the rest of the paper.

We now rewrite this relation as follows.

$$\mathsf{LAT}(\lambda, \gamma) + 2^{n-1} = \#\{\vec{i} \in \mathbb{F}_2^n : \vec{\lambda} \cdot \vec{i} \oplus \vec{\gamma} \cdot \vec{y_i} = 0\} \tag{21}$$

From (21), it is evident that if we can model the number of $\vec{i}$'s such that

$$\vec{\lambda} \cdot \vec{i} \oplus \vec{\gamma} \cdot \vec{y_i} = 0, \tag{22}$$

then we can map $\{y_i\}, \lambda, \gamma$ with $\mathsf{LAT}(\lambda, \gamma)$. Suppose, $\vec{y_i} = (y_{i,0}, \ldots, y_{i,n-1})$. Then each integer variable y_i of the S-box is connected as

$$y_i = \sum_{k=0}^{n-1} 2^k \cdot y_{i,k} \ .$$

Before we model the solution of (22) (in terms of $\vec{i}$), we need to model the two scalar product terms therein. These steps are as follows.

1. **Modeling $\vec{\lambda} \cdot \vec{i}$**
   Let $\vec{\lambda} = (\lambda_0, \ldots, \lambda_{n-1})$ and $\vec{i} = (i_0, \ldots, i_{n-1})$. Suppose $u = \vec{\lambda} \cdot \vec{i}$, then we have

$$u = \bigoplus_{k=0}^{n-1} \lambda_k \cdot i_k, \ \ u \in \{0, 1\}. \tag{23}$$

   We know the value of u as the input vectors $\vec{i} \in \mathbb{F}_2^n$ and input mask $\vec{\lambda} \in \mathbb{F}_2^n$ are known.

2. **Modeling** (22)
   Suppose $\vec{\gamma} = (\gamma_0, \ldots, \gamma_{n-1})$. Then (22) is an XOR of $n+1$ terms:

$$u \oplus \left(\bigoplus_{k=0}^{n-1} \gamma_k \cdot y_{i,k} \right) = 0.$$

   So this equation can be modeled as

$$\sum_{k=0}^{n-1} \gamma_k \cdot y_{i,k} + u + z_{\lambda,\gamma,i} = 2 \cdot v_{\lambda,\gamma,i}, \;\; z_{\lambda,\gamma,i} \in \{0,1\}, v_{\lambda,\gamma,i} \in \{0, \ldots, \left\lceil \frac{n+1}{2} \right\rceil\}. \quad (24)$$

3. **Modeling the number of solutions of** (22)
   We introduce binary variables $w_{\lambda,\gamma,i}$ for all $i = 0, \ldots, 2^n - 1$ with the following property:

$$w_{\lambda,\gamma,i} = 1 \quad \text{if and only if} \quad \vec{\lambda} \cdot \vec{i} \oplus \vec{\gamma} \cdot \vec{y_i} = 0 \; .$$

   Then from (21), it is now clear that

$$\mathsf{LAT}(\lambda, \gamma) + 2^{n-1} = \sum_{i=0}^{2^n-1} w_{\lambda,\gamma,i}, \quad w_{\lambda,\gamma,i} \in \{0,1\} \; . \quad (25)$$

   After this step, it only remains to connect $w_{\lambda,\gamma,i}$ variables with y_i.

4. **Linking** (24) **with** (25)
   From (24), we note that $z_{\lambda,\gamma,i} = 0$ if and only if $\vec{\lambda} \cdot \vec{i} \oplus \vec{\gamma} \cdot \vec{y_i} = 0$. Therefore for $i = 0, \ldots, 2^n - 1$,

$$z_{\lambda,\gamma,i} = 1 - w_{\lambda,\gamma,i}, \;\; z_{\lambda,\gamma,i} \in \{0,1\}. \quad (26)$$

Recovering S-box from a full LAT. We repeat the above procedure for modeling all the linear equations for all λ and γ. As the first row and first column of LAT is constant for all S-boxes, so we only consider $\lambda = 1, \ldots, 2^n - 1$ and $\gamma = 1, \ldots, 2^n - 1$. The entire algorithm to recover an S-box from a given LAT is depicted in Algo. 5 where Lines 6 to 18 describe the aforementioned modeling.

Recovering S-box from a partial LAT. In case of partial LAT, where l rows corresponding to input masks $\{\lambda_1, \lambda_2, \ldots, \lambda_l\} \subset \{1, 2, \ldots, 2^{n-1}\}$ are given, we replace Line 6 of the algorithm with:

$$\text{for } \lambda \in \{\lambda_1, \lambda_2, \ldots, \lambda_l\}. \quad (27)$$

5.2 Experimental Validation and Discussion

We checked Algo. 5 with the LAT of inverse mapping in dimensions 4, 5 and 6, LAT of S-boxes of 3-bit χ, PRESENT, GIFT, SKINNY, ASCON, Keccak and APN6. We successfully recovered these S-boxes which shows the correctness of our algorithm. In Table 7, we report the timings to recover these S-boxes along with the model sizes. We observe from Table 7 that within 2 minutes we recovered 6-bit S-boxes from their LAT using Gurobi. This suggests that in case of a partial LAT the timings will increase with reduced number of rows. In fact, we find that given the first 48 rows of the LAT of APN6, we could recover the exact S-box in 40 minutes using Gurobi.

Algorithm 5: model_lat_to_sbox

Input: n, LAT and empty $\mathcal{M}$
Output: S-box $= [y_0, y_1, \ldots, y_{2^n-1}]$ or given LAT is invalid

$\mathcal{M} \leftarrow$ model_sbox_with_permutation_matrix$(n, \mathcal{M})$
for $i = 0$ *to* $2^n - 1$ **do**
  $y_{i,k} \in \{0, 1\}$ for $k = 0, \ldots, n-1$ ▷ Variables
  $y_i = \sum_{k=0}^{n-1} 2^k \cdot y_{i,k}$ ▷ Binary decomposition
end
for $\lambda = 1$ *to* $2^n - 1$ **do**
  for $\gamma = 1$ *to* $2^n - 1$ **do**
    for $i = 0$ *to* $2^n - 1$ **do**
      $u = \bigoplus_{k=0}^{n-1} \lambda_k \cdot i_k$
      $v_{\lambda,\gamma,i} \in \{0, \ldots, \left\lceil \frac{n+1}{2} \right\rceil\}$ ▷ Variable
      $z_{\lambda,\gamma,i} \in \{0, 1\}$ ▷ Variable
      $\sum_{k=0}^{n-1}(\gamma_k \cdot y_{i,k}) + u + z_{\lambda,\gamma,i} = 2 \cdot v_{\lambda,\gamma,i}$ ▷ $\vec{\lambda} \cdot \vec{i} \oplus \vec{\gamma} \cdot \vec{y_i}$ constraint
      $w_{\lambda,\gamma,i} \in \{0, 1\}$ ▷ Variable
      $z_{\lambda,\gamma,i} = 1 - w_{\lambda,\gamma,i}$ ▷ Constraint
    end
    $\sum_{i=0}^{2^n-1} w_{\lambda,\gamma,i} = \mathsf{LAT}(\lambda, \gamma) + 2^{n-1}$ ▷ LAT constraint
  end
end
if $\mathcal{M}$ *is feasible* **then**
  return $[y_0, y_1, \ldots, y_{2^n-1}]$
end
else
  return LAT is invalid
end

Table 7: Model size and timings for LAT to S-box.

LAT of	Model details			Time (sec.)	
	Variables		Linear Constraints	Gurobi	OR-Tools
	Integer	Binary			
3-bit χ	456	984	984	0.01	0.06
4-bit Inverse	3856	8000	7984	0.07	0.8
PRESENT	3856	8000	7984	0.07	0.72
GIFT	3856	8000	7984	0.06	0.73
SKINNY	3856	8000	7984	0.06	0.78
ASCON	31776	64672	64608	3.56	20.91
Keccak	31776	64672	64608	3.40	15.10
5-bit Inverse	31776	64672	64608	4.56	15.09
6-bit Inverse	258112	520576	520384	78.22	672.76
APN6	258112	520576	520384	55.08	539.92

Considering these time constraints, we focus on the LAT of 5-bit inverse S-box to understand what percentage of rows gives the exact S-box. We give our results in Table 8. We find that given at least 17 rows, the inverse S-box could be recovered.

On comparing Table 5 and 8, it is evident that we require less number of rows (almost half) in case of reconstruction of 5-bit inverse S-box from its partial DDT.

We note that each S-box in Table 7 can also be recovered directly given the relationship

Table 8: Reconstructing S-box from the first l rows of 5-bit inverse's LAT with Gurobi. The minimum number of rows up to which we obtain the inverse S-box is highlighted in blue. First row corresponds to the full LAT.

# rows (l)	Time (s)	Diff. unif.	Linearity	Differential-linear unif.	Boomerang unif.
32	4	2	12	4	2
24	10	2	12	4	2
20	15	2	12	4	2
17	22	2	12	4	2
16	10	4	16	8	10

between the full LAT and Walsh-Hadamard transform. However, given the partial LAT, obtaining the same S-box using this relation is an open question.

6 Recovering S-boxes from (Partial) Differential-Linear Connectivity Table

We now describe how we can combine the two approaches as mentioned for DDT and LAT to solve the problem for DLCT. Precisely, we do the following steps, namely (1) model S-box with a permutation matrix similar to Sec. 3 and (2) enumerate solutions for a given input difference, model XOR of their S-box output values (similar to Sec. 4.1), model scalar product of output mask with output differences and then finally map it to DLCT.

6.1 Modeling the Differential-linear Equation for an Input Difference and a Output Mask

Let α and λ be the input difference and output mask, respectively. From Def. 3, we have

$$\mathsf{DLCT}(\alpha,\lambda) + 2^{n-1} = \#\{\vec{i} \in \mathbb{F}_2^n : \vec{\lambda}\cdot\vec{y_i} \oplus \vec{\lambda}\cdot\vec{y}_{i\oplus\alpha} = 0\}, \tag{28}$$

where the notations are from Sec. 5. Following (28), our aim is to model the equation

$$\vec{\lambda}\cdot\vec{y_i} \oplus \vec{\lambda}\cdot\vec{y}_{i\oplus\alpha} = 0 \tag{29}$$

and then count the number of solutions which will relate to the $\mathsf{DLCT}(\alpha,\lambda)$ value. We derive the set

$$\mathcal{X}_n(\alpha) \coloneqq \{(i,j) \mid i < j \text{ and } i \oplus j = \alpha\}$$

as we did in Sec. 4.1. Note that $|\mathcal{X}_n(\alpha)| = 2^{n-1}$ as it excludes symmetric pairs. We rewrite (29) as

$$\vec{\lambda}\cdot\vec{y_i} \oplus \vec{\lambda}\cdot\vec{y_j} = 0, \quad \text{where } (i,j) \in \mathcal{X}_n(\alpha)\ . \tag{30}$$

It now remains to model (30) and its number of solutions.

1. **Model** (30)
   As done in Sec. 5.1, we model the scalar products by the XOR operations of the binary vectors of the variables and rewrite (30) as follows:

$$\bigoplus_{k=0}^{n-1}(\lambda_k \cdot y_{i,k}) \quad \oplus \quad \bigoplus_{k=0}^{n-1}(\lambda_k \cdot y_{j,k}) = 0\ , \quad \text{where } (i,j) \in \mathcal{X}_n(\alpha).$$

As this equation now simply contains $2n$-XOR operations, therefore we can model it as follows:

$$\sum_{k=0}^{n-1}(\lambda_k \cdot y_{i,k}) + \sum_{k=0}^{n-1}(\lambda_k \cdot y_{j,k}) + z_{i,j,\lambda} = 2 \cdot v_{i,j,\lambda},$$
$$z_{i,j,\lambda} \in \{0,1\}, \;\; v_{i,j,\lambda} \in \{0,\ldots,n\}. \tag{31}$$

2. **Modeling the number of solutions of (30)**
If a pair $(i < j) \in \mathcal{X}_n(\alpha)$ is a solution of (30), so is (j,i), however, solution pairs' membership to $\mathcal{X}_n(\alpha)$ ensures that total number of solutions of (30) is actually $\frac{\mathsf{DLCT}_{(\alpha,\lambda)}+2^{n-1}}{2}$.
We introduce 2^{n-1} binary variables $w_{i,j,\lambda}$ ($(i,j) \in \mathcal{X}_n(\alpha)$) with the following property:

$$w_{i,j,\lambda} = 1 \quad \text{if and only if } \lambda \cdot y_i \oplus \lambda \cdot y_j = 0.$$

Then we have

$$\sum_{(i,j)\in\mathcal{X}_n(\alpha)} w_{i,j,\lambda} = \frac{\mathsf{DLCT}(\alpha,\lambda) + 2^{n-1}}{2} \; . \tag{32}$$

3. **Linking (31) with (32)**
We now link the $w_{i,j,\lambda}$ variables with rest of the model as follows:

$$z_{i,j,\lambda} = 1 - w_{i,j,\lambda} \; . \tag{33}$$

Recovering S-box from a full DLCT. We model the whole DLCT by running the above steps for all (α,λ) and in Algo. 6 we illustrate the complete procedure to recover an S-box from DLCT.

Recovering S-box from a partial DLCT. Given a partial DLCT with rows corresponding to input differences $\{\alpha_1,\alpha_2,\ldots,\alpha_l\} \subset \{1,2,\ldots,2^{n-1}\}$, we model it by replacing Line 5 of Algo. 6 with

$$\text{for } \alpha \in \{\alpha_1,\alpha_2,\ldots,\alpha_l\}.$$

Remark 1. Algorithm 6 can be easily tweaked to recover an S-box from a given ACT. Since $\mathsf{DLCT}(\alpha,\lambda) = \frac{\mathsf{ACT}_{(\alpha,\lambda)}}{2}$, we simply replace Line 14 of Algo. 6 to $\sum_{(i,j)\in\mathcal{X}_n(\alpha)} w_{i,j,\lambda} = \frac{\mathsf{ACT}_{(\alpha,\lambda)/2+2^{n-1}}}{2}$.

6.2 Experimental Validation

Similar to the DDT and LAT experiments, we validated the correctness of Algo. 6 using DLCT/ACT of inverse mapping in dimensions 4 and 5, DLCT/ACT of S-boxes of 3-bit χ, PRESENT, GIFT, SERPENT, SKINNY, ASCON and Keccak. We successfully recovered S-boxes having the same DLCT as of the mentioned S-boxes. Table 9 presents the efficiency of this reconstruction for different DLCT. We notice that especially for the DLCT of 5-bit inverse S-box, Gurobi could not recover the S-box in one hour while OR-Tools gave the corresponding S-box in 36 minutes. While none of the solvers could recover an S-box from the DLCT of the 6-bit inverse S-box. In Table 10, we show the correctness of our approach using the partial DLCT of 4-bit inverse S-box.

Algorithm 6: model_dlct_to_sbox

Input: n, DLCT and empty $\mathcal{M}$
Output: S-box $= [y_0, y_1, \ldots, y_{2^n-1}]$ or given DLCT is invalid

for $\alpha = 1$ *to* $2^n - 1$ **do**
  Compute $\mathcal{X}_n(\alpha)$
end
$\mathcal{M} \leftarrow$ model_sbox_with_permutation_matrix$(n, \mathcal{M})$
for $\alpha = 1$ *to* $2^n - 1$ **do**
  for $\lambda = 1$ *to* $2^n - 1$ **do**
    $w_{i,j,\lambda} \in \{0, 1\}$, for $(i, j) \in \mathcal{X}_n(\alpha)$ ▷ Variables
    for $(i, j) \in \mathcal{X}_m(\alpha)$ **do**
      $v_{i,j,\lambda} \in \{0, \ldots, n\}$ ▷ Variable
      $z_{i,j,\lambda} \in \{0, 1\}$ ▷ Variable
      $\sum_{k=0}^{n-1}(\lambda_k \cdot y_{i,k}) + \sum_{k=0}^{n-1}(\lambda_k \cdot y_{j,k}) + z_{i,j,\lambda} = 2 \cdot v_{i,j,\lambda}$
      $z_{i,j,\lambda} = 1 - w_{i,j,\lambda}$
    end
    $\sum_{(i,j)\in\mathcal{X}_n(\alpha)} w_{i,j,\lambda} = \frac{\mathsf{DLCT}_{(\alpha,\lambda)+2^{n-1}}}{2}$ ▷ DLCT constraint
  end
end
if $\mathcal{M}$ *is feasible* **then**
  return $[y_0, y_1, \ldots, y_{2^n-1}]$
end
else
  return DLCT is invalid
end

Table 9: Model size and timings for DLCT to S-box. The symbol '-' denotes that no solution is obtained with a time limit of one hour.

DLCT of	Model details			Time (sec.)	
	Variables		Linear	Gurobi	OR-Tools
	Integer	Binary	Constraints		
3-bit χ	204	480	473	0.01	0.05
4-bit Inverse	1816	3920	3889	19.68	0.74
SERPENT	1816	3920	3889	7.69	0.87
PRESENT	1816	3920	3889	0.99	0.87
GIFT	1816	3920	3889	4.96	1.04
SKINNY	1816	3920	3889	0.71	0.98
ASCON	15408	31936	31841	0.45	22.78
Keccak	15408	31936	31841	0.44	21.76
5-bit Inverse	15408	31936	31841	-	2154.22
6-bit Inverse	127072	31841	258241	-	-

7 Recovering S-boxes from (Partial) Boomerang Connectivity Table

In this section, we show how to recover an S-box from (partial) BCT, thereby presenting the first novel application of mixed integer quadratic programming in the context of S-boxes reconstruction.

There are two equivalent notions of BCT relation which are given in (7) and (8). To us (8) appears much simpler than (7) as the latter involves both the S-box and its inverse.

Table 10: Reconstructing S-box from the first l rows of DLCT of 4-bit inverse S-box with OR-Tools. The minimum number of rows up to which the properties of obtained S-boxes remain identical is highlighted in blue. First row corresponds to the full DLCT.

# rows (l)	Time (s)	Diff. unif.	Linearity	Differential-linear unif.	Boomerang unif.
16	0.41	4	8	4	6
12	0.33	4	8	4	6
8	0.24	4	8	4	6
5	0.22	4	8	4	6
4	0.13	6	12	8	10

So, we prefer to model BCT based on (8).

At a high level, our method has the following steps, namely 1) model the S-box with a permutation matrix similar to Sec. 3, 2) model the BCT condition and the number of solutions for the pairs of input-output difference.

As before, we model the S-box by an integer variable array $[y_0, \ldots, y_{2^n-1}]$ and

$$y_i = \sum_{k=0}^{n-1} 2^k \cdot y_{i,k}, \quad \text{for } i = 0, \ldots, 2^n - 1 \text{ (from Sec. 3).}$$

As per (8), $\mathsf{BCT}(\alpha, \beta)$ equals the number of pairs (i, j) satisfying

$$y_i \oplus y_j = \beta \text{ and } y_{i\oplus\alpha} \oplus y_{j\oplus\alpha} = \beta. \tag{34}$$

7.1 Modeling the BCT Condition (34)

To do this we first show how to model the two equalities. If (i, j) is a solution of (34), then so is (j, i), therefore, it is enough to consider $(i < j)$. Let

$$\mathcal{Y} = \{(i, j) : i < j\}.$$

Note that for some $(i, j) \in \mathcal{Y}$, it may happen that $(i \oplus \alpha, j \oplus \alpha) \in \mathcal{Y}$, and for some it may not. If $(i \oplus \alpha, j \oplus \alpha) \notin \mathcal{Y}$ then we have $(i \oplus \alpha) > (j \oplus \alpha)$ which means $(j \oplus \alpha, i \oplus \alpha) \in \mathcal{Y}$.

Now, for each $(i, j) \in \mathcal{Y}$ we create the integer variables $c_{i,j}$. Next, we assign $y_i \oplus y_j = c_{i,j}$ and model the XOR operation $y_i \oplus y_j = c_{i,j}$. We then map each (i, j) to the correct $(i \oplus \alpha, j \oplus \alpha)$ or $(j \oplus \alpha, i \oplus \alpha)$. Following the above discussion, we rewrite (34) as follows.

$$c_{i,j} = \beta, \text{ and } \begin{cases} c_{i\oplus\alpha, j\oplus\alpha} = \beta, & \text{if } (i \oplus \alpha, j \oplus \alpha) \in \mathcal{Y} \\ c_{j\oplus\alpha, i\oplus\alpha} = \beta, & \text{o.w.} \end{cases} \tag{35}$$

For the sake of simplicity, in our modeling description, we consider $(i \oplus \alpha, j \oplus \alpha) \in \mathcal{Y}$ if $(i, j) \in \mathcal{Y}$.

1. **Modeling the XOR in (34)**

   We model the XOR operation $y_i \oplus y_j = c_{i,j}$ bit-by-bit as follows.

$$\begin{aligned} c_{i,j,k},\ d_{i,j,k} &\in \{0, 1\}, \quad \text{for } k = 0, \ldots, n-1 \\ y_{i,k} + y_{j,k} + c_{i,j,k} &= 2 \cdot d_{i,j,k}, \quad \text{for } k = 0, \ldots, n-1 \\ c_{i,j} &= \sum_{k=0}^{n-1} 2^k \cdot c_{i,j,k} \end{aligned} \tag{36}$$

   It is to be noted that the above modeling needs to be done only once as it does not depend on α and β.

2. **Modeling the solution of (34)**

   Among all the pairs $(i,j) \in \mathcal{Y}$, some satisfy (35) and others do not. If (i,j) does not satisfy, then we have the following eight cases:

   (a) $c_{i,j} = \beta$ and $c_{i\oplus\alpha,j\oplus\alpha} \geq \beta + 1$,
   (b) $c_{i,j} = \beta$ and $c_{i\oplus\alpha,j\oplus\alpha} \leq \beta - 1$,
   (c) $c_{i\oplus\alpha,j\oplus\alpha} = \beta$ and $c_{i,j} \geq \beta + 1$,
   (d) $c_{i\oplus\alpha,j\oplus\alpha} = \beta$ and $c_{i,j} \leq \beta - 1$,
   (e) $c_{i,j} \geq \beta + 1$ and $c_{i\oplus\alpha,j\oplus\alpha} \geq \beta + 1$,
   (f) $c_{i,j} \geq \beta + 1$ and $c_{i\oplus\alpha,j\oplus\alpha} \leq \beta - 1$,
   (g) $c_{i,j} \leq \beta - 1$ and $c_{i\oplus\alpha,j\oplus\alpha} \geq \beta + 1$,
   (h) $c_{i,j} \leq \beta - 1$ and $c_{i\oplus\alpha,j\oplus\alpha} \leq \beta - 1$.

   It is clear that for a given $(i,j) \in \mathcal{Y}$ only 1 out of the 9 ((35) and (a) – (h)) conditions is true. To model this, we use quadratic constraints. First, let $w_{\alpha,\beta,i,j} \in \{0,1\}$ for all $(i,j) \in \mathcal{Y}$ such that

   $$w_{\alpha,\beta,i,j} = 1 \quad \text{if and only if (35) holds.}$$

   It means when $w_{\alpha,\beta,i,j} = 0$, then (35) is false and only one of remaining 8 conditions is true. Thus, we add 8 different kinds of binary variables $t_{\alpha,\beta,i,j,\ell}$, for $\ell = 0,\ldots,7$ and the following constraints for the modeling of (35).

$$\sum_{\ell=0}^{7} t_{\alpha,\beta,i,j,\ell} = 1 - w_{\alpha,\beta,i,j}, \qquad t_{\alpha,\beta,i,j,\ell} \in \{0,1\},\ \ell = 0,\ldots,7 \tag{37}$$

$$w_{\alpha,\beta,i,j} \cdot (c_{i,j} - \beta) \geq 0, \qquad w_{\alpha,\beta,i,j} \cdot (\beta - c_{i,j}) \geq 0 \tag{38}$$

$$w_{\alpha,\beta,i,j} \cdot (c_{i\oplus\alpha,j\oplus\alpha} - \beta) \geq 0, \qquad w_{\alpha,\beta,i,j} \cdot (\beta - c_{i,j,\alpha}) \geq 0 \tag{39}$$

$$t_{\alpha,\beta,i,j,0} \cdot (c_{i,j} - \beta) = 0, \qquad t_{\alpha,\beta,i,j,0} \cdot (c_{i\oplus\alpha,j\oplus\alpha} - \beta - 1) \geq 0 \tag{40}$$

$$t_{\alpha,\beta,i,j,1} \cdot (c_{i,j} - \beta) = 0, \qquad t_{\alpha,\beta,i,j,1} \cdot (c_{i\oplus\alpha,j\oplus\alpha} - \beta + 1) \leq 0 \tag{41}$$

$$t_{\alpha,\beta,i,j,2} \cdot (c_{i\oplus\alpha,j\oplus\alpha} - \beta) = 0, \qquad t_{\alpha,\beta,i,j,2} \cdot (c_{i,j} - \beta - 1) \geq 0 \tag{42}$$

$$t_{\alpha,\beta,i,j,3} \cdot (c_{i\oplus\alpha,j\oplus\alpha} - \beta) = 0, \qquad t_{\alpha,\beta,i,j,3} \cdot (c_{i,j} - \beta + 1) \leq 0 \tag{43}$$

$$t_{\alpha,\beta,i,j,4} \cdot (c_{i,j} - \beta - 1) \geq 0, \qquad t_{\alpha,\beta,i,j,4} \cdot (c_{i\oplus\alpha,j\oplus\alpha} - \beta - 1) \geq 0 \tag{44}$$

$$t_{\alpha,\beta,i,j,5} \cdot (c_{i,j} - \beta - 1) \geq 0, \qquad t_{\alpha,\beta,i,j,5} \cdot (c_{i\oplus\alpha,j\oplus\alpha} - \beta + 1) \leq 0 \tag{45}$$

$$t_{\alpha,\beta,i,j,6} \cdot (c_{i,j} - \beta + 1) \leq 0, \qquad t_{\alpha,\beta,i,j,6} \cdot (c_{i\oplus\alpha,j\oplus\alpha} - \beta - 1) \geq 0 \tag{46}$$

$$t_{\alpha,\beta,i,j,7} \cdot (c_{i,j} - \beta + 1) \leq 0, \qquad t_{\alpha,\beta,i,j,7} \cdot (c_{i\oplus\alpha,j\oplus\alpha} - \beta + 1) \leq 0 \tag{47}$$

   Observe that $w_{\alpha,\beta,i,j} = 1$ implies $t_{\alpha,\beta,i,j,\ell} = 0$ for all ℓ. Then (38) and (39) enforce $c_{i,j} = \beta$ and $c_{i\oplus\alpha,j\oplus\alpha} = \beta$. At the same time (40) to (47) also hold. On the other hand, $w_{\alpha,\beta,i,j} = 0$ means only one of $t_{\alpha,\beta,i,j,\ell}$ equals 1 for some ℓ. For instance, if $t_{\alpha,\beta,i,j,2} = 1$, then we only have $c_{i\oplus\alpha,j\oplus\alpha} = \beta$ and $c_{i,j} \geq \beta + 1$.

7.2 Mapping Solutions to Entries of BCT

For a given (α,β), there are $\mathsf{BCT}(\alpha,\beta)$ pairs (i,j) satisfying (34). When we consider solution pairs from $\mathcal{Y}$, then we omit the symmetric solutions. Therefore, the number of pairs $(i,j) \in \mathcal{Y}$ such that (35) holds should sum up to $\frac{\mathsf{BCT}(\alpha,\beta)}{2}$. As we have $w_{\alpha,\beta,i,j} = 1$

if and only if (35) holds, then we have an important constraint that directly links the number of solutions to BCT values as follows:

$$\sum_{(i,j)\in\mathcal{Y}} w_{\alpha,\beta,i,j} = \frac{\mathsf{BCT}(\alpha,\beta)}{2}. \tag{48}$$

We repeat the procedure as discussed above for all (α, β) and present it in Algo. 7.

Algorithm 7: model_map_solutions_to_bct

Input: n, $\mathcal{M}$ and BCT
Output: $\mathcal{M}$

1 $\mathcal{Y} = \{(i,j) : i < j\}$
2 $\{c_{i,j}\}_{(i,j)\in\mathcal{Y}}$ such that $c_{i,j} \in \{1, \ldots, 2^n - 1\}$ ▷ Variables
3 **for** *each* $(i,j) \in \mathcal{Y}$ **do**
4 $\quad c_{i,j,k},\ d_{i,j,k} \in \{0,1\},$ for $k = 0, \ldots, n-1$
5 $\quad y_{i,k} + y_{j,k} + c_{i,j,k} = 2 \cdot d_{i,j,k},$ for $k = 0, \ldots, n-1$
6 $\quad c_{i,j} = \sum_{k=0}^{n-1} 2^k \cdot c_{i,j,k}$
7 **end**
8 **for** $\alpha = 1$ *to* $2^n - 1$ **do**
9 $\quad$ **for** $\beta = 1$ *to* $2^n - 1$ **do**
10 $\quad\quad w_{\alpha,\beta,i,j} \in \{0,1\}$, for $(i,j) \in \mathcal{Y}$ ▷ Variables
11 $\quad\quad$ **for** $i = 0$ *to* $2^n - 1$ **do**
12 $\quad\quad\quad$ **for** $j = i + 1$ *to* $2^n - 1$ **do**
13 $\quad\quad\quad\quad t_{\alpha,\beta,i,j,\ell},\quad$ for $\ell = 0, \ldots, 7$ ▷ Variables
14 $\quad\quad\quad\quad$ Add constraints of (37) to (47)
15 $\quad\quad\quad$ **end**
16 $\quad\quad$ **end**
17 $\quad\quad \sum_{(i,j)\in\mathcal{Y}} w_{\alpha,\beta,i,j} = \frac{\mathsf{BCT}(\alpha,\beta)}{2}$ ▷ BCT constraint
18 $\quad$ **end**
19 **end**
20 **return** $\mathcal{M}$

7.3 Modeling (partial) BCT to S-box

Recovering S-box from a full BCT. We now have all the steps for modeling the BCT, the next task is to check the feasibility of the model. An infeasible model means the given BCT is not valid, i.e., it does not correspond to any S-box. On the other hand, a feasible model will return an S-box. We present the complete algorithm to recover an S-box from a full BCT in Algo. 8.

Recovering S-box from a partial BCT. Given a partial BCT with rows corresponding to input differences $\{\alpha_1, \alpha_2, \ldots, \alpha_l\} \subset \{1, 2, \ldots, 2^{n-1}\}$, we model it by replacing Line 8 of Algo. 7 with

$$\text{for } \alpha \in \{\alpha_1, \alpha_2, \ldots, \alpha_l\}.$$

The rest of the steps are exactly same as Algo. 8.

7.4 Experimental Validation

Similar to the previous experiments on DDT, LAT and DLCT, we checked the correctness of Algo. 8 using full BCT of different S-boxes and partial BCT of the 4-bit inverse S-box.

Algorithm 8: model_bct_to_sbox

Input: n, BCT and empty $\mathcal{M}$
Output: S-box $= [y_0, y_1, \ldots, y_{2^n-1}]$ or given BCT is invalid

```
M ← model_sbox_with_permutation_matrix(n, M)
M ← model_map_solutions_to_bct(n, M, BCT)
if M is feasible then
    return [y_0, y_1, ..., y_{2^n-1}]
end
else
    return BCT is invalid
end
```

We report the timings and model sizes in Table 11 and Table 12 using only Gurobi. We did not include OR-Tools as our model has quadratic constraints and we are not aware of how this solver handles such constraints.[2]

Table 11: Model size and timings for BCT to S-box. The symbol '-' denotes that no solution is obtained with a time limit of one hour.

BCT of	Model details				Time (sec.)
	Variables		Constraints		
	Integer	Binary	Linear	Quadratic	Gurobi
3-bit χ	36	12604	1565	27440	1.12
4-bit Inverse	136	244280	27889	540000	48.18
PRESENT	136	244280	27889	540000	35.68
GIFT	136	244280	27889	540000	47.1
SKINNY	136	244280	27889	540000	55.97
ASCON	528	4296048	480721	9533120	-
Keccak	528	4296048	480721	9533120	-
5-bit Inverse	528	4296048	480721	9533120	-

Table 12: Reconstructing S-box from the first l rows of BCT of 4-bit inverse S-box with Gurobi. The minimum number of rows up to which the properties of obtained S-boxes remain identical is highlighted in blue. First row corresponds to the full BCT.

# rows (l)	Time (s)	Boomerang unif.	Differential-linear unif.	Diff. unif.	Linearity
16	48.18	6	4	4	8
12	112	6	4	4	8
8	17	6	4	4	8
6	13	6	4	4	8
5	2740	6	4	4	8
4	436	10	8	6	12

From Table 11, it is evident that the model size is too large even for BCT's of 4 and 5-bit S-boxes. Because of the same reason, the solver could not produce a solution for BCT's of 5-bit S-boxes with a time limit of one hour.

[2] In fact one could do this by using in-built APIs, but then we will be running two different models on two different solvers which defeats the purpose of comparison.

8 Applications

In this section, we present several applications of our proposed techniques in the context of searching for cryptographically significant S-boxes and Boolean functions.

First, we show how to tweak our MILP models to search for S-boxes with a given cryptographic property such as differential uniformity, linearity, differential-linear uniformity or boomerang uniformity. Most notably, we present the first novel and generic MILP-based algorithm to search for δ-differential uniform S-boxes. We also discuss how to extend these techniques to recover a Boolean function from a given Walsh spectrum.

Second, knowing that $[Y_0, \ldots, Y_{2^n-1}]$ is a solution of the MILP problem based on a cryptographic property, say $\mathcal{P}$, that is $[Y_0, \ldots, Y_{2^n-1}]$ is an S-box that satisfies $\mathcal{P}$, we start with it as the initial solution to MILP, introduce a random objective function in MILP and then optimize the model for the maximum or minimum value. We obtain a new S-box for each objective value which is different from the initial solution's objective. We call this heuristic as the *Optimistic MILP objective*. In particular, we highlight the impact of this technique by obtaining 6-bit S-boxes which are CCZ-inequivalent to the inverse S-box and have the same differential uniformity, which is 4.

8.1 Find an S-box with a Given Cryptographic Property

We discuss how to obtain an S-box with a given value of differential uniformity, linearity, differential-linear uniformity or boomerang uniformity. This method is different from the previous one as in this case we do not have a specific DDT, LAT, DLCT and BCT, instead we use the given value as the input. This value acts as an upper bound. For instance, if we are looking for 4 differential uniform S-boxes, it can produce both 2 and 4 differential uniform S-boxes.

1. **Differential uniformity**
   We wish to find an S-box with differential uniformity δ. To achieve this goal, we first have a close look at Algo. 4. There is only one step

$$\sum_{(i,j)\in\mathcal{X}_n(\alpha)} w_{i,j,k} = \frac{\mathsf{DDT}[\alpha][k]}{2}, \tag{49}$$

   in procedure model_map_output_difference_to_ddt (Line 8 of Algo. 3) which connects the entire MILP modeling to the entries of DDT. Replacing (49) by

$$\sum_{(i,j)\in\mathcal{X}_n(\alpha)} w_{i,j,k} \leq \frac{\delta}{2}, \tag{50}$$

   is sufficient to get a new MILP model whose feasible solution is an S-box with at most δ-differential uniformity. Notably, for $\delta = 2$, the model will search for APN permutations. In summary, we modify Algo. 4 as follows.

   (a) Replace the input DDT by differential uniformity value δ.

   (b) Replace Line 8 of Algo. 3 with

$$\sum_{(i,j)\in\mathcal{X}_n(\alpha)} w_{i,j,k} \leq \frac{\delta}{2}\,.$$

2. **Linearity**
   Let λ and γ be the input and output masks, respectively. Let $\mathcal{L}$ be the desired linearity. Then for all nonzero λ and γ, (4) should hold. Now, the only link between LAT entries and MILP modeling in Algo. 5 (Line 16) is via (25) which is given by

$$\sum_{i=0}^{2^n-1} w_{\lambda,\gamma,i} = \mathsf{LAT}(\lambda,\gamma) + 2^{n-1}, \quad w_{\lambda,\gamma,i} \in \{0,1\} \ . \tag{51}$$

Thus, by ensuring sum of $w_{\lambda,\gamma,i}$ to be at most $(2^{n-1} + \frac{\mathcal{L}}{2})$ and at least $(2^{n-1} - \frac{\mathcal{L}}{2})$, we construct a MILP model $\mathcal{M}$ whose feasible solution is an S-box with linearity at most $\mathcal{L}$. In summary, the new model is exactly the same as [Algo. 5], but with the two modifications given below.

(a) Replace the input LAT by linearity $\mathcal{L}$.

(b) Replace Line 16 with

$$\left(2^{n-1} - \frac{\mathcal{L}}{2}\right) \leq \sum_{i=0}^{2^n-1} w_{\lambda,\gamma,i} \leq \left(2^{n-1} + \frac{\mathcal{L}}{2}\right) .$$

3. **Differential-linear uniformity**

We wish to find an S-box with d-differential-linear uniformity. Suppose α is the input difference and λ is the output mask. Then as per (5), for all nonzero α and $\lambda = 0, \ldots, 2^n - 1$, the target S-box should satisfy

$$|\#\{x \mid \lambda \cdot x = \lambda \cdot S(x \oplus \alpha)\} - 2^{n-1}| \leq d \ . \tag{52}$$

Therefore,

$$\begin{cases} \#\{x \mid \lambda \cdot x = \lambda \cdot S(x \oplus \alpha)\} \leq 2^{n-1} + d \\ \#\{x \mid \lambda \cdot x = \lambda \cdot S(x \oplus \alpha)\} \geq 2^{n-1} - d. \end{cases} \tag{53}$$

Now, consider Line 14 of Algo. 6 which maps solutions of equations of the form $\lambda \cdot x = \lambda \cdot S(x \oplus \alpha)$ to the entries of DLCT via

$$\sum_{(i,j)\in\mathcal{X}_n(\alpha)} w_{i,j,\lambda} = \frac{\mathsf{DLCT}(\alpha,\lambda) + 2^{n-1}}{2} \ .$$

By ensuring that the sum of $w_{i,j,\lambda}$ is at most $2^{n-2} + \frac{d}{2}$ and at least $2^{n-2} - \frac{d}{2}$ (division by 2 to exclude symmetric solution pairs), we obtain a new MILP model whose feasible solution is an S-box with differential-linear uniformity at most d. The model is similar to Algo. 6 with the following two changes.

(a) Replace the input DLCT by differential-linear uniformity d.

(b) Replace Line 14 with

$$2^{n-2} - \frac{d}{2} \leq \sum_{(i,j)\in\mathcal{X}_n(\alpha)} w_{i,j,\lambda} \leq \ 2^{n-2} + \frac{d}{2} \ .$$

4. **Boomerang uniformity**

Suppose our goal is to find an S-box with boomerang uniformity μ. For nonzero α and β, the MILP constraints in Algo. 7 (Line 17) are linked to $\mathsf{BCT}(\alpha,\beta)$ via

$$\sum_{(i,j)\in\mathcal{Y}} w_{\alpha,\beta,i,j} = \frac{\mathsf{BCT}(\alpha,\beta)}{2} \ .$$

Thus, simply changing the above by

$$\sum_{(i,j)\in\mathcal{Y}} w_{\alpha,\beta,i,j} \leq \frac{\mu}{2}$$

gives a new MILP model whose feasible solution is an S-box with boomerang uniformity at most μ. To summarize, the new model is similar to Algo. 8 with the following changes.

(a) Replace the input BCT by boomerang uniformity μ.

(b) Replace Line 17 of Algo. 7 with

$$\sum_{(i,j)\in\mathcal{Y}} w_{\alpha,\beta,i,j} \leq \frac{\mu}{2} \ .$$

Experimental validation. To verify the correctness, we run all MILP models with small parameters (3 and 4-bit S-boxes) with several values of differential uniformity, linearity, differential-linear uniformity and boomerang uniformity. Our models successfully find S-boxes with the desired property.

8.2 Recovering Boolean functions from their Walsh Spectrum

Walsh transform of a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ at $\lambda \in \mathbb{F}_2^n$ is given by

$$W_f(\lambda) = \sum_{x\in\mathbb{F}_2^n} (-1)^{f(x)\oplus\lambda\cdot x}. \tag{54}$$

Walsh spectrum of f is the multi-set $\{W_f(\lambda)\}_{\lambda\in\mathbb{F}_2^n}$. Walsh spectrum is used to analyse various cryptographic properties of a Boolean function such as nonlinearity and correlation immunity. Since nonlinearity of a Boolean function is directly related to the linearity of an S-box, so it is worth mentioning the reconstruction of a Boolean function from a given Walsh spectrum.

We can rewrite (54) as

$$\begin{aligned} W_f(\lambda) &= \#\{x : f(x) = \lambda\cdot x\} - \#\{x : f(x) \neq \lambda\cdot x\} \\ &= 2\cdot(\#\{x : f(x) = \lambda\cdot x\} - 2^{n-1}). \end{aligned} \tag{55}$$

It is clear from Def. 2 that $\mathsf{LAT}_S(\lambda,\gamma)$ is actually twice the Walsh transform value of the component Boolean function $\gamma\cdot S$ of the S-box S. Then Walsh spectrum of the Boolean function $\gamma\cdot S$ is the multi-set $\{\mathsf{LAT}_S(\lambda,\gamma) : \lambda\in\mathbb{F}_2^n\}$, which is basically the column corresponding to γ in the LAT of S.

Suppose $[f_0,\ldots,f_{2^n-1}]$ is the truth table of f. Following (20), we can rewrite (55) as

$$\frac{W_f(\lambda)}{2} + 2^{n-1} = \#\{\vec{i}\in\mathbb{F}_2^n : \vec{\lambda}\cdot\vec{i} = f_i\}. \tag{56}$$

Thus, as done in the modeling of LAT, we can take similar approach here to find the $[f_0,\ldots,f_{2^n-1}]$ from the given Walsh spectrum $\{W_f(\lambda)\}_{\lambda\in\mathbb{F}_2^n}$.

Recovering a Boolean function with a given nonlinearity. For an n-variable Boolean function f, nonlinearity is defined as

$$nl_f = 2^{n-1} - \frac{1}{2}\cdot\max_{\lambda}|W_f(\lambda)|.$$

Therefore, we can easily target a nonlinearity and can find a corresponding Boolean function (if such nonlinearity value is valid). More specifically, if we we want to search for a Boolean function f with nonlinearity nl, then we need to consider Walsh spectrum values $\{W_f(\lambda) : \lambda \in \mathbb{F}_2^n$ such that $|W_f(\lambda)| \leq 2^n - 2 \cdot nl\}$ and use this bound in 56 and model it for all $\lambda \in \mathbb{F}_2^n$.

8.3 Optimistic MILP Objectives

We propose *Optimistic MILP objective* heuristic with the goal to find multiple S-boxes having the same cryptographic property $\mathcal{P}$. Our idea is to introduce a random objective function in the model and then optimize it for the maximum or minimum value. For each objective value which is different from the initial solution's objective value, we obtain a new S-box. We then check the equivalence of these S-boxes with the original one using the SboxU tool [Leo]. We emphasize that we check the equivalence only on the S-boxes obtained from Gurobi as output. This is because there are no equivalence conditions imposed in the model itself (see Sec. 9.3 for more discussion).

8.3.1 Modeling with Random MILP Objective

Suppose $\mathcal{M}$ is the MILP model and the target S-box is given by the integer variable array $[y_0, \ldots, y_{2^n-1}]$. It is known that the S-box $[Y_0, \ldots, Y_{2^n-1}]$ has the cryptographic property $\mathcal{P}$. We update $\mathcal{M}$ in two steps.

1. **Starting solution**
   We force the solver to start with initial solution as $[Y_0, \ldots, Y_{2^n-1}]$. In Gurobi C++ interface, this can be done as follows.

$$y_i.\mathsf{set}(\mathsf{GRB\_DoubleAttr\_Start}, Y_i), \quad \text{for } i = 0, \ldots, 2^n - 1 \ . \tag{57}$$

2. **Random objective**
   Suppose $[b_0, \ldots, b_{2^n-1}]$ is an array of 2^n random bits. We introduce a random linear objective in terms of $[y_0, \ldots, y_{2^n-1}]$ to $\mathcal{M}$. It is defined as

$$\max \Big(\sum_{i=0}^{2^n-1} r_i \cdot y_i \Big) \tag{58}$$

   where

$$r_i = \begin{cases} 1 & \text{if } b_i = 0 \ , \\ -1 & \text{if } b_i = 1 \ . \end{cases}$$

   Different objective values give different solutions for $[y_0, \ldots, y_{2^n-1}]$ as we will see next.

8.3.2 CCZ-inequivalent S-boxes with a given Differential Uniformity

We demonstrate the impact of this technique to find a 4-differential uniform 6-bit S-box. As we know the 6-bit inverse S-box ($x \mapsto x^{-1}$) is 4-differential uniform, so our goal is to find another 4-differential uniform 6-bit S-box. Let the 64-bit random bit-string $b_0 \cdots b_{63}$ is given by

$$0101101010101110101000101111010100000111000101010011111111000111 \ .$$

We now proceed as follows.

1. Create an MILP model as given in Sec. 4.

2. Update the model by adding 6-bit inverse S-box as the initial solution.

3. Update the model by adding the objective function in (58) using the above bit-string $b_0 \cdots b_{63}$.

Table 13: 6-bit S-boxes with differential uniformity 4

S.no	Obj. value	S-box
S_0 (Inverse S-box)	-94	0, 1, 45, 54, 59, 18, 27, 30, 48, 10, 9, 49, 32, 62, 15, 14, 24, 51, 5, 58, 41, 56, 53, 35, 16, 50, 31, 6, 42, 38, 7, 26, 12, 63, 52, 23, 47, 61, 29, 43, 57, 20, 28, 39, 55, 2, 60, 36, 8, 11, 25, 17, 34, 22, 3, 44, 21, 40, 19, 4, 46, 37, 13, 33
S_1	-88	1, 0, 44, 55, 58, 19, 26, 31, 49, 11, 8, 48, 33, 63, 14, 15, 25, 50, 4, 59, 40, 57, 52, 34, 17, 51, 30, 7, 43, 39, 6, 27, 13, 62, 53, 22, 46, 60, 28, 42, 56, 21, 29, 38, 54, 3, 61, 37, 9, 10, 24, 16, 35, 23, 2, 45, 20, 41, 18, 5, 47, 36, 12, 32
S_2	-80	1, 0, 44, 55, 58, 19, 26, 31, 49, 11, 8, 48, 33, 63, 10, 15, 25, 50, 4, 59, 40, 57, 52, 34, 17, 51, 30, 7, 43, 39, 6, 27, 13, 62, 53, 22, 46, 60, 28, 42, 56, 21, 29, 38, 54, 3, 61, 37, 9, 14, 24, 16, 35, 23, 2, 45, 20, 41, 18, 5, 47, 36, 12, 32
S_3	-26	1, 0, 44, 55, 58, 19, 26, 31, 49, 42, 8, 48, 33, 63, 14, 15, 25, 50, 4, 59, 40, 57, 52, 34, 17, 51, 30, 7, 43, 39, 6, 27, 13, 62, 53, 22, 46, 60, 28, 11, 56, 21, 29, 38, 54, 3, 61, 37, 9, 10, 24, 16, 35, 23, 2, 45, 20, 41, 18, 5, 47, 36, 12, 32
S_4	-20	5, 4, 40, 51, 62, 23, 30, 27, 53, 15, 12, 52, 37, 59, 10, 11, 29, 54, 0, 63, 44, 61, 48, 38, 21, 17, 26, 3, 47, 35, 2, 31, 9, 58, 49, 18, 42, 56, 24, 46, 60, 55, 25, 34, 50, 7, 57, 33, 13, 14, 28, 20, 39, 19, 6, 41, 16, 45, 22, 1, 43, 32, 8, 36
S_5	-16	7, 4, 40, 51, 62, 23, 30, 27, 53, 15, 12, 52, 37, 59, 10, 11, 29, 54, 0, 63, 44, 61, 48, 38, 21, 17, 26, 3, 47, 35, 2, 31, 9, 58, 49, 18, 42, 56, 24, 46, 60, 55, 25, 34, 50, 5, 57, 33, 13, 14, 28, 20, 39, 19, 6, 41, 16, 45, 22, 1, 43, 32, 8, 36
S_6	-12	1, 0, 44, 55, 58, 19, 26, 31, 49, 12, 8, 48, 33, 63, 14, 15, 25, 50, 4, 59, 40, 57, 52, 34, 17, 51, 30, 7, 43, 39, 6, 27, 13, 62, 53, 22, 46, 60, 28, 11, 56, 21, 29, 38, 54, 3, 61, 37, 9, 10, 24, 16, 35, 23, 2, 45, 20, 41, 18, 42, 47, 36, 5, 32
S_7	48	7, 4, 40, 51, 62, 23, 30, 27, 53, 15, 12, 52, 37, 59, 10, 11, 29, 54, 0, 63, 44, 61, 48, 38, 21, 17, 26, 5, 47, 35, 2, 31, 9, 58, 49, 18, 42, 56, 24, 14, 60, 55, 25, 34, 50, 3, 57, 33, 13, 46, 28, 20, 39, 19, 6, 41, 16, 45, 22, 1, 43, 32, 8, 36

On solving the model, we find several solutions with different values of the objective function as shown in Table 13.[3] Note that in Table 13, the inverse S-box denoted as S_0 has objective value -94. We have following observations on other S-boxes mentioned in the same table.

1. S-boxes S_0 and S_1 are CCZ-equivalent to each other. However, they are CCZ-inequivalent to $\mathsf{S}_2, \mathsf{S}_3, \mathsf{S}_4, \mathsf{S}_5, \mathsf{S}_6$ and S_7.

2. S-boxes S_2, S_3 and S_4 are CCZ-equivalent to each other, but in-equivalent to S-boxes $\mathsf{S}_5, \mathsf{S}_6$ and S_7.

[3]This is not the exhaustive list of S-boxes that are obtained.

3. S-boxes S_5, S_6 and S_7 are CCZ-inequivalent.

In the end, we find five CCZ-inequivalent S-boxes (including the inverse) of 6-bit S-boxes, i.e., $\{\mathsf{S}_0, \mathsf{S}_2, \mathsf{S}_5, \mathsf{S}_6, \mathsf{S}_7\}$ which have differential uniformity 4. Moreover, all these S-boxes are also CCZ-inequivalent to the 4-differential uniform 6-bit S-box of [Ras22].

Experiments on 6-bit APN. We did similar experiments with APN6 to find if there exists another CCZ-inequivalent S-box with differential uniformity 2. However, all the S-boxes that we found turned out to be XOR-equivalent.

Remark 2. We could change (58) to find the minimum value or choose completely different objective function. Also, rather than differential property, one may consider other properties such as linearity, differential-linearity or boomerang uniformity to find more S-boxes.

9 Discussion and Comparison with Related Works

In this section, we discuss the advantages and limitations of our approach, and compare with the related works.

9.1 Application of Permutation Matrix

The permutation matrices are well-known object in mathematics. They are used in literature in several contexts, for e.g., solving traveling salesman problem [DFJ54], finding efficient implementation of linear layers [BKD21] or alternative key-schedules of AES [BDF24]. In this work, we propose the application of permutation matrices in the context of S-boxes search and reconstruction problem related to cryptographic tables for the first time.

9.2 Efficiency Comparison

Though our current approach is limited to 6-bit S-boxes for DDT and LAT, 5-bit S-boxes for DLCT and 4-bit S-boxes for BCT, the biggest advantage is that it is highly effective with partial cryptographic tables. For instance, given the first 17 rows of the DDT of APN6, we could recover the APN S-boxes CCZ-equivalent to APN6 in seconds (see Table 5). For the same setting, previous works [BCJS19, DH19, TBP20] can find the first 17 values of S-box in seconds, but then require an exhaustive search over the remaining 47 elements. All in all, the existing works [BCJS19, DH19, TBP20] though efficient for up to 8-bit S-boxes do not work with partial tables.

In case of [LMC+22], all the reported results consider up to 5-bit S-boxes only. Since their tool is not publicly available, it is difficult to say whether their approach works for 6-bit S-boxes or not. In our case, we have shown that reconstructing an S-box from the DDT and LAT of 6-bit S-boxes can be done in seconds (see Tables 4 and 7). When it comes to comparing the exact performance of our approach with that of [LMC+22], we have carefully studied their reported performance results and based on that we understand that our approach is much more efficient than theirs. We justify in Table 14 why our method outperforms theirs in terms of performance.

Table 14: Time to obtain all S-boxes from the DDT of PRESENT and Keccak.

DDT of	# S-boxes	Time	
		Ours	[LMC+22]
PRESENT	256	20 s	10 min.
Keccak	1024	36 s	7.5 hr

Clearly, Table 14 establishes our method as the winner over [LMC+22] as far as PRESENT and Keccak are concerned. For further clarity, we would also like to note the following.

1. We run our experiments on AMD EPYC 7763 64-Core CPU (2.45 GHz clock) using 8 threads. The authors in [LMC+22] used AMD EPYC 7302 16-Core CPU (3.0 GHz clock) with 8 threads. As far as we understand, for the same number of threads, the two platforms are almost similar in terms of efficiency.

2. We only include PRESENT and Keccak in the table as [LMC+22] reported the timing for the DDT of these two ciphers only.

3. We chose Gurobi 10 as the MILP solver in this regard.

Our MILP models are generic and can be plugged into any MILP solver. We used Gurobi and OR-Tools in our experiments where we leverage unique advantages that each solver offers. For instance, some efficiency advantages in our approach have been attributed by Gurobi quite uniquely: for BCT, we directly can apply the Mixed Integer Quadratic Constraint Programming; for optimistic MILP objective heuristics, we use the inbuilt APIs for finding other solutions.

9.3 Classification of S-boxes

One important contribution of our work to the state-of-the-art is the introduction of optimistic MILP objective heuristics, which addresses CCZ-inequivalence to some extent in the context of the reconstruction problem. However, our technique at the moment can not classify S-boxes – we only check equivalence/inequivalence outside the MILP model. For instance, 4-differential uniform S-boxes in Table 13 (obtained via Gurobi) are checked for CCZ-equivalence using the SboxU tool. We found some of the S-boxes generated by our heuristic (Optimistic MILP objective) are CCZ-inequivalent to the initial S-box and have the similar cryptographic properties. We have not studied how this heuristic is helping the generation of CCZ-inequivalent S-boxes in this work as we think this question needs a separate attention as a whole. On the other hand, we also believe that finding an answer to the problem of applying MILP in classifying S-boxes needs separate attention for a thorough investigation, where our results could serve as a stepping stone. In the following, we shed some initial insights on this problem.

While generating S-boxes for a given cryptographic property, we obtain S-boxes which are affine equivalent. However, if we fix some output value, then at least some affine equivalent S-boxes will not be generated. For instance, if $S(0) \neq 0$, then the S-box S' given by $S'(x) = S(x) \oplus S(0)$ is affine equivalent to S. However, if we add the constraint $S(0) = 0$, then such equivalent S-boxes will not be repeated.

The second strategy could be searching for affine inequivalent S-boxes by incorporating frequencies of DDT in the model. For example, given that S is 4-differential with #2's and #4's being $\mathcal{N}_2$ and $\mathcal{N}_4$ in the DDT. Then, we can search for S' by adding constraints that #2's and #4's are not $\mathcal{N}_2$ and $\mathcal{N}_4$, respectively.

10 Conclusions

In this paper, we presented the novel applications of mixed integer linear programming and mixed integer quadratic constraint programming to solve open problems related to S-boxes. Our major breakthrough is that we have been able to reconstruct S-boxes from partial cryptographic tables for the first time in the literature. This also for the first time shows the applicability of MILP techniques for the S-box reconstruction problem. Our

second advancement is the generic MILP models which can search for S-boxes with a desired cryptographic property such as differential uniformity, linearity, differential-linear uniformity and boomerang uniformity. We also showed how the same approach could be extended to reconstruct S-boxes from a given Walsh spectrum and consequently find Boolean function with a given nonlinearity. Finally, we introduced Optimistic MILP objective heuristic which could successfully find 4-differential uniform S-boxes which are CCZ-inequivalent to the 6-bit inverse S-box.

We believe that similar to the progress achieved over the decade in differential and linear cryptanalysis with MILP, the presented techniques will also advance with time. Hence we expect to see more applications of our ideas in the future. We list down some of the interesting problems for future research.

1. This work considered partial tables with first few rows, but it is interesting to answer whether the technique can be extended to tables consisting of both partial rows and partial columns. Is it possible to theoretically guarantee an optimal choice of partial rows and partial columns for such reconstruction problems ?

2. Analyzing optimistic MILP objective heuristic to search for new classes of S-boxes. One intriguing problem is to investigate this heuristic to find whether there exists a 6-bit APN permutation which is CCZ-inequivalent to the Dillon et al.'s APN permutation [BDMW10].

3. Incorporating inequivalence conditions into MILP modeling of S-box search.

4. An independent research interest branches out from our detailed comparative experimental study of Gurobi and OR-Tools – Gurobi performs better in certain problems (LAT problem), while OR-Tools excels in others (DDT and DLCT problems). It will be interesting to extend the study of comparing different MILP solvers in the context of S-box reconstruction problems.

Acknowledgments

The authors would like to thank the reviewers of ToSC 2023/2024 for providing us with insightful feedback that helped in significantly improving the quality of the paper. Furthermore, Sumanta Sarkar acknowledges the research grants from Engineering and Physical Sciences Research Council, EP/T014784/1 and EP/X036669/1.

References

[AST+17] Ahmed Abdelkhalek, Yu Sasaki, Yosuke Todo, Mohamed Tolba, and Amr M. Youssef. MILP Modeling for (Large) S-boxes to Optimize Probability of Differential Characteristics. *IACR Trans. Symmetric Cryptol.*, 2017(4):99–129, 2017.

[BBDK18] Achiya Bar-On, Eli Biham, Orr Dunkelman, and Nathan Keller. Efficient Slide Attacks. *J. Cryptol.*, 31(3):641–670, 2018.

[BC20] Christina Boura and Daniel Coggia. Efficient MILP Modelings for Sboxes and Linear Layers of SPN ciphers. *IACR Trans. Symmetric Cryptol.*, 2020(3):327–361, 2020.

[BCJS19] Christina Boura, Anne Canteaut, Jérémy Jean, and Valentin Suder. Two notions of differential equivalence on Sboxes. *Des. Codes Cryptogr.*, 87(2-3):185–202, 2019.

[BDF24] Christina Boura, Patrick Derbez, and Margot Funk. Alternative key schedules for the AES. In Christina Pöpper and Lejla Batina, editors, *Applied Cryptography and Network Security - 22nd International Conference, ACNS 2024, Abu Dhabi, United Arab Emirates, March 5-8, 2024, Proceedings, Part II*, volume 14584 of *Lecture Notes in Computer Science*, pages 485–506. Springer, 2024.

[BDG+21] Zhenzhen Bao, Xiaoyang Dong, Jian Guo, Zheng Li, Danping Shi, Siwei Sun, and Xiaoyun Wang. Automatic Search of Meet-in-the-Middle Preimage Attacks on AES-like Hashing. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 771–804. Springer, 2021.

[BDKW19] Achiya Bar-On, Orr Dunkelman, Nathan Keller, and Ariel Weizman. DLCT: A New Tool for Differential-Linear Cryptanalysis. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 313–342. Springer, 2019.

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

[BGG+23] Emanuele Bellini, David Gérault, Juan Grados, Rusydi H. Makarim, and Thomas Peyrin. Fully Automated Differential-Linear Attacks Against ARX Ciphers. In Mike Rosulek, editor, *Topics in Cryptology - CT-RSA 2023 - Cryptographers' Track at the RSA Conference 2023, San Francisco, CA, USA, April 24-27, 2023, Proceedings*, volume 13871 of *Lecture Notes in Computer Science*, pages 252–276. Springer, 2023.

[BGLS19] Zhenzhen Bao, Jian Guo, San Ling, and Yu Sasaki. PEIGEN - a Platform for Evaluation, Implementation, and Generation of S-boxes. *IACR Trans. Symmetric Cryptol.*, 2019(1):330–394, 2019.

[BKD21] Anubhab Baksi, Banashri Karmakar, and Vishnu Asutosh Dasu. POSTER: optimizing device implementation of linear layers with automated tools. In Jianying Zhou, Chuadhry Mujeeb Ahmed, Lejla Batina, Sudipta Chattopadhyay, Olga Gadyatskaya, Chenglu Jin, Jingqiang Lin, Eleonora Losiouk, Bo Luo, Suryadipta Majumdar, Mihalis Maniatakos, Daisuke Mashima, Weizhi Meng, Stjepan Picek, Masaki Shimaoka, Chunhua Su, and Cong Wang, editors, *Applied Cryptography and Network Security Workshops - ACNS 2021 Satellite Workshops, AIBlock, AIHWS, AIoTS, CIMSS, Cloud S&P, SCI, SecMT, and SiMLA, Kamakura, Japan, June 21-24, 2021, Proceedings*, volume 12809 of *Lecture Notes in Computer Science*, pages 500–504. Springer, 2021.

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

[BS91] Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In *Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '90, pages 2–21, London, UK, UK, 1991. Springer-Verlag.

[BV14] Alex Biryukov and Vesselin Velichkov. Automatic search for differential trails in arx ciphers. In Josh Benaloh, editor, *Topics in Cryptology – CT-RSA 2014*, pages 227–250, Cham, 2014. Springer International Publishing.

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

[CHP+17] Carlos Cid, Tao Huang, Thomas Peyrin, Yu Sasaki, and Ling Song. A Security Analysis of Deoxys and its Internal Tweakable Block Ciphers. *IACR Trans. Symmetric Cryptol.*, 2017(3):73–107, 2017.

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

[CJF+16] Tingting Cui, Keting Jia, Kai Fu, Shiyao Chen, and Meiqin Wang. New Automatic Search Tool for Impossible Differentials and Zero-Correlation Linear Approximations. *IACR Cryptol. ePrint Arch.*, page 689, 2016.

[CKL+19] Anne Canteaut, Lukas Kölsch, Chao Li, Chunlei Li, Kangquan Li, Longjiang Qu, and Friedrich Wiemer. On the Differential-Linear Connectivity Table of Vectorial Boolean Functions. *CoRR*, abs/1908.07445, 2019.

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

[DDV20] Stéphanie Delaune, Patrick Derbez, and Mathieu Vavrille. Catching the Fastest Boomerangs Application to SKINNY. *IACR Trans. Symmetric Cryptol.*, 2020(4):104–129, 2020.

[DFJ54] George B. Dantzig, D. Ray Fulkerson, and Selmer M. Johnson. Solution of a large-scale traveling-salesman problem. *Oper. Res.*, 2(4):393–410, 1954.

[DGV94] Joan Daemen, René Govaerts, and Joos Vandewalle. Correlation Matrices. In Bart Preneel, editor, *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings*, volume 1008 of *Lecture Notes in Computer Science*, pages 275–285. Springer, 1994.

[DH19] Orr Dunkelman and Senyang Huang. Reconstructing an S-box from its Difference Distribution Table. *IACR Trans. Symmetric Cryptol.*, 2019(2):193–217, 2019.

[DHS+21] Xiaoyang Dong, Jialiang Hua, Siwei Sun, Zheng Li, Xiaoyun Wang, and Lei Hu. Meet-in-the-Middle Attacks Revisited: Key-Recovery, Collision, and Preimage Attacks. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part III*, volume 12827 of *Lecture Notes in Computer Science*, pages 278–308. Springer, 2021.

[DL22] Patrick Derbez and Baptiste Lambin. Fast MILP Models for Division Property. *IACR Trans. Symmetric Cryptol.*, 2022(2):289–321, 2022.

[FJ] Jean-Pierre Flori and Jérémy Jean. Libapn. https://github.com/ANSSI-FR/libapn.

[FWG+16] Kai Fu, Meiqin Wang, Yinghua Guo, Siwei Sun, and Lei Hu. MILP-Based Automatic Search Algorithms for Differential and Linear Trails for Speck. In Thomas Peyrin, editor, *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, volume 9783 of *Lecture Notes in Computer Science*, pages 268–288. Springer, 2016.

[GFG+16] Gerald Gamrath, Tobias Fischer, Tristan Gally, Ambros Gleixner, Gregor Hendel, Thorsten Koch, Stephen J Maher, Matthias Miltenberger, Benjamin Müller, Marc Pfetsch, et al. The SCIP Optimization Suite 3.2. 2016.

[gur] GUROBI Optimizer. https://www.gurobi.com/.

[HDS+22] Jialiang Hua, Xiaoyang Dong, Siwei Sun, Zhiyu Zhang, Lei Hu, and Xiaoyun Wang. Improved MITM Cryptanalysis on Streebog. *IACR Trans. Symmetric Cryptol.*, 2022(2):63–91, 2022.

[HNE22] Hosein Hadipour, Marcel Nageler, and Maria Eichlseder. Throwing Boomerangs into Feistel Structures Application to CLEFIA, WARP, LBlock, LBlock-s and TWINE. *IACR Trans. Symmetric Cryptol.*, 2022(3):271–302, 2022.

[HST+21] Kai Hu, Siwei Sun, Yosuke Todo, Meiqin Wang, and Qingju Wang. Massive Superpoly Recovery with Nested Monomial Predictions. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part I*, volume 13090 of *Lecture Notes in Computer Science*, pages 392–421. Springer, 2021.

[IBM] IBM. ILOG CPLEX Optimization Studio. https://www.ibm.com/products/ilog-cplex-optimization-studio/cplex-optimizer.

[LDB+19] Zheng Li, Xiaoyang Dong, Wenquan Bi, Keting Jia, Xiaoyun Wang, and Willi Meier. New Conditional Cube Attack on Keccak Keyed Modes. *IACR Trans. Symmetric Cryptol.*, 2019(2):94–124, 2019.

[Leo] Leo Perrin. SboxU v1.1. https://github.com/lpp-crypto/sboxU.

[LH94] Susan K. Langford and Martin E. Hellman. Differential-Linear Cryptanalysis. In Yvo Desmedt, editor, *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, volume 839 of *Lecture Notes in Computer Science*, pages 17–25. Springer, 1994.

[LJC23] Guangqiu Lv, Chenhui Jin, and Ting Cui. A MIQCP-Based Automatic Search Algorithm for Differential-Linear Trails of ARX Ciphers(Long Paper). *IACR Cryptol. ePrint Arch.*, page 259, 2023.

[LMC+22] Zhenyu Lu, Sihem Mesnager, Tingting Cui, Yanhong Fan, and Meiqin Wang. An stp-based model toward designing s-boxes with good cryptographic properties. *Des. Codes Cryptogr.*, 90(5):1179–1202, 2022.

[LMR22] Virginie Lallemand, Marine Minier, and Loïc Rouquette. Automatic Search of Rectangle Attacks on Feistel Ciphers: Application to WARP. *IACR Trans. Symmetric Cryptol.*, 2022(2):113–140, 2022.

[mag] Magma Computational Algebra System. http://magma.maths.usyd.edu.au/magma/.

[Mat94] Mitsuru Matsui. Linear Cryptanalysis Method for DES Cipher. In *Workshop on the Theory and Application of Cryptographic Techniques on Advances in Cryptology*, EUROCRYPT '93, pages 386–397, Secaucus, NJ, USA, 1994. Springer-Verlag New York, Inc.

[MR22] Rusydi H. Makarim and Raghvendra Rohit. Towards Tight Differential Bounds of Ascon A Hybrid Usage of SMT and MILP. *IACR Trans. Symmetric Cryptol.*, 2022(3):303–340, 2022.

[MWGP11] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming. In Chuankun Wu, Moti Yung, and Dongdai Lin, editors, *Information Security and Cryptology - 7th International Conference, Inscrypt 2011, Beijing, China, November 30 - December 3, 2011. Revised Selected Papers*, volume 7537 of *Lecture Notes in Computer Science*, pages 57–76. Springer, 2011.

[NPE23] Marcel Nageler, Felix Pallua, and Maria Eichlseder. Finding Collisions for Round-Reduced Romulus-H. *IACR Trans. Symmetric Cryptol.*, 2023(1):67–88, 2023.

[ort] OR-Tools. https://developers.google.com/optimization.

[Ras22] Shahram Rasoolzadeh. Low-latency boolean functions and bijective s-boxes. *IACR Trans. Symmetric Cryptol.*, 2022(3):403–447, 2022.

[RHSS21] Raghvendra Rohit, Kai Hu, Sumanta Sarkar, and Siwei Sun. Misuse-Free Key-Recovery and Distinguishing Attacks on 7-Round Ascon. *IACR Trans. Symmetric Cryptol.*, 2021(1):130–155, 2021.

[Rus] Rusydi H. Makarim, Yann Laigle-Chapuy, and Martin R. Albrecht. SAGE: S-Boxes and Their Algebraic Representations. https://doc.sagemath.org/html/en/reference/cryptography/sage/crypto/sbox.html.

[SG18] Ling Song and Jian Guo. Cube-Attack-Like Cryptanalysis of Round-Reduced Keccak Using MILP. *IACR Trans. Symmetric Cryptol.*, 2018(3):182–214, 2018.

[SHW+14] Siwei Sun, Lei Hu, Peng Wang, Kexin Qiao, Xiaoshuang Ma, and Ling Song. Automatic Security Evaluation and (Related-key) Differential Characteristic Search: Application to SIMON, PRESENT, LBlock, DES(L) and Other Bit-Oriented Block Ciphers. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the*

Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I, volume 8873 of *Lecture Notes in Computer Science*, pages 158–178. Springer, 2014.

[SSS+20] Dhiman Saha, Yu Sasaki, Danping Shi, Ferdinand Sibleyras, Siwei Sun, and Yingjie Zhang. On the Security Margin of TinyJAMBU with Refined Differential and Linear Cryptanalysis. *IACR Trans. Symmetric Cryptol.*, 2020(3):152–174, 2020.

[SSS+23] Danping Shi, Siwei Sun, Ling Song, Lei Hu, and Qianqian Yang. Exploiting Non-full Key Additions: Full-Fledged Automatic Demirci-Selçuk Meet-in-the-Middle Cryptanalysis of SKINNY. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part IV*, volume 14007 of *Lecture Notes in Computer Science*, pages 67–97. Springer, 2023.

[ST17] Yu Sasaki and Yosuke Todo. New Impossible Differential Search Tool from Design and Cryptanalysis Aspects - Revealing Structural Properties of Several Ciphers. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part III*, volume 10212 of *Lecture Notes in Computer Science*, pages 185–215, 2017.

[SWLW16] Ling Sun, Wei Wang, Ru Liu, and Meiqin Wang. MILP-Aided Bit-Based Division Property for ARX-Based Block Cipher. *IACR Cryptol. ePrint Arch.*, page 1101, 2016.

[TBP20] Shizhu Tian, Christina Boura, and Léo Perrin. Boomerang uniformity of popular s-box constructions. *Des. Codes Cryptogr.*, 88(9):1959–1989, 2020.

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

[WW11] Shengbao Wu and Mingsheng Wang. Security Evaluation against Differential Cryptanalysis for Block Cipher Structures. *IACR Cryptol. ePrint Arch.*, page 551, 2011.

[XZBL16] Zejun Xiang, Wentao Zhang, Zhenzhen Bao, and Dongdai Lin. Applying MILP Method to Searching Integral Distinguishers Based on Division Property for 6 Lightweight Block Ciphers. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 648–678, 2016.