Journal of Computer Virology and Hacking Techniques (2024) 20:113–125
https://doi.org/10.1007/s11416-023-00502-0

ORIGINAL PAPER

Provably minimum data complexity integral distinguisher based on conventional division property

Akram Khalesi[1] · **Zahra Ahmadian**[1]

Received: 23 October 2022 / Accepted: 23 August 2023 / Published online: 28 September 2023

Abstract
Division property is an effective method for finding integral distinguishers for block ciphers, performing cube attacks on stream ciphers, and studying the algebraic degree of boolean functions. One of the main problems in this field is how to provably find the smallest input multiset leading to a balanced output. In this paper, we propose a new method, using the division property, to find integral distinguishers for permutation functions and block ciphers, with provably-minimum data complexity, in the conventional division property model. The new method is based on a precise and efficient analysis of the target output bit's algebraic normal form. We examine the proposed method on LBlock, TWINE, SIMON, Present, Gift, and Clyde-128 block ciphers. Although in most cases, the results are consistent with the distinguishers reported in previous work, their *optimality* is *proved*, in the conventional division property model. Moreover, the proposed method can find distinguishers for 8-round Clyde-128 with less data complexity than previously reported. Based on the proposed method, we also develop an algorithm capable of determining the maximum number of balanced output bits for integral distinguishers with a certain number of active bits. Accordingly, for the ciphers under study, we determine the maximum number of balanced bits for integral distinguishers with data complexities set to minimum and slightly higher, resulting in improved distinguishers for Gift-64, Present, and SIMON64, in the conventional model.

Keywords Division property · Integral distinguisher · MILP · Clyde-128 · LBlock · TWINE · SIMON · Gift · Present

1 Introduction

Higher-order differential and *integral* characteristics [1–3] are generalizations of the differential characteristic that can be used as distinguishers for block ciphers. In these types of characteristics, the XOR of outputs, corresponding to a set of chosen inputs, equals zero in specific output positions. The output bits with such property are called *balanced*.

Division property, proposed by Todo at Eurocrypt 2015 [4], is an efficient method for finding these types of characteristics. Todo used this property in a breadth-first search algorithm and proposed improved integral characteristics for SIMON, Keccak, and Serpent. At Asiacrypt 2016, Xiang et al. proposed an MILP model describing the division property propagation, based on which they developed an automated method for finding integral distinguisher [5].

The division property comes in two models: *conventional division property* [4] and *three-subset division property* [6]. The conventional model is easier to be automated though it is less accurate and may miss some potential integral characteristics. In contrary, the three-subset model is precise, not missing any characteristic, but its automation is a challenging process. Despite some research on MILP modeling of three-subset division property for block ciphers, the proposed methods are time-consuming, working either for the small block size target ciphers [7] or if one mitigates its accuracy to some extent [8]. It deserves to be noted that in a modified variant of the three-subset division property called *three-subset division property without unknown subset* some challenges in MILP modeling of three-subset division property can be handled. This method is suited for analyzing public functions such as update functions in stream ciphers in cube attacks [9].

Division property has been studied to a great extent, based on which the existence of integral characteristics for most of the block ciphers have been investigated, commonly using an MILP-or SAT-aided tool [5, 10–15]. One of the main effi-

✉ Zahra Ahmadian
z_ahmadian@sbu.ac.ir

Akram Khalesi
a_khalesi@sbu.ac.ir

[1] Department of Electrical Engineering, Shahid Beheshti University, Tehran, Iran

ciency criteria for the integral distinguisher of a block cipher is its data complexity, for a specific number of rounds. So, a systematic and possibly provable method for finding the integral distinguisher with minimal data (and hence time) complexity would be of great importance. However, to the best of our knowledge, none of the papers published so far, except [11], has considered this problem. Hence, it seems that most of the reported division-based integral distinguishers are achieved by an extensive search.

The proposed method in [11] for finding integral distinguishers with reduced-data complexity works as follows. First, they search for balanced output bits corresponding to the input sets with one plaintext constant bit. Then, among the results found in the previous step, they try out all the possible combinations of the plaintext constant bits that lead to the same balanced bit in the output. This process is continued until none of the combinations of the input constants leads to a balanced output bit. The output of this algorithm is an integral distinguisher with minimum data complexity if the algorithm can be completely executed to the end. However, the authors stated that *"This approach improves the complexity of finding distinguishers with lower data complexity significantly, but often it is still computationally infeasible to find an optimal distinguisher."* One can refer to [11] for more details on this method.

1.1 Our contribution

In this paper, we propose a structured method for analyzing the algebraic normal form (ANF) of the Boolean function corresponding to the target output bit of the block cipher. As a result, the proposed algorithm returns the provably minimum data complexity for distinguishing the target output bit in the conventional division property model.

An integral characteristic needs a set of plaintexts active in some bits that are not included in any single monomial of the ANF of the target output bit. In other words, a set of plaintexts active in some bits does not lead to a balanced output bit if there is a monomial in the target ANF containing the active bits in the given input set of plaintexts. For example, suppose a target ANF of the form $f(x_0, x_1, x_2, x_3) = x_1x_2x_3 \oplus x_0x_3 \oplus x_1x_3 \oplus x_1 \oplus x_2$. Then, $f(\cdot)$ evaluated over a set of plaintexts with active bits $\{x_0, x_1, x_2\}$ sums to zero, which is an integral distinguisher, but the set of active bits $\{x_0, x_3\}$ or $\{x_1x_2\}$ is not so.

The division property enables us to check if a given monomial is present in the ANF of the target output bit or not. This assessment is accurate in the three-subset model, though a bit less accurate in the conventional model. In more detail, the absent monomials based on the conventional model are realized accurately, but some monomials which are discovered as existing ones in the ANF may not be really present. However, in this paper, we use the conventional division property as a tool for finding the minimum data integral distinguisher.

Suppose that using the division-based method, it has been determined that a specific monomial is present in the ANF of the cipher. As a result, all its submonomials (i.e. the monomials whose variables are a subset of the reference monomial) would be discarded from the candidate input active bits to be checked. For example, if we find $x_1x_2x_3$ as an existing term in the target ANF, none of its submonomials $x_1, x_2, x_3, x_1x_2, x_2x_3, x_1x_3$ can direct us to an integral distinguisher. As a result, we just need to study the monomials from the highest degree to the lower ones, while the submonomials of the discovered monomials will be excluded from this search. This process continues until all the monomials are checked, and the minimum size monomial not existing in the ANF of the target bit is identified. This monomial directly leads us to the minimum-sized data complexity integral distinguisher.

Using the new method, we studied the minimum data complexity integral distinguishers for the whole output bits of TWINE, SIMON, Gift-64, Gift-128, Present, LBlock, and partial output bits of 16-round LBlock and 8-round Clyde-128. Although, in most cases, the results are compliant with the reported distinguishers in the previous papers, the proposed method proves that these distinguishers are optimal in data complexity, in the conventional model. Furthermore, for LBlock, we found a 16-round integral distinguisher with data complexity 2^{62} which was not found in the conventional model, till now. We also found 8-round integral distinguishers for Clyde-128 with data complexity 2^{126} which requires one less active plaintext bit in comparison with the previously reported distinguishers. The results are summarized in Table 1.

Moreover, the proposed method for analyzing the ANF of the target output bit provides some information that can be exploited for another purpose. We represent how to utilize this information to find the maximum possible number of balanced bits in an integral distinguisher with a specified number of active bits in the conventional model.

Outline. The paper is organized as follows: we first introduce the notations used in the paper as well as the division property definition, its propagation rules, and MILP modeling of the propagation rules in Sect. 2. The new method and its application on different ciphers are presented in Sects. 3 and 4, respectively. Finally, we conclude the paper in Sect. 5.

2 Preliminaries

In this section, the notations used in the paper are introduced. Then a brief review of the division property and its MILP modeling is brought.

Table 1 Summary of the results

Cipher	Round	Active bits	Type	Balanced bits	References
LBlock	16	63	C	32	[5]
	16	62	T	18	[7]
	16	**62**	C	10*	This paper
	17	63	C, T	4	This paper, [7, 11]
Clyde-128	8	127	C	2	[16]
	8	**126**	C	1*	This paper
TWINE	16	63	C	32	This paper, [5]
SIMON-32	14	31	C	16	This paper, [5]
SIMON-48	16	47	C	24	This paper, [5]
SIMON-64	18	63	C	22	[5, 11]
	18	63	T	23	[7]
	18	63	C	**23**	This paper
SIMON-96	22	95	C	5	This paper, [5]
SIMON-128	26	127	C	3	This paper, [5]
Present	9	60	T	4	[7]
	9	60	C	1	This paper, [5]
	9	61	C	**4**	This paper
	9	62	C	**4**	This paper
	9	63	T	28	[7]
	9	63	C	**28**	This paper
Gift-64	9	61	C	5	This paper, [11]
	9	62	C	11	[11]
	9	62	C	**12**	This paper
	9	63	C	30	[11]
	9	63	C	**32**	This paper
Gift-128	11	127	C	32	This paper, [11]

The bolded entries refers to the improved distinguishers obtained in this paper compared to previous findings
C and T denote Conventional and Three subset division property, respectively
The *-marked cases are suboptimal since we have not studied all output bits

2.1 Notations

Throughout this paper, binary values are denoted by lowercase letters like x, vectors by boldface like $\boldsymbol{x}$, and (multi) sets by blackboard bold uppercase letters like $\mathbb{X}$. Let $\mathbb{F}_2$ denote the binary finite field and $\boldsymbol{a} = (a_0, a_1, \ldots, a_{n-1}) \in \mathrm{F}_2^n$ be an n-bit vector, where a_i denotes the i-th bit of $\boldsymbol{a}$. The vector $\boldsymbol{e}_i$ is the unit vector whose i-th element is 1. By $Hw(\boldsymbol{a})$, we mean the Hamming weight of $\boldsymbol{a}$, i.e. $\sum_{i=0}^{n} a_i$. For any $\boldsymbol{k} \in \mathrm{F}_2^n$ and $\boldsymbol{k}' \in \mathrm{F}_2^n$, we define $\boldsymbol{k} \succeq \boldsymbol{k}'$ if $k_i \geq k'_i$ for all $i = 0, 1, \ldots, n-1$. For n-bit vectors $\boldsymbol{x}$ and $\boldsymbol{u}$, a monomial is defined as $\boldsymbol{x}^{\boldsymbol{u}} = \prod_{i=0}^{n-1} {x_i}^{u_i}$. For a given monomial $\boldsymbol{x}^{\boldsymbol{u}}$, its submonomials are defined as $\boldsymbol{x}^{\boldsymbol{v}}$, where $\boldsymbol{v} \preceq \boldsymbol{u}$. The ANF of Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is $f(\boldsymbol{x}) = \bigoplus_{\boldsymbol{u} \in \mathbb{F}_2^n} a_{\boldsymbol{u}} \boldsymbol{x}^{\boldsymbol{u}}$, where $a_{\boldsymbol{u}} \in \mathbb{F}_2$. The boolean function f *contains* a monomial $\boldsymbol{x}^{\boldsymbol{v}}$ if there exists at least one $\boldsymbol{u}$ in the ANF of f where $a_{\boldsymbol{u}} = 1$ for $\boldsymbol{v} \preceq \boldsymbol{u}$. By the symbol $\leftarrow$, we mean adding a member to a (multi) set.

2.2 Division property

The division property can be modeled in two ways, conventional division property [4] and division property with three subsets [6]. The latter is more precise, based on which some innovative techniques for MILP-aided cube attack are proposed [9]. But, the MILP modeling of this method for block ciphers has some serious challenges and difficulties. In this paper, we use the conventional model and propose the new search algorithm, based on that.

Definition 1 Let $\mathbb{X}$ be a multiset whose elements take values from F_2^n. The multiset $\mathbb{X}$ has the conventional bit-based division property $\mathrm{D}_{\mathbb{K}}^{1^n}$ if it fulfills the following condition:

$$\bigoplus_{x \in \mathbb{X}} \boldsymbol{x}^{\boldsymbol{u}} = \begin{cases} unknown, & \text{if there exists } \boldsymbol{k} \in \mathbb{K} \text{ s.t. } \boldsymbol{u} \succeq \boldsymbol{k}, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

where $\mathbb{K}$ denotes a set of n-dimensional binary vectors.

2.2.1 Propagation rules for conventional bit-based division property

The rules for the conventional bit-based division property propagation through basic operations, proven in [4], are summarized as follows.

Rule 1 (Copy). *Let $\boldsymbol{y} = f(\boldsymbol{x})$ be the Copy function with $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1}) \in \mathrm{F}_2^n$ as the input, and $\boldsymbol{y} = (x_0, x_0, x_1, \ldots, x_{n-1}) \in \mathrm{F}_2^{n+1}$ as the output. If the input multiset $\mathbb{X}$ has $\mathrm{D}_{\mathbb{K}}^{1^n}$, then the output multiset $\mathbb{Y}$ has $\mathrm{D}_{\mathbb{K}'}^{1^{n+1}}$, where $\mathbb{K}'$ is computed from all $\boldsymbol{k} \in \mathbb{K}$ as*

$$\mathbb{K}' \leftarrow \begin{cases} (0, 0, k_1, \ldots, k_{n-1}), & \text{if } k_0 = 0, \\ (1, 0, k_1, \ldots, k_{n-1}), (0, 1, k_1, \ldots, k_{n-1}), & \text{if } k_0 = 1. \end{cases} \tag{2}$$

Rule 2 (And). *Let $\boldsymbol{y} = f(\boldsymbol{x})$ be the And function with $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1}) \in \mathrm{F}_2^n$ as the input, and $\boldsymbol{y} = (x_0 \wedge x_1, \ldots, x_{n-1}) \in \mathrm{F}_2^{n-1}$ as the output. If the input multiset $\mathbb{X}$ has $\mathrm{D}_{\mathbb{K}}^{1^n}$, then the output multiset $\mathbb{Y}$ has $\mathrm{D}_{\mathbb{K}'}^{1^{n-1}}$, where $\mathbb{K}'$ is computed from all $\boldsymbol{k} \in \mathbb{K}$ as*

$$\mathbb{K}' \leftarrow \left(\lceil \frac{k_0 + k_1}{2} \rceil, k_2, \ldots, k_{n-1} \right). \tag{3}$$

Rule 3 (Xor). *Let $\boldsymbol{y} = f(\boldsymbol{x})$ be the Xor function with $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1}) \in \mathrm{F}_2^n$ as the input, and $\boldsymbol{y} = (x_0 \oplus x_1, \ldots, x_{n-1}) \in \mathrm{F}_2^{n-1}$ as the output. If the input multiset $\mathbb{X}$ has $\mathrm{D}_{\mathbb{K}}^{1^n}$, then the output multiset $\mathbb{Y}$ has $\mathrm{D}_{\mathbb{K}'}^{1^{n-1}}$, where $\mathbb{K}'$ is computed from all $\boldsymbol{k} \in \mathbb{K}$ satisfying $(k_0, k_1) = (0, 0), (1, 0)$* or $(0, 1)$ as

$$\mathbb{K}' \leftarrow (k_0 + k_1, k_2, \ldots, k_{n-1}). \tag{4}$$

Rule 4 (S-Box). *Let $\boldsymbol{y} = f(\boldsymbol{x})$ be the S-Box function with $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1}) \in \mathrm{F}_2^n$ as the input, and $\boldsymbol{y} = (y_0, y_1, \ldots, y_{m-1}) \in \mathrm{F}_2^m$ as the output. Then, every $y_i, i \in 0, 1, \ldots, m-1$ can be expressed as a boolean function of $(x_0, \ldots, x_n - 1)$. For the input multiset $\mathbb{X}$ with $\mathrm{D}_{\mathbb{K}}^{1^n}$, then the output multiset $\mathbb{Y}$ has $\mathrm{D}_{\mathbb{K}'}^{1^m}$, calculated as follows:*

$$\mathbb{K}' = \{\boldsymbol{u}' \in \mathrm{F}_2^m \mid \text{for any } \boldsymbol{u} \in \mathbb{K}, \text{ if } \boldsymbol{y}^{\boldsymbol{u}'} \text{ contains any term } \boldsymbol{x}^{\boldsymbol{v}} \text{ satisfying } \boldsymbol{v} \succeq \boldsymbol{u}\}. \tag{5}$$

2.2.2 The MILP modeling of conventional bit-based division property

In [5], Xiang et al. defined the concept of *division trail* and used linear inequalities to model propagation rules of division property. As a result, they could develop an MILP model which can efficiently describe the propagation of conventional bit-based division property through a cipher. We summarize the concept of the division trail and the MILP models proposed by them. The new method in this paper is based on Xiang et al.'s MILP-aided division property propagation for finding an integral distinguisher.

Definition 2 (Division Trail). Let $D_{\mathbb{K}_i}$ be the division property of the input for the i-th round function. Consider the propagation of the conventional bit-based division property $\{\boldsymbol{k_I}\} \overset{def}{=} \mathbb{K}_0 \overset{f}{\rightarrow} \mathbb{K}_1 \overset{f}{\rightarrow} \cdots \overset{f}{\rightarrow} \mathbb{K}_r$, where $\mathbb{K}_i \overset{f}{\rightarrow} \mathbb{K}_{i+1}$ denotes the propagation of $\mathbb{K}_i$ to $\mathbb{K}_{i+1}$ through the round function f. For any vector $\boldsymbol{k_{i+1}} \in \mathbb{K}_{i+1}$, there must exist a vector $\boldsymbol{k_i} \in \mathbb{K}_i$ such that $\boldsymbol{k_i}$ can propagate to $\boldsymbol{k_{i+1}}$ according to the propagation rules. Beside, for $(\boldsymbol{k_0}, \boldsymbol{k_1}, \ldots, \boldsymbol{k_r}) \in \mathbb{K}_0 \times \mathbb{K}_1 \times \ldots \times \mathbb{K}_r$, we call $\boldsymbol{k_0} \overset{f}{\rightarrow} \boldsymbol{k_1} \overset{f}{\rightarrow} \cdots \overset{f}{\rightarrow} \boldsymbol{k_r}$ an r-round conventional bit-based division property trail if $\boldsymbol{k_i}$ can propagate to $\boldsymbol{k_{i+1}}$ for all $i \in \{0, 1, \ldots, r-1\}$.

They also modeled the conventional bit-based division property propagation of three basic components (Copy, Xor, And) as well as S-Box by linear inequalities. Therefore, they could generate a MILP model that covers all division trails. Finally, they propose the idea that if it can be shown that there is no division trail $\boldsymbol{k_0} \overset{E_k}{\rightarrow} \boldsymbol{e}_i$, then the i-th bit of the r-round output is always balanced.

The MILP models of the basic operations are summarized as follows, according to [5].

MILP Model 1 (Copy). The propagation of the division property for Copy operation $a \rightarrow (b_1, b_0)$ can be adequately described by the following linear constraint:

$$\begin{cases} a - b_1 - b_0 = 0, \\ a, b_0, b_1 \text{ are binaries}. \end{cases} \tag{6}$$

MILP Model 2 (Xor). The propagation of the division property for Xor operation $(a_1, a_0) \rightarrow b$ can be adequately described by the following linear constraint:

$$\begin{cases} a_1 + a_0 - b = 0, \\ a_0, a_1, b \text{ are binaries}. \end{cases} \tag{7}$$

MILP Model 3 (And). The propagation of the division property for And operation $(a_1, a_0) \rightarrow b$, can adequately be described by the following linear constraints:

$$\begin{cases} b - a_0 \geq 0, \\ b - a_1 \geq 0, \\ b - a_0 - a_1 \leq 0, \\ a_0, a_1, b \text{ are binaries}. \end{cases} \tag{8}$$

We use Xiang et al.'s proposed method to get a set of linear inequalities for modeling the conventional bit-based division

property propagation through an S-Box. Their method proposes to use the $inequality\_generator()$ function in Sage to obtain the H-representation of the convex hull of division trails through S-Box using Rule 4. As a result, one can achieve a set of linear inequalities for modeling the conventional bit-based division property propagation through an S-Box. Applying the greedy algorithm proposed by Sun et al. [17], the large set of linear inequalities reduces into a smaller one and the MILP model becomes computationally feasible.

2.3 Gurobi

Gurobi [18] is an optimization solver for different types of problems ranging from Linear Programming (LP) and Mixed-Integer Linear Programming (MILP) to different types of Quadratic Programming (QP).

Applying Gurobi as an MILP solver for cryptanalysis purposes has become prevalent during the last decade and lots of new techniques for different cryptanalysis methods have been proposed based on its capabilities. We also utilized Gurobi optimizer and its Python interface for our proposed method. It should be noted that its interfaces for other programming languages such as C or C++ can be utilized as well.

3 Minimum-data integral distinguisher

In this section, we describe the new method for finding integral distinguishers with a provably minimum data complexity in the conventional division property model. Suppose that we aim to find the integral distinguisher with the minimum data complexity for the j^{th} output bit of the cipher, called the target bit. Let $f(\boldsymbol{x}) = \bigoplus_{\boldsymbol{u}\in\mathbb{F}_2^n} a_{\boldsymbol{u}}\boldsymbol{x}^{\boldsymbol{u}}$ be the ANF of the target bit, where $a_{\boldsymbol{u}} \in \mathbb{F}_2$. We are looking for the lowest degree monomial which is not itself a monomial of f, and is not a submonomial of any monomials existing in f, as well. In other words, we should find $\alpha^* = \arg\min Hw(\boldsymbol{\alpha})$ where $\boldsymbol{\alpha} \in \{\boldsymbol{v} \in \mathbb{F}_2^n | \boldsymbol{v} \not\preceq \boldsymbol{u}, a_{\boldsymbol{u}} = 1\}$. Let $J = \{i | \alpha_i^* = 1\}$, then any set of plaintexts with active bits in positions J results in a balanced output.

Moreover, the proposed algorithm is capable of returning suboptimal distinguishers in case of time constraints. We use Gurobi as a solver for MILP problems and describe the new method based on its features.

3.1 The proposed search algorithm

The proposed method is described in Algorithm 2, based on the Python interface for Gurobi which is summarized in Table 2, and is detailed in this section. The flowchart of the proposed method is given in Fig. 1.

First of all, we make an MILP model in which the constraints describe the propagation of division property (these constraints are listed in $\mathcal{L}_1$), the division property of the output equals to the unit vector $\boldsymbol{e}_j$ and the objective is maximization of the sum of plaintext bits (Obj_1). Such a model finds the monomial with the highest algebraic degree existing in the boolean function of the target bit (line 6 of Algorithm 1). Using an optimization solver, Gurobi in our case, we solve the model. If the model is feasible, from the optimization result we extract the highest-degree monomial, as well as the plaintext bits not existing in this monomial (lines 11–17 of Algorithm 1).

As an example, suppose that the unknown boolean function of the target output bit is $f_1(x_0, x_1, x_2, x_3) = x_1x_2x_3 \oplus x_0x_3 \oplus x_1x_3 \oplus x_1 \oplus x_2$, for which we aim to find an integral distinguisher with the minimum data complexity. By solving the above model for this function, the discovered monomial at this step would be $x_1x_2x_3$.

Notice that in order to distinguish f from a random function, one should choose a set of plaintexts whose active bits

Algorithm 1: Finding minimum data complexity integral distinguisher

```
1  Input: L_1 = Division property propagation constraints
2  Output: Active plaintext bits
3  begin
4      Obj_1 ← max(Σ_{i=0}^{n-1} x_i)// x_i, i = 0, ..., n − 1 are plaintext
       bits
5      Obj_2 ← min(Σ_{i=0}^{n-1} x_i)
6      L_2 ← null
7      while (1) do
8          M ← M(Obj_1, L_1, L_2)
9          M.Optimize()
10         if (M.Status = 2) then // model is feasible
11             obj ← M.GetObjective()
12             NewConstr ← null
13             for i in range (0, blockSize) do
14                 var ← obj.GetVar(i)
15                 val ← var.GetAttr('x')
16                 if val = 0 then
17                     NewConstr.Append(var.GetAttr('VarName'))
18                 end
19             end
20             L_2.Append(NewConstr ≥ 1)
21         else if M.Status = 3 then // model is infeasible
22             M ← M(Obj_2, L_2)
23             M.Optimize()
24             ActiveBits ← null
25             for i in range range(0, blockSize) do
26                 var ← obj.GetVar(i)
27                 val ← var.GetAttr('x')
28                 if val = 1 then
29                     ActiveBits.Append(var.GetAttr('VarName'))
30                 end
31             end
32             return ActiveBits
33         end
34     end
35 end
```

Table 2 Interface for utilizing Gurobi

Notation	Explanation
$M(Obj, \mathcal{L})$	Model M with objective Obj and constraints $\mathcal{L}$
M.OPTIMIZE()	Optimizes model M
M.STATUS	Indicates the current optimization status of model M
M.GETOBJECTIVE()	Returns a linear expression equal to the linear objective, in our case
obj.GETVAR(i)	Returns variable for the term at index i in the linear expression of obj
var.GETATTR($'x'$)	Returns the value of the variable var in the current solution
var.GETATTR($'VarName'$)	Returns variable name of var

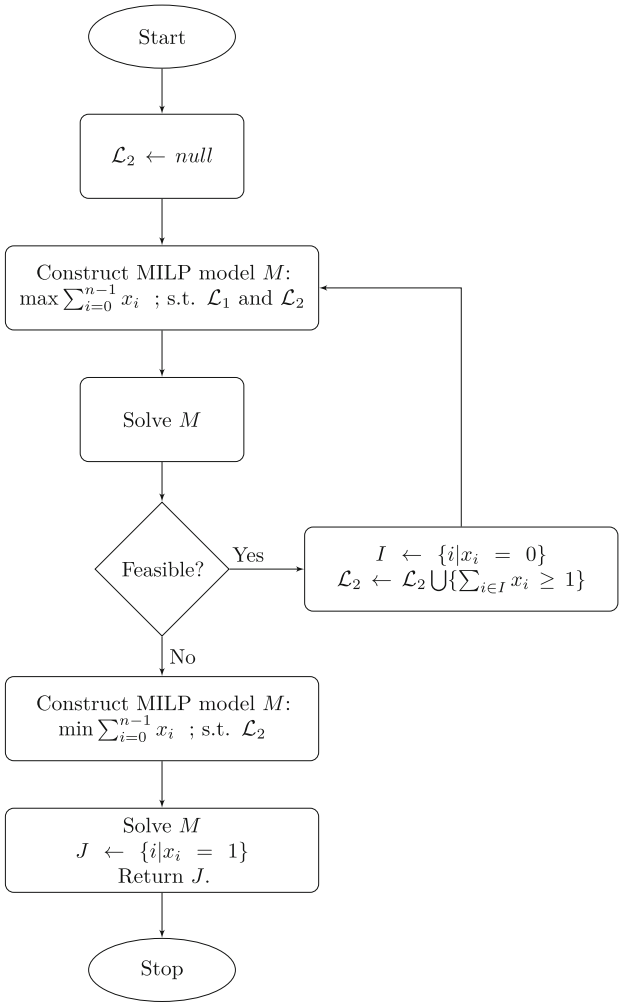

Fig. 1 Flowchart of the proposed method

are not a submonomial of any monomial of f. For example, having obtained the monomial $x_1x_2x_3$ in f, no set of plaintexts, active only in any submonomial of $x_1x_2x_3$, i.e. any combination of x_1, x_2 and x_3, can result in balanced output, according to the conventional bit-based division property.

Based on this fact, we then add a new constraint to the model to filter out all the submonomials of the obtained monomial from the feasible set of the model, in search of the subsequent monomial of f. This constraint should imply that at least one of the plaintext bits non-existing in the obtained monomial must be present in the next monomial. So, sum of the plaintext bits not included in the obtained monomial should be equal to or greater than one (line 18 of Algorithm 1). Therefore, in the subsequent search there is no need to study all the monomials in the ANF of f. In the above example, this constraint is $x_0 \geq 1$, which removes monomials x_1x_3, x_1, and x_2 of f from the feasible set.

The newly generated constraint corresponding to the obtained monomial is now included in the main model, and is saved in a separate list ($\mathcal{L}_2$, line 18 in Algorithm 1), as well. Having added the new constraint to the main model, it is updated and solved to find the next highest degree monomial of f, satisfying the new constraint. In our example, we expect to get the monomial x_0x_3 after solving the updated model.

Once the new monomial is realized, we add another constraint to the model (as well as list $\mathcal{L}_2$) to remove all of its submonomials from the feasible set. Then, we solve it to find the next highest degree monomial, not being a submonomial of the previously obtained monomials. In our example, this new constraint is $x_1 + x_2 \geq 1$. These steps are repeated until the model becomes infeasible (line 19 in Algorithm 1). This means that f does not contain a monomial satisfying the constraints in list $\mathcal{L}_2$, anymore. At this step, all the monomials in the ANF of the target function have been studied and we have come up with a set of constraints by which the minimum set of plaintexts which make the j^{th} output bit balanced, can be obtained.

To find this minimum set, we make a new model in which the constraints are as list $\mathcal{L}_2$, and the objective is to minimize the sum of plaintext bits (line 20 in Algorithm 1). By solving this model, we would be directed to the integral distinguisher for the target output with the minimum data complexity, since it returns a monomial with the lowest degree which is not a submonomial of any monomials of f.

In our example, the objective function is to minimize $x_0 + x_1 + x_2 + x_3$, subject to the constraints $x_0 \geq 1$ and $x_1 + x_2 \geq 1$. This model returns the active plaintext bits $\{x_0, x_1\}$ or $\{x_0, x_2\}$ as the active bit sets with the smallest possible size,

which result in a balanced property in the target bit, which is the minimal data complexity integral distinguisher in the conventional division model.

Finally, we establish the subsequent theorem as evidence supporting our assertion that Algorithm 1 is capable of identifying the integral distinguisher with the minimum data complexity, based on conventional division property.

Theorem 1 *The output of Algorithm 1 is the set of active plaintext bits of minimum size that results in a balanced property in the target output bit.*

Proof We prove this theorem by contradiction. Suppose the output of Algorithm 1, denoted as $\tilde{J}$, is not the set of active plaintext bits of the minimum size that results in a balanced property in the target output bit. Hence, there must exist another set, namely J^*, such that $|J^*| < |\tilde{J}|$ while for any $\boldsymbol{u} \succeq \boldsymbol{v}$, where $v_i = 1, i \in J^*$, it holds that $a_{\boldsymbol{u}} = 0$. In other words, the monomial corresponding to J^* is not a (sub)monomial of the ANF of the target output bit.

J^* either satisfies the constraints of $\mathcal{L}_2$ or it does not. If J^* satisfies $\mathcal{L}_2$, it directly contradicts the fact that $\tilde{J}$ is the minimum-sized set satisfying $\mathcal{L}_2$ (Line 22 of Algorithm 1). Alternatively, if J^* does not satisfy $\mathcal{L}_2$, it means that there exists at least one constraint in $\mathcal{L}_2$ that is not satisfied by J^*. Let's denote the first unsatisfied constraint as $const^*$, which is generated at the r^*-th iteration of Line 7 of Algorithm 1. This implies the existence of a feasible model $M(Obj_1, \mathcal{L}_1, \mathcal{L}_2^{(r^*-1)})$ with a solution $\boldsymbol{u}^*$. It means that $\boldsymbol{x}^{\boldsymbol{u}^*}$ exists in the ANF of the target output bit. Hence, the constraint $const^*$, generated in Line 20 of this iteration, would be $\sum_{i,\boldsymbol{u}_i^*=0} x_i \geq 1$.

Since we assumed that J^* does not satisfy $const^*$, it must hold that $\sum_{i\in J^*,\boldsymbol{u}_i^*=0} x_i = 0$. As a result, the monomial associated with J^* becomes a submonomial of the target ANF, which leads to a contradiction. □

3.2 Advantages and limitations

In this section, we bring some discussion about the capabilities of the proposed algorithm, as well as its limitations.

Provably minimum data in the conventional model. The most important advantage of the proposed method is that it finds the integral distinguisher with a provably minimum data complexity for a target output bit, in the conventional model for division property. To be more specific, the proposed algorithm returns the smallest monomial not being a submonomial of any monomials of the ANF of the target bit. By choosing a set of plaintexts, which is active in the same bits of this monomial, we have an integral distinguisher with balanced property in the target bit.

Suboptimal integral distinguishers. The new method is based on analyzing the structure of the ANF of the target output bit that gives us valuable information from the ANF. This information can be utilized for different purposes such as finding sub-optimal distinguishers in time-consuming cases. Note that any d-degree monomial of f, discovered in an iteration of Line 7 of Algorithm 1, directs us to an integral distinguisher with data complexity 2^{d+1}, conditioned that $d < n-1$. In this distinguisher, the set of plaintext bits is active in all the bits present in the monomial plus an arbitrary absent bit. Since it is guaranteed that the discovered monomials during running the algorithm are not a submonomial of any monomial in the target ANF, such a set of plaintexts results in balanced property in the target output bit. In this way, for each d-degree monomial discovered in Line 7 of Algorithm 1, one can make $(n-d)$ distinct integral distinguishers with data complexity 2^{d+1}, conditioned that $d < n-1$.
As an example, consider the following Boolean function as the target output bit in the cipher:

$$\begin{aligned} f_2(x_0, x_1, x_2, x_3, x_4) = {} & x_0x_1x_2x_3 \oplus x_0x_2x_3x_4 \\ & \oplus x_0x_1x_4 \oplus x_0x_2x_3 \oplus x_1x_2x_4 \oplus \\ & x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_0x_1 \\ & \oplus x_0x_3 \oplus x_1x_3 \oplus x_1x_4 \oplus \\ & x_2x_3 \oplus x_1 \oplus x_2 \oplus x_4 \oplus 1 \end{aligned} \tag{9}$$

Table 3 is the trace table of Algorithm 1 applying to f_2, which shows that the algorithm finishes after 5 iterations. Now, assume that we terminate Algorithm 1 after the 3^{rd} iteration, where monomial $x_0x_1x_4$ is found. The algorithm ensures that there is no monomial of higher degrees containing $x_0x_1x_4$. Hence, any set of plaintexts active in $\{x_0, x_1, x_4\}$ plus one (or more) input bits leads to a balanced output. So, the sets of plaintexts with active bits $\{x_0, x_1, x_2, x_4\}, \{x_0, x_1, x_3, x_4\}$ lead to a balanced output. Clearly, there is no guarantee that such distinguishers are the minimum-data ones. So, they are the suboptimal integral distinguishers.

Multi-fold integral distinguishers. The other capability of the proposed method is that it can be used to provably determine the maximum number of balanced output bits, all with the same minimum-size set of plaintexts. Notice that for the i^{th} individual output bit, by executing Algorithm 1, we have extracted a set of constraints, listed in $\mathcal{L}_2^{(i)}$, guaranteeing that the i^{th} output bit is balanced with the minimum number of plaintext active bits, $m \leq n-1$. One can construct a new model by conditionally merging all the constraints of $\mathcal{L}_2^{(i)}$, for $i = 0, \ldots, n-1$. For this

Table 3 Trace table of Algorithm 1 applying to boolean function f_2

Iteration	Extracted monomial	Constraint	Removed monomials
1	$x_0x_1x_2x_3$	$x_4 \geq 1$	$x_0x_2x_3, x_0x_1, x_0x_3, x_1x_3, x_2x_3, x_1, x_2$
2	$x_0x_2x_3x_4$	$x_1 \geq 1$	$x_2x_3x_4, x_4$
3	$x_0x_1x_4$	$x_2 + x_3 \geq 1$	x_1x_4
4	$x_1x_2x_4$	$x_0 + x_3 \geq 1$	–
5	$x_1x_3x_4$	$x_0 + x_2 \geq 1$	–

Algorithm 2: Finding the maximum number of balanced output bits with the same minimal input set

Input: $\begin{cases} \mathcal{L}_2^{(0)} = \{A^{(0)} \cdot \boldsymbol{x} \geq \boldsymbol{b}^{(0)}\}, \\ \vdots \\ \mathcal{L}_2^{(n-1)} = \{A^{(n-1)} \cdot \boldsymbol{x} \geq \boldsymbol{b}^{(n-1)}\} \\ m = \textit{minimum data} \end{cases}$

Output: *maximum number of balanced output bits*

begin

  $Obj \leftarrow \max(B = \sum_{i=0}^{n-1} B_i)$

  $\mathcal{L}_3 \leftarrow null$

  for i **in range** $(0, n-1)$ **do**

    $\mathcal{L}_3$.APPEND$(\boldsymbol{A}^{(i)} \cdot \boldsymbol{x} + C(1 - B_i)\mathbf{1} \geq \boldsymbol{b}^{(i)})$

  end

  $\mathcal{L}_3$.APPEND$(\sum_{i=0}^{n-1} x_i = m)$

  $M \leftarrow M(Obj, \mathcal{L}_3)$

  M.OPTIMIZE

  return M.GETOBJECTIVE

end

paper, we use the conditional constraints method [19] as follows.

Suppose that the binary variable B_i implies that the i^{th} output bit is included in the set of candidate target bits or not. In other words, if $B_i = 1$, then the i^{th} output bit is taken into account, and hence the set of constraints associated with its balance ($\mathcal{L}_2^{(i)}$) should be included in the model. Otherwise, if $B_i = 0$, this output bit is not considered and the constraints in $\mathcal{L}_2^{(i)}$ should become ineffective. For this purpose, for all constraints in $\mathcal{L}_2^{(i)}$ of the form $\boldsymbol{A}^{(i)} \cdot \boldsymbol{x} \geq \boldsymbol{b}^{(i)}$ we add the following constraint to the model:

$$\boldsymbol{A}^{(i)} \cdot \boldsymbol{x} + C(1 - B_i)\mathbf{1} \geq \boldsymbol{b}^{(i)} \tag{10}$$

where C is a large enough constant (in our case, it suffices to set $C = \max_{i,j} b_j^{(i)}$ [19]) and $\mathbf{1}$ is the all-ones column vector. Moreover, another constraint, ensuring that the data complexity remains minimal, is added to the model as $\sum_{i=0}^{n-1} x_i = m$, and finally, the objective would be $\max B = \sum_{i=0}^{n-1} B_i$. It deserves to be noted that constraint $\sum_{i=0}^{n-1} x_i = m$, with m higher than the minimum active bits, determines the maximum number of balanced bits in distinguishers with higher data complexities.

Efficiency The new method is more time-efficient for the target number of rounds equal or close to the maximum distinguishable one. However, for cases with fewer number of rounds, the method becomes more time-consuming and in some cases, one may settle for the sub-optimal distinguisher, by early terminating the program. This is due to the fact that the ANF of the target output of the higher number of rounds contains (potentially many) monomials with maximum or near to maximum degrees. Since monomials with higher degrees remove more submonomials, the problem gets simpler and the final result will be achieved faster, as well. On the other hand, this method may become computationally impractical for fewer number of rounds. However, this drawback does not affect the use cases of the new method since we are usually interested in analyzing the minimum data complexity for the maximum distinguishable number of rounds.

4 Applications to LBlock, TWINE, SIMON, PRESENT, Gift, and Clyde-128

We applied Algorithm 1 to LBlock, TWINE, SIMON, Present, Gift, and Clyde-128 to find the minimum data complexity integral distinguishers for them based on conventional division property. As an application of the capability of the new method in provably determining the maximum number of balanced output bits, we applied Algorithm 2 on 17-round LBlock and determined the maximum number of balanced bits in any 17-round integral distinguisher on LBlock in the conventional model. The extracted distinguishers are reported in Appendix A.

4.1 Application to LBlock

LBlock [20] is a lightweight block cipher with 64-bit block size and 80-bit key size. It has a Feistel structure, consisting of 32 rounds. LBlock round function is depicted in Fig. 2.

Integral characteristic for LBlock has been studied based on the conventional [11] as well as three-subset division property [7]. Both result in a 17-round integral distinguisher with data complexity 2^{63}. The best previous 16-round integral distinguisher in the conventional model requires 2^{63} cho-

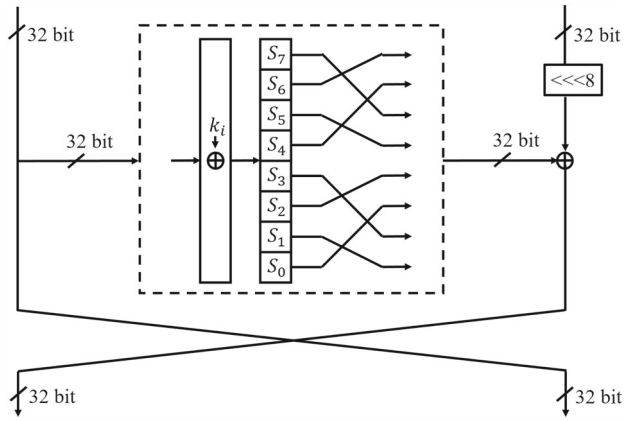

Fig. 2 Round function of LBlock

sen plaintexts [5], while Wang et al. reported an improved 16-round integral distinguisher with data complexity 2^{62} in three-subset model [7]. However, we found a 16-round integral distinguisher with data complexity 2^{62} in the conventional model by utilizing the new method.

In order to show the capability of the new method to find multi-fold integral distinguishers, presented in section 3.2, we applied Algorithm 2 on 17-round LBlock and found out the number of balanced bits in a 17-round integral distinguisher is at most 4, for a single set of plaintexts with 63 active bits. All these multi-fold distinguishers are listed in Appendix A.1. Hence, the reported distinguisher in [11] cannot be improved in the conventional model neither in data complexity nor in the number of balanced bits.

4.2 Application to TWINE

TWINE [21] is a lightweight block cipher with 64-bit block size and 80-bit key size. It is based on Feistel construction and its round function is depicted in Fig. 3. The best integral distinguisher based on conventional division property covers 16 rounds with data complexity 2^{63} [5, 11]. We analyzed all the 64 bits of the 16^{th} round output with the new method and the results are summarized in Table 4. As reported in [11], the new method also shows that there is no integral distinguisher for 17 or more rounds of TWINE using conventional division property.

Table 4 Summary of distinguishers for TWINE block cipher

Output bit no.	0	1	2	3	4	5	6	7
Min. data compl.	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	2^{64}	2^{64}	2^{64}	2^{64}
Output bit no.	8	9	10	11	12	13	14	15
Min. data compl.	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	2^{64}	2^{64}	2^{64}	2^{64}
Output bit no.	16	17	18	19	20	21	22	23
Min. data compl.	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	2^{64}	2^{64}	2^{64}	2^{64}
Output bit no.	24	25	26	27	28	29	30	31
Min. data compl.	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	2^{64}	2^{64}	2^{64}	2^{64}
Output bit no.	32	33	34	35	36	37	38	39
Min. data compl.	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	2^{64}	2^{64}	2^{64}	2^{64}
Output bit no.	40	41	42	43	44	45	46	47
Min. data compl.	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	2^{64}	2^{64}	2^{64}	2^{64}
Output bit no.	48	49	50	51	52	53	54	55
Min. data compl.	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	2^{64}	2^{64}	2^{64}	2^{64}
Output bit no.	56	57	58	59	60	61	62	63
Min. data compl.	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	2^{64}	2^{64}	2^{64}	2^{64}

Distinguishers highlighted in bold represent cases with valid (non-full codebook) data complexity

We applied Algorithm 2 on 16-round TWINE and found out the maximum number of balanced bits in a 16-round integral distinguisher is at most 32. This multi-fold distinguisher is listed in Appendix A.3. Hence, the reported distinguisher in [5] can be improved in the conventional model neither in data complexity nor in the number of balanced bits.

4.3 Application to SIMON

SIMON [22] is a family of lightweight block ciphers of different block sizes (32, 48, 64, 96 or 128) and different key sizes (64, 72, 96, 128, 144, 192, or 256). It is based on Fiestel construction and consists of a variety of number of rounds (32, 36, 42, 44, 52, 54, 68, 69, or 72) depending on the block and key sizes. The round function is depicted in Fig. 4.

For different versions of SIMON, we searched for the minimum data complexity for the most number of distinguishable rounds, and the results are summarized in Table 5. Contrary to SPN block ciphers, SIMON structure has a rotational symmetry, which causes the minimum data complexity for the

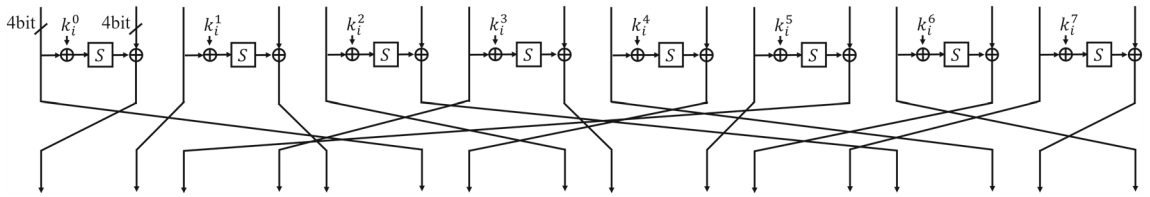

Fig. 3 Round function of TWINE

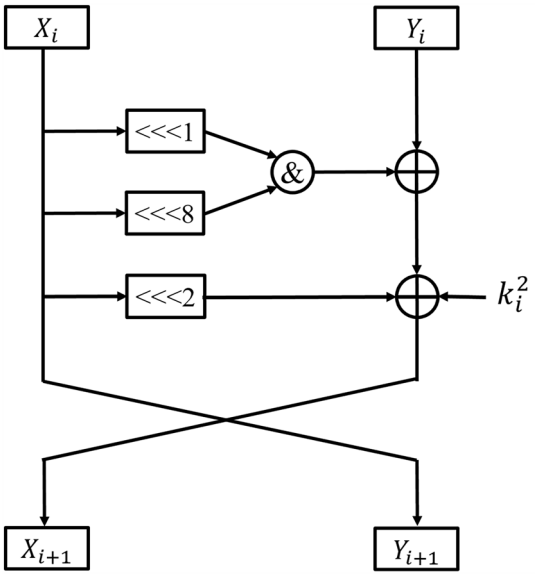

Fig. 4 Round function of SIMON

integral distinguisher of half of the bits of the output block (due to its Fiestel construction) to be the same.

We also applied Algorithm 2 to different versions of SIMON. The maximum number of balanced bits in versions with block lengths 32, 48, 96, and 128 are the same as the reported ones in [5, 11] in the conventional model. However, for block length 64, the maximum number of balanced bits is 23, which exceeds the previously reported distinguishers [5, 11] one bit in the conventional model.

Table 5 Summary of distinguishers for SIMON block cipher

Block length	32	48	64	96	128
No. of rounds	14	16	18	22	26
Min. data compl	$\mathbf{2^{31}}$	$\mathbf{2^{47}}$	$\mathbf{2^{63}}$	$\mathbf{2^{95}}$	$\mathbf{2^{127}}$

Distinguishers highlighted in bold represent cases with valid (non-full codebook) data complexity

4.4 Application to PRESENT

PRESENT [23] is an SPN block cipher with a bit permutation linear layer. It consists of 31 rounds and its round function is depicted in Fig. 5. It has two versions with 80- and 128-bit key size and 64-bit block size. The best integral distinguisher based on conventional division property covers 9 rounds of PRESENT with data complexity 2^{60} [5, 11]. We analyzed all the 64 bits of the 9^{th} round output with the proposed method. The results are summarized in Table 6.

Applying Algorithm 2, the maximum number of balanced bits are studied for 9-round integral distinguishers with different numbers of active bits. As shown in Table 1, the maximum number of balanced bits with 61 and 62 active bits is the same, i.e. increasing the number of active bits from 61 to 62 does not increase the maximum number of balanced bits. Moreover, we found that the maximum number of balanced bits with 63 active bits, equals 28 which was previously achieved just in the three subset model.

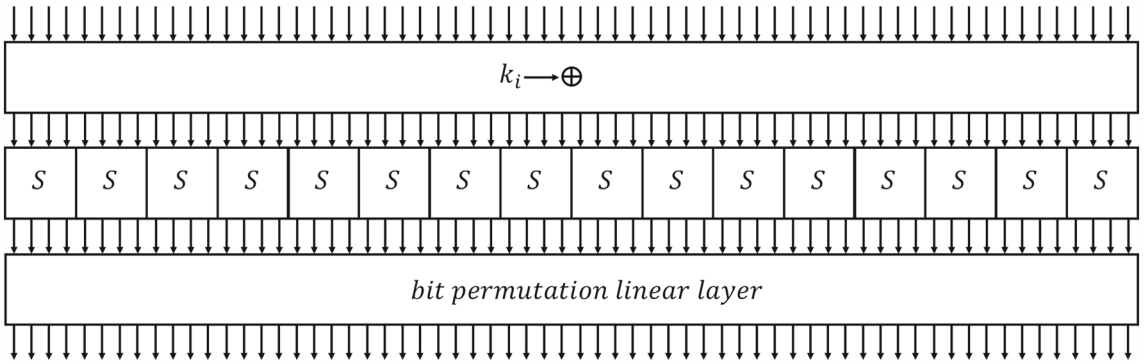

Fig. 5 Round function of PRESENT

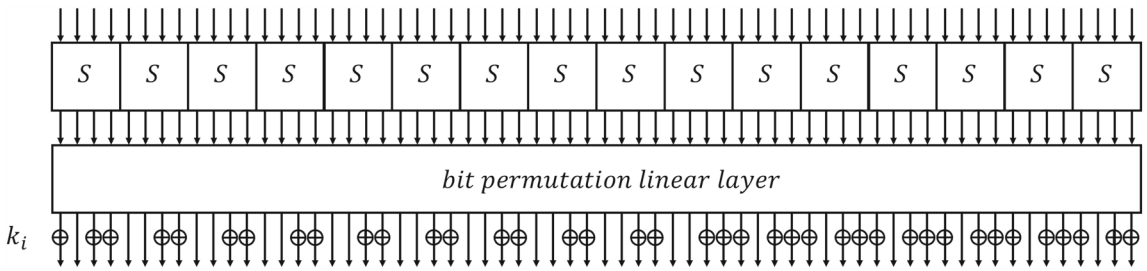

Fig. 6 Round function of Gift-64

Table 6 Summary of distinguishers for PRESENT block cipher

Output bit no.	0	1	2	3	4	5	6	7
Min. data compl.	2^{64}	2^{64}	2^{64}	2^{64}	2^{64}	2^{64}	2^{64}	2^{64}
Output bit no.	8	9	10	11	12	13	14	15
Min. data compl.	2^{64}	2^{64}	2^{64}	2^{64}	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{61}}$
Output bit no.	16	17	18	19	20	21	22	23
Min. data compl.	2^{64}	2^{64}	2^{64}	2^{64}	2^{64}	2^{64}	2^{64}	2^{64}
Output bit no.	24	25	26	27	28	29	30	31
Min. data compl.	2^{64}	2^{64}	2^{64}	2^{64}	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{61}}$
Output bit no.	32	33	34	35	36	37	38	39
Min. data compl.	2^{64}	2^{64}	2^{64}	2^{64}	2^{64}	2^{64}	2^{64}	2^{64}
Output bit no.	40	41	42	43	44	45	46	47
Min. data compl.	2^{64}	2^{64}	2^{64}	2^{64}	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{61}}$
Output bit no.	48	49	50	51	52	53	54	55
Min. data compl.	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$
Output bit no.	56	57	58	59	60	61	62	63
Min. data compl.	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{63}}$	$\mathbf{2^{60}}$

Distinguishers highlighted in bold represent cases with valid (non-full codebook) data complexity

Table 7 Summary of distinguishers for Gift-64 block cipher

Output bit no.	0	1	2	3	4	5	6	7
Min. data compl.	$\mathbf{2^{61}}$	$\mathbf{2^{63}}$	2^{64}	2^{64}	$\mathbf{2^{61}}$	$\mathbf{2^{63}}$	2^{64}	2^{64}
Output bit no.	8	9	10	11	12	13	14	15
Min. data compl.	$\mathbf{2^{61}}$	$\mathbf{2^{63}}$	2^{64}	2^{64}	$\mathbf{2^{61}}$	$\mathbf{2^{63}}$	2^{64}	2^{64}
Output bit no.	16	17	18	19	20	21	22	23
Min. data compl.	$\mathbf{2^{61}}$	$\mathbf{2^{63}}$	2^{64}	2^{64}	$\mathbf{2^{61}}$	$\mathbf{2^{63}}$	2^{64}	2^{64}
Output bit no.	24	25	26	27	28	29	30	31
Min. data compl.	$\mathbf{2^{61}}$	$\mathbf{2^{63}}$	2^{64}	2^{64}	$\mathbf{2^{61}}$	$\mathbf{2^{63}}$	2^{64}	2^{64}
Output bit no.	32	33	34	35	36	37	38	39
Min. data compl.	$\mathbf{2^{61}}$	$\mathbf{2^{63}}$	2^{64}	2^{64}	$\mathbf{2^{61}}$	$\mathbf{2^{63}}$	2^{64}	2^{64}
Output bit no.	40	41	42	43	44	45	46	47
Min. data compl.	$\mathbf{2^{61}}$	$\mathbf{2^{63}}$	2^{64}	2^{64}	$\mathbf{2^{61}}$	$\mathbf{2^{63}}$	2^{64}	2^{64}
Output bit no.	48	49	50	51	52	53	54	55
Min. data compl.	$\mathbf{2^{61}}$	$\mathbf{2^{63}}$	2^{64}	2^{64}	$\mathbf{2^{61}}$	$\mathbf{2^{63}}$	2^{64}	2^{64}
Output bit no.	56	57	58	59	60	61	62	63
Min. data compl.	$\mathbf{2^{61}}$	$\mathbf{2^{63}}$	2^{64}	2^{64}	$\mathbf{2^{61}}$	$\mathbf{2^{63}}$	2^{64}	2^{64}

Distinguishers highlighted in bold represent cases with valid (non-full codebook) data complexity

4.5 Application to Gift

Gift [24] is an SPN lightweight block cipher designed based on revising the PRESENT cipher. It comes in two versions, Gift-64 and Gift-128 with block sizes 64 and 128 bits, respectively. The key size in both versions is 128 bits. The round functions consist of applying 4-bit S-boxes and bit permutation linear layers. The round function iterates 28 and 40 times in Gift-64 and Gift-128, respectively. The round function of Gift-64 is depicted in Fig. 6 which is very similar to the round function of Gift-128, as well (the number of S-Boxes is 32 instead of 16 and the bit permutation layer is over 128 instead of 64 bits).

We applied the proposed method to all the 64 bits of 9^{th} round output of Gift-64. The results are summarized in Table 7. As reported in the previous papers, the new method also shows that there is no integral distinguisher for 10 or more rounds of Gift-64 based on conventional division property.

We also applied Algorithm 2 to Gift-64 with different numbers of active bits. As a result, we achieve an increase of 1 and 2 in the number of balanced bits for integral distinguishers with 62 and 63 active bits, respectively.

4.6 Application to Clyde-128

Clyde-128 [16] is a tweakable SPN block cipher used in Spook authenticated encryption scheme. Clyde-128 updates the 128-bit input state by iterating N_s steps equivalent to $2N_r$ rounds as depicted in Fig. 7. The designers' recommended number of Rounds is 12 (6 Steps). Its round function consists of applying 32 parallel 4-bit S-boxes followed by 2 parallel 64-bit L-boxes (linear layer).

Designers evaluated the division properties for Clyde-128 over r rounds, for $r \in \{1, \ldots, 8\}$ and the longest reported distinguisher is an 8-round integral distinguisher with data complexity 2^{127}. We examined the proposed method on the first output bit of 8-round Clyde, by which we found many integral distinguishers with data complexity 2^{126}. However, these findings are the suboptimum 8-round integral distinguishers, since we terminated the algorithm execution before it completely be executed.

As pointed out in section 3.2, any d-degree monomial in the ANF of the target bit, discovered by Algorithm 1, implies $(n-d)$ distinct integral distinguishers with data complexity 2^{d+1} provided that $d < n-1$. Hence, the 3500 125-degree monomials found in the ANF of the analyzed output, by partial execution of Algorithm 1 on Clyde, introduced a large number of integral distinguishers with data complexity 2^{126}, one of which is brought in Appendix A.

5 Conclusion

In this paper, we proposed a new method for searching integral distinguishers based on conventional division property. In the new method, the ANF of the target output bit is analyzed and the constraints for getting balanced property in the

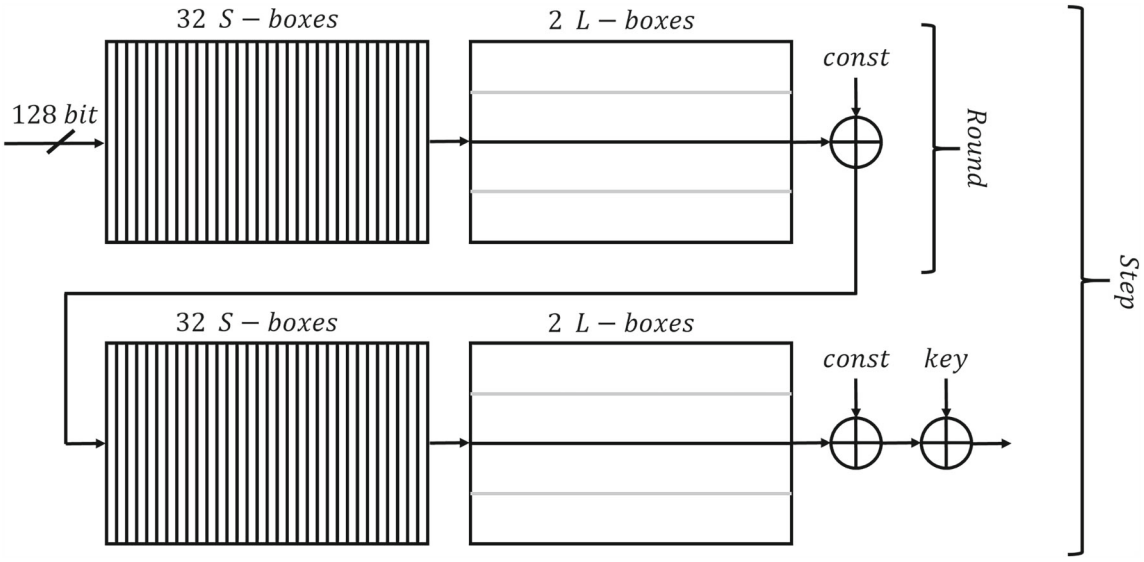

Fig. 7 Round and Step of Clyde-128

target output bit will be extracted. Finally, by optimizing the model consisting of the extracted constraints and the objective function as minimizing the sum of plaintext bits, the minimum data complexity for getting the balanced property will be determined provably in the conventional model. We applied the new method on LBlock, TWINE, SIMON, Gift, Present, and Clyde-128 and reported some improvements on LBlock and Clyde-128. We also discussed the advantages of the new method and introduced its capability to determine the maximum number of balanced bits in a single distinguisher and its application on the studied ciphers.

Data Availability The codes underlying this article are available on GitHub, at: https://github.com/khalesiakram/DivisionMinData.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

A Distinguishers

The integral distinguishers based on the conventional division property of the studied ciphers are summarized as follows, where x_{i_s} and y_{j_t} in $\{x_{i_0}, \ldots x_{i_{n-1}}\} \rightarrow \{y_{j_0}, \ldots y_{j_{n-1}}\}$ imply the constant input and balanced output bits, respectively.

A.1 17-round LBlock

$\{32\} \rightarrow \{2, 3, 30, 31\}$

$\{36\} \rightarrow \{4, 7, 10, 11\}$

$\{40\} \rightarrow \{13, 15, 16, 18\}$

$\{52\} \rightarrow \{22, 23, 25, 27\}$

A.2 16-round LBlock

$\{32, 34\} \rightarrow \{28\}$

A.3 16-round TWINE

$\{0\} \rightarrow \{0-3, 8-11, 16-19, 24-27, 32-35, 40-43, 48-51, 56-59\}$

A.4 SIMON

A.4.1 14-round SIMON32

$\{0\} \rightarrow \{16-31\}$

A.4.2 16-round SIMON48

$\{0\} \rightarrow \{24-47\}$

A.4.3 18-round SIMON64

$\{0\} \rightarrow \{35, 37, 43-63\}$

A.4.4 22-round SIMON96

$\{12\} \rightarrow \{53, 58, 60, 62, 67\}$

A.4.5 26-round SIMON128

$\{12\} \rightarrow \{73, 75, 77\}$

A.5 9-round Gift-64

$\{61, 62, 63\} \rightarrow \{12, 32, 36, 40, 60\}$

A.6 9-round Present

$\{0, 1, 2, 3\} \rightarrow \{63\}$

A.7 8-round Clyde-64

$\{0, 4\} \rightarrow \{0\}$

References

1. Lai, X.: Higher order derivatives and differential cryptanalysis. In Communications and cryptography, pp. 227–233. Springer, (1994)
2. Daemen, J., Knudsen, L., Rijmen, V.: The block cipher square. In International Workshop on Fast Software Encryption, pp. 149–165. Springer, (1997)
3. Knudsen, L., Wagner, D.: Integral cryptanalysis (extended abstract. In Proceedings of Fast Software Encryption–FSE'02, number 2365 in Lecture Notes in Computer Science. Citeseer, (2002)
4. Todo, Y.: Structural evaluation by generalized integral property. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 287–314. Springer, (2015)
5. Xiang, Z., Zhang, W., Bao, Z., Lin, D.: Applying milp method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In International Conference on the Theory and Application of Cryptology and Information Security, pp. 648–678. Springer, (2016)
6. Todo, Y., Morii, M.: Bit-based division property and application to simon family. In International Conference on Fast Software Encryption, pp. 357–377. Springer, (2016)
7. Wang, S., Hu, B., Guan, J., Zhang, K., Shi, T.: Milp-aided method of searching division property using three subsets and applications. In International Conference on the Theory and Application of Cryptology and Information Security, pp. 398–427. Springer, (2019)
8. Hu, K., Wang, M.: Automatic search for a variant of division property using three subsets. In Cryptographers' Track at the RSA Conference, pp. 412–432. Springer, (2019)
9. Hao, Y., Leander, G., Meier, W., Todo, Y., Wang, Q.: Modeling for three-subset division property without unknown subset. vol. 34, pp. 1–69. Springer, (2021)
10. Todo, Y.: Integral cryptanalysis on full misty1. J. Cryptol. **30**(3), 920–959 (2017)
11. Eskandari, Z., Kidmose, A.B., Kölbl, S., Tiessen, T.: Finding integral distinguishers with ease. In International Conference on Selected Areas in Cryptography, pp. 115–138. Springer, (2018)
12. Hu, K., Sun, S., Wang, M., Wang, Q.: An algebraic formulation of the division property: revisiting degree evaluations, cube attacks, and key-independent sums. In International Conference on the Theory and Application of Cryptology and Information Security, pp. 446–476. Springer, (2020)
13. Sun, L., Wang, W., Wang, M.Q.: Milp-aided bit-based division property for primitives with non-bit-permutation linear layers. IET Inf. Secur. **14**(1), 12–20 (2019)
14. Derbez, P., Fouque, P.-A.: Increasing precision of division property. IACR Trans. Symmetr. Cryptol. 173–194 (2020)
15. Khalesi, A., Ahmadian, Z.: Integral analysis of saturnin using bit-based division property. In 2021 18th International ISC Conference on Information Security and Cryptology (ISCISC), pp. 63–67. IEEE, (2021)
16. Bellizia, D., Berti, F., Bronchain, O., Cassiers, G., Duval, S., Guo, C., Leander, G., Leurent, G., Levi, I., Momin, C., et al.: Spook: sponge-based leakage-resistant authenticated encryption with a masked tweakable block cipher. IACR Trans. Symmetr. Cryptol. **2020**, 295–349 (2020)
17. Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (related-key) differential characteristic search: application to simon, present, lblock, des (l) and other bit-oriented block ciphers. In International Conference on the Theory and Application of Cryptology and Information Security, pp. 158–178. Springer, (2014)
18. Optimization, G.: LLC. Gurobi Optimizer Reference Manual, (2021)
19. Bisschop, J.: AIMMS optimization modeling. Lulu. com, (2006)
20. Wu, W., Zhang, L.: Lblock: a lightweight block cipher. In International conference on applied cryptography and network security, pp. 327–344. Springer, (2011)
21. Suzaki, T., Minematsu, K., Morioka, S., Kobayashi, E.: Twine: a lightweight block cipher for multiple platforms. In International Conference on Selected Areas in Cryptography, pp. 339–354. Springer, (2012)
22. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The simon and speck lightweight block ciphers. In Proceedings of the 52nd Annual Design Automation Conference, pp. 1–6. (2015)
23. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C: Present: an ultra-lightweight block cipher. In International workshop on cryptographic hardware and embedded systems, pp. 450–466. Springer, (2007)
24. Banik, S., Pandey, S.K., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y.: Gift: a small present. In International Conference on cryptographic hardware and embedded systems, pp. 321–345. Springer, (2017)