On MILP-based Automatic Search for Bit-Based Division Property for Ciphers with (large) Linear Layers

Muhammad ElSheikh and Amr M. Youssef

Concordia Institute for Information Systems Engineering,
Concordia University, Montréal, Québec, Canada
{m_elshei,youssef}@ciise.concordia.ca

Abstract. With the introduction of the division trail, the bit-based division property (BDP) has become the most efficient method to search for integral distinguishers. The notation of the division trail allows us to automate the search process by modelling the propagation of the DBP as a set of constraints that can be solved using generic Mixed-integer linear programming (MILP) and SMT/SAT solvers. The current models for the basic operations and Sboxes are efficient and accurate. In contrast, the two approaches to model the propagation of the BDP for the non-bit-permutation linear layer are either inaccurate or inefficient. The first approach relies on decomposing the matrix multiplication of the linear layer into `COPY` and `XOR` operations. The model obtained by this approach is efficient, in terms of the number of the constraints, but it is not accurate and might add invalid division trails to the search space, which might lead to missing the balanced property of some bits. The second approach employs a one-to-one map between the valid division trails through the primitive matrix represented the linear layer and its invertible sub-matrices. Despite the fact that the current model obtained by this approach is accurate, it is inefficient, *i.e.,* it produces a large number of constraints for large linear layers like the one of Kuznyechik. In this paper, we address this problem by utilizing the one-to-one map to propose a new MILP model and a search procedure for large non-bit-permutation layers. As a proof of the effectiveness of our approach, we improve the previous 3- and 4-round integral distinguishers of Kuznyechik and the 4-round one of PHOTON's internal permutation (P_{288}). We also report, for the fist time, a 4-round integral distinguisher for Kalyna block cipher and a 5-round integral distinguisher for PHOTON's internal permutation (P_{288}).

Keywords: Bit-based division property · integral · linear layer · MILP · Kuznyechik · Kalyna · PHOTON

1 Introduction

The division property is a generalized integral property that exploits the algebraic degree of the nonlinear components of block ciphers [17]. Since it was

proposed by Todo at Eurocrypt 2015 , it has become one of the most efficient methods to build integral distinguishers.It has been used to analyze the security claims of many symmetric-key primitives, *e.g.,* the full round MISTY1 is broken using a 6-round integral distinguisher found by the division property [18]. The division property was succeeded by a more precise version called the *bit-based division property* (BDP) in [19] which exploits the internal structure of the non-linear components to analyzes block ciphers at the bit level. Even though the BDP is more accurate and can find better integral distinguishers, handling its propagation is computationally intensive. The first search tool utilized the bit-based division property was limited to building integral distinguishers for block ciphers with block size less than 32 bits since the complexity of the search is around $\mathcal{O}(2^n)$ where n is the block size [19].

Xiang *et al.* [20] have overcome the problem of the restriction on the block size by proposing the *division trails*. Using the division trial, the search process for an integral distinguisher can be converted to checking whether a specific division trail exists or not. They also proposed a systematic method to model the propagation rules of the BDP as a set of linear constraints. Hence, the search process can be efficiently automated with the help of generic Mixed Integer Linear Programming (MILP) and SAT solvers. Moreover, Xiang *et al.* provided an accurate model for the propagation of the BDP through the basic operations; `COPY`, `XOR`, and `AND`, in addition to an accurate model for Sboxes. With the help of these models, it is now feasible to look for integral distinguishers for many ciphers that utilize these operations when the used linear layer is a bit-permutation.

For ciphers with non-bit-permutation linear layers, Sun *et al.* [16] proposed a model relying on decomposing the matrix corresponding to the linear layer into its basic operations; `COPY` and `XOR`. We refer to this model through our paper as *Disjointed Representation* and we will provide more details about it in the following sections. The main two advantages of this model are: (i) it is applicable to all kinds of linear layers, and (ii) the number of constraints needed to model the propagation of the BDP is small, precisely, $2n$ where n denotes the size of the matrix input in bits. However, this representation does not model the propagation accurately and might add invalid division trails to the search space which might lead to missing the balanced property of some bits.

Another model for the propagation of the BDP through non-bit-permutation linear layers is presented by Zhang and Rijmen in [21]. They observed that there is a one-to-one map between each valid division trail and one of the invertible sub-matrices of the matrix, M, representing the linear layer. They were able to convert this map to a set of MILP constraints. Unlike the first model provided by [16], the new model is more accurate. However, the number of the MILP constraints grows exponentially with the size of M. Recently, Hu *et al.* partially solved this problem in [10] by utilizing the one-to-one relation to build a model of 4-degree constraints that can be solved using SMT/SAT. The new number of the constraints is proportional to the square of matrix size. Unfortunately, this model is still not suitable for some large linear layers such as the one of Kuznyechik [4].

Our Contributions. In this paper, we propose a new model for the propagation of the BDP through large linear layers. In particular, we utilize the same one-to-one map proposed by Zhang and Rijmen to derive a set of constraints that filter out all non-invertible sub-matrices, part of them during the offline modelling process and the other part on-the-fly during the search process. In order to validate the correctness of our approach, we use our model to reproduce the results of the 4- and 5-round key-dependent integral distinguishers of AES reported in [10]. With the help of our model, we improved the previous 3- and 4-round integral distinguishers of Kuznyechik block cipher and the 4-round one of PHOTON's internal permutation (P_{288}). We also report, for the fist time, a 4-round integral distinguisher for Kalyna block cipher [13] and a 5-round integral distinguisher for PHOTON's internal permutation (P_{288}) [8]. Table 1 summarizes our results.

Table 1: Integral distinguishers for Kuznyechik, Kalyna and PHOTON.

Ciphers	#Rounds	$\log_2$(Data)	Reference
Kuznyechik	3	116*	[2]
	3	56	Section 5.1
	4	127*	[2]
	4	120	Section 5.1
Kalyna-128	4†	64	Section 5.2
	4§	96	Section 5.2
	4‡	62	Section 5.2
PHOTON (P_{288})	4	48	[16]
	4	40	Section 5.3
	5	280	Section 5.3

* Higher-order differential.
† Without pre-whitening operation.
§ With pre-whitening operation.
‡ A key-dependent distinguisher which depends on the 32 least significant bits of the pre-whitening key.

Outline. The rest of this paper is organized as follows. In Section 2 we recall some relevant definitions and revisit the MILP model for the basic operations. In Section 3, we revisit the previous MILP models for the linear layers. Next, we illustrate in details our new model and search approach in Section 4. In Section 5, we show some applications of the new model. Finally, the paper is concluded in Section 6.

2 Preliminaries

2.1 Notations and Definitions

We represent n-bit vectors using bold letters, *e.g.,* $\boldsymbol{u} \in \mathbb{F}_2^n$. The i-th element of $\boldsymbol{u}$ is expressed as u_i and the Hamming weight $hw(\boldsymbol{u})$ is calculated as $hw(\boldsymbol{u}) = \sum_{i=0}^{n-1} u_i$. For a matrix $M \in \mathbb{F}_2^{p \times q}$, we use the notation $M(i,j)$ to represent the element of M located at the i-th row and j-th column, $r_i = M(i,*)$ to represent the i-th row, and $c_j = M(*,j)$ to represent the j-th column of M. Given two q-bit and p-bit vectors $\boldsymbol{u}$ and $\boldsymbol{v}$, we define $M_{\boldsymbol{v},\boldsymbol{u}} \in \mathbb{F}_2^{hw(\boldsymbol{v}) \times hw(\boldsymbol{u})}$ as a sub-matrix of M as follows

$$M_{\boldsymbol{v},\boldsymbol{u}} = [M(i,j)],\ s.t.\ v_i = u_j = 1, \forall\ 0 \le i \le p-1,\ 0 \le j \le q-1$$

Given a q-bit vector $\boldsymbol{u}$, we define $M_{\boldsymbol{u}} \in \mathbb{F}_2^{p \times hw(\boldsymbol{u})}$ as a sub-matrix of M as follows

$$M_{\boldsymbol{u}} = [M(*,j)],\ s.t.\ u_j = 1, \forall\ 0 \le j \le q-1$$

Definition 1 (Bit product function). *For two n-bit vectors $\boldsymbol{x}$ and $\boldsymbol{u}$, the bit product function $\pi_{\boldsymbol{u}}(\boldsymbol{x}) : \mathbb{F}_2^n \to \mathbb{F}_2$ is defined as*

$$\pi_{\boldsymbol{u}}(\boldsymbol{x}) = \prod_{i=0}^{n-1} x_i^{u_i}$$

Definition 2 (Bit-based Division Property[19]). *Let $\mathbb{X}$ be a multiset whose elements take a value of $\mathbb{F}_2^n$. When the multiset $\mathbb{X}$ has the division property $\mathcal{D}_{\mathbb{K}}^n$, where $\mathbb{K}$ denotes a set of n-dimensional vectors whose i-th element takes 0 or 1, it fulfills the following conditions for any $\boldsymbol{u} \in \mathbb{F}_2^n$:*

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

Definition 3 (Division Trail[20]). *Let f denote the round function of an iterated block cipher. Assume that the input multiset to the block cipher has the initial division property $\mathcal{D}_{\{\boldsymbol{k}\}}^n$, and denote the division property after i-round propagation through f by $\mathcal{D}_{\mathbb{K}_i}^n$. Thus, we have the following chain of division property propagations: $\{\boldsymbol{k}\} \overset{\text{def}}{=} \mathbb{K}_0 \xrightarrow{f} \mathbb{K}_1 \xrightarrow{f} \mathbb{K}_2 \xrightarrow{f} \cdots \xrightarrow{f} \mathbb{K}_r$. Moreover, for any vector $\boldsymbol{k}_i^* \in \mathbb{K}_i (i \ge 1)$, there must exist a vector $\boldsymbol{k}_{i-1}^* \in \mathbb{K}_{i-1}$ such that $\boldsymbol{k}_{i-1}^*$ can propagate to $\boldsymbol{k}_i^*$ by the division property propagation rules. Furthermore, for $(\boldsymbol{k}_0, \boldsymbol{k}_1, \ldots, \boldsymbol{k}_r) \in \mathbb{K}_0 \times \mathbb{K}_1 \times \cdots \times \mathbb{K}_r$, if $\boldsymbol{k}_{i-1}$ can propagate to $\boldsymbol{k}_i$ for all $i \in \{1, 2, \ldots, r\}$, we call $(\boldsymbol{k}_0, \boldsymbol{k}_1, \ldots, \boldsymbol{k}_r)$ an r-round division trail.*

Definition 4 (Binary Matrix [10]). *Suppose for a matrix $M' \in \mathbb{F}_{2^m}^{s \times s}$, we represent the element $M'(i,j)$ in M' as a polynomial in the extension field $\mathbb{F}_{2^m} \simeq \mathbb{F}[x]/(f)$, where f is the irreducible polynomial over $\mathbb{F}_2$ with degree m, then we call M' a binary matrix if all such polynomials in M' can only be 0 or 1. Otherwise, M' is called a non-binary matrix.*

2.2 MILP-based Automated Search for Bit-based Division Property

As we mentioned above, the first automated search tool for the bit-based division property was limited to building integral distinguishers for block ciphers with block size less than 32 bits [19]. Then, Xiang *et al.* [20] proposed the *division trails* to solve this problem. In particular, with the help of the division trial, the search process for an integral distinguisher is converted to checking if the division trail $\boldsymbol{k_0} \rightarrow \cdots \rightarrow e_i$ (a unit vector whose i-th element is 1) does exist or not. If it does not exist, then the i-th bit of r-round output is balanced.

In the following, we summarize the MILP constraints that are used to model the propagation rules of the bit-based division property through the basic operations in block ciphers. For more details, we refer the reader to [20,16,5,7].

- Model for COPY: Let $(a) \xrightarrow{\texttt{COPY}} (b_1, b_2, \ldots, b_m)$ denote the division trail through COPY function, where a single bit (a) is copied to m bits. Then, it can be described using the following MILP constraints:

$$a - b_1 - b_2 - \cdots - b_m = 0, \text{ where } a, b_1, b_2, \ldots, b_m \text{ are binary variables.}$$

- Model for XOR: Let $(a_1, a_2, \ldots, a_m) \xrightarrow{\texttt{XOR}} (b)$ denote the division trail through an XOR function, where m bits are compressed to a single bit (b) using an XOR operation. Then, it can be described using the following MILP constraints:

$$a_1 + a_2 + \cdots + a_m - b = 0, \text{ where } a_1, a_2, \ldots, a_m, b \text{ are binary variables.}$$

- Model for Sboxes: The bit-based division property introduced in [19] is limited to bit-orientated ciphers and cannot be applied to ciphers with Sboxes. Xiang *et al.* [20] complemented this work by proposing an algorithm to accurately compute the bit-based division property through an Sbox. Briefly, they represented the Sbox using its algebraic normal form (ANF). Then,the division trail though an n-bit Sbox can be represented as a set of $2n$-dimensional binary vectors $\in \{0,1\}^{2n}$ which has a convex hull. The H-Representation of this convex hull can be computed using readily available functions such as `inequality_generator()` function in SageMath[1] which returns a set of linear inequalities that describe these vectors. We use this set of inequalities as MILP constraints to present the division trail though the Sbox.

3 Previous MILP-based Modelling for Linear Layers

The propagation of the bit-based division property through bit-permutation linear layers, *e.g.,* the linear layers of PRESENT [3] and GIFT [1], can be easily modelled by rearranging the variables based on the permutation. In contrast, the non-bit-permutation linear layers, *e.g.,* the linear layers of AES and Kuznyechik [4], needs a more complex model.

[1] http://www.sagemath.org/

In this section, we revisit the two methods used to model the propagation of the BDP through non-bit-permutation linear layers. These methods relay on representing the matrix multiplication in the linear layer at the bit level. Suppose the linear layer can be represented as a matrix multiplication over the field $\mathbb{F}_{2^m}$ using the matrix $M^{'} \in \mathbb{F}_{2^m}^{s \times s}$. Given the irreducible polynomial of the field $\mathbb{F}_{2^m}$, we can derive a unique equivalent matrix $M \in \mathbb{F}_2^{n \times n}$ called the primitive matrix at the bit level where $n = s \times m$.

3.1 Disjointed Representation

Since the primitive matrix M is presented at the bit level, *i.e.,* $M(i,j) \in \{0,1\}$, we can decompose the linear layer into its basic operations, *i.e.,* `AND` with 0 or 1 and `XOR` operations. Consequently, the propagation of the BDP can be easily modelled using the models of the basic operations [16].

Let $\boldsymbol{u} \xrightarrow{M} \boldsymbol{v}$ denote the division trail through the linear layer where $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{F}_2^n$. By defining a set of auxiliary binary variables $\boldsymbol{t} = \{t_{(i,j)}$ if $M(i,j) = 1, 0 \leq i,j \leq n-1\}$, we can model the propagation of the BDP at the bit level in two steps as follows:

- $(u_j) \xrightarrow{\texttt{COPY}} (t_{(0,j)}, t_{(1,j)}, \ldots, t_{(n-1,j)})$ where

$$-u_j + \sum_{\substack{i=0 \\ M(i,j)=1}}^{n-1} t_{(i,j)} = 0$$

- $(t_{(i,0)}, t_{(i,1)}, \ldots, t_{(i,n-1)}) \xrightarrow{\texttt{XOR}} (v_i)$ where

$$-v_i + \sum_{\substack{j=0 \\ M(i,j)=1}}^{n-1} t_{(i,j)} = 0$$

Hence, the total number of constraints $\#\mathcal{L} = 2n$.

Limitations. Despite the fact that this method is simple and efficient in terms of the number of constraints, it cannot handle the cancellation between monomials since it handles each output bit individually. Hence, it is not precise and it might produce invalid division trails leading to missing the balanced property of some output bits. For further details, See [21].

3.2 Compact Representation

One method to overcome the problem of the monomial cancellations is to deal with the linear layer as a one single block like an S-box. However, the large size of the linear layer renders this approach computationally infeasible in many cases.

In this context, Zhang and Rijmen observed that there is a one-to-one map between the accurate division trails of the primitive matrix M and invertible sub-matrices of M [21]. This observation is stated in the following theorem.

Theorem 1 ([21]). *Let M be the $n \times n$ primitive matrix of an invertible linear transformation and $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{F}_2^n$. Then $\boldsymbol{u} \xrightarrow{M} \boldsymbol{v}$ is one of the valid division trails of the linear transform M if and only if $M_{\boldsymbol{v},\boldsymbol{u}}$ is invertible.*

Using this one-to-one map, they proposed a systematic method to model a binary matrix $M^{'} \in \mathbb{F}_{2^m}^{s \times s}$ as a set of MILP constraints. For more derails, see [21]. In this case, the total number of constraints $\#\mathcal{L} = m \times (2^s - 1)$.

Regarding the non-binary matrices, we can still use the same method, but the number of constrains will exponentially increase with the size of the primitive matrix, *i.e.,* if the primitive matrix M is $n \times n$, then the total number of constraints $\#\mathcal{L} = 2^n - 1$.

Hu *et al.* presented an updated version of Theorem 1 in [10]. They removed the restriction that the primitive matrix M must be invertible to have valid division trails. Consequently, the primitive matrix M could be in general of size $p \times q$. Hence, $\boldsymbol{u} \xrightarrow{M} \boldsymbol{v}$ is one of the valid division trails of M if and only if $M_{\boldsymbol{v},\boldsymbol{u}}$ is invertible where $\boldsymbol{u}$ and $\boldsymbol{v}$ are q- and p-bit vectors, and $hw(\boldsymbol{u}) = hw(\boldsymbol{v})$. Hu *et al.* also utilized this one-to-one map to present a new model for the propagation of the BDP through a non-binary matrix using less number of constraints. If a primitive matrix M is $p \times q$, then the total number of constraints will be $\#\mathcal{L} = p^2$. It should be mentioned that the constraints are 4-degree ones, therefore it is solvable using SMT/SAT solvers and cannot be handled using MILP solvers. For more details, see [10].

Limitations. Even though the models by Zhang and Rijmen, and Hu *et al.* are accurate, they are inefficient for large linear layers, *e.g.,* the primitive matrix corresponding to the linear layer of Kuznyechik is 128×128, therefore we will need 2^{128} or $128^2 = 2^{14}$ constraints to model a single linear layer if we use Zhang and Rijmen and Hu *et al.* methods receptively. Therefore, when the distinguisher covers many rounds, it will be computationally infeasible for current MILP/SAT solvers to handle the model due to the large number of the constraints.

4 MILP-based Modelling for (large) Linear Layers

As mentioned in the previous section, the current models for the non-bit-permutation linear layer in the literature are either inaccurate or inefficient for large linear layers. In this paper, we tackle this problem by proposing an accurate model for the linear layer when its input division property is priorly known before the modelling step. Thereby, this model is more suitable for the first round of the distinguisher. Regarding the other rounds of the distinguisher when the input division property cannot be determined during the modelling, we use the disjointed representation described in Section 3.1 and address its inaccuracy by discarding any invalid trails on-fly during the search process.

4.1 Prior-Known Input Division Property to the Linear Layer

Suppose the primitive matrix M is of size $p \times q$ and let $\boldsymbol{u}$ be the input division property to M and assume it is determined a priori. Consequently, we can utilize Theorem 1 and its updated version in [10] to derive all correct division trails. The naive method to do so is by exhaustively trying all the values of the output division property $\boldsymbol{v}$ such that $hw(\boldsymbol{u}) = hw(\boldsymbol{v})$ and checking if the sub-matrix $M_{\boldsymbol{v},\boldsymbol{u}}$ is invertible. Despite the correctness of this method, we need to try $\binom{p}{hw(\boldsymbol{u})}$ sub-matrices which is a very large number in almost all the cases. Moreover, we have to find a method to encode these division trails as MILP constraints to build a large model that covers many number of rounds. In the following, we explain our main idea to overcome this problem.

Main Idea. Based on Theorem 1, the sub-matrix $M_{\boldsymbol{v},\boldsymbol{u}}$ must be invertible to have a valid trail $\boldsymbol{u} \xrightarrow{M} \boldsymbol{v}$, *i.e.,* the sub-matrix $M_{\boldsymbol{v},\boldsymbol{u}}$ must not include linearly dependant rows. Given the input division property $\boldsymbol{u}$, we can construct the column matrix $M_{\boldsymbol{u}}$. Subsequently, we can get the row echelon form of $M_{\boldsymbol{u}}$ using the Gaussian eliminations, and obtain all the sets of linearly dependent rows. Then, instead of checking each value of $\boldsymbol{v}$ (as in the naive method), we derive a set of constraints that guarantee the bits v_i do not lead to including any set of linearly dependent rows from $M_{\boldsymbol{u}}$. In order to complete the model, one more constraint should be added to enforce $hw(\boldsymbol{u}) = hw(\boldsymbol{v})$. Hence, the value of $\boldsymbol{v}$ that satisfies these constraints is indeed a valid output division property.

The following examples illustrates our idea.

Detailed Example. Assume a toy linear layer where its primitive matrix M is 8×8. Given the input division property $\boldsymbol{u} = (1, 1, 1, 1, 1, 0, 0, 0)$, we can construct the column matrix $M_{\boldsymbol{u}}$ by choosing the columns of M that correspond to the nonzero bits in $\boldsymbol{u}$.

$$M = \begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 1&1&0&0&1&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&1&0&0&1&1&0&0 \\ 1&1&0&0&0&0&1&0 \\ 0&0&1&1&0&0&0&1 \end{bmatrix} \xrightarrow{\boldsymbol{u}} M_{\boldsymbol{u}} = \begin{bmatrix} 1&0&0&0&0 \\ 1&1&0&0&1 \\ 0&1&0&0&0 \\ 0&0&1&0&0 \\ 0&0&0&1&0 \\ 0&1&0&0&1 \\ 1&1&0&0&0 \\ 0&0&1&1&0 \end{bmatrix}$$

We follow the procedure given below to derive a set of linear constraints as a function in the output division property $\boldsymbol{v} = (v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7)$ to trace the propagation of the division property for $M_{\boldsymbol{u}}$.

1. Check whether $Rank(M_{\boldsymbol{u}}) = hw(\boldsymbol{u})$ to ensure that there is at least one full rank (invertible) sub-matrix, and hence at least one valid division trail. Otherwise, we conclude that $\boldsymbol{u}$ cannot be propagated to any valid $\boldsymbol{v}$.

2. Use Gaussian eliminations to put $M_{\boldsymbol{u}}$ in its row echelon form while keeping track the row operations. Hence, each all-zero row in the row echelon form implies a set of linearly dependent rows in the original matrix $M_{\boldsymbol{u}}$, *e.g.,* the first all-zero row in our example can be expressed as $r_0 + r_1 + r_5 = \mathbf{0}$ which means that the rows $\{r_0, r_1, r_5\}$ from $M_{\boldsymbol{u}}$ are linearly dependent. The details of the Gaussian eliminations for our example can be found in Appendix A.

$$\left[\begin{array}{ccccc|c} 1&0&0&0&0&r_0\\ 1&1&0&0&1&r_1\\ 0&1&0&0&0&r_2\\ 0&0&1&0&0&r_3\\ 0&0&0&1&0&r_4\\ 0&1&0&0&1&r_5\\ 1&1&0&0&0&r_6\\ 0&0&1&1&0&r_7 \end{array}\right] \xrightarrow[Elimination]{Gaussian} \left[\begin{array}{ccccc|l} 1&0&0&0&0&r_0\\ 0&1&0&0&1&r_1+r_0\\ 0&0&1&0&0&r_3\\ 0&0&0&1&0&r_4\\ 0&0&0&0&1&r_2+r_1+r_0\\ 0&0&0&0&0&r_5+r_1+r_0\\ 0&0&0&0&0&r_6+r_2+r_0\\ 0&0&0&0&0&r_7+r_3+r_4 \end{array}\right] \rightarrow \left\{\begin{array}{l} r_0+r_1+r_5=\mathbf{0}\\ r_0+r_2+r_6=\mathbf{0}\\ r_3+r_4+r_7=\mathbf{0} \end{array}\right\}$$

In general, if $M_{\boldsymbol{u}}$ is $p \times hw(\boldsymbol{u})$, then there are $p - hw(\boldsymbol{u})$ all-zero rows in the row echelon form given that $Rank(M_{\boldsymbol{u}}) = hw(\boldsymbol{u})$.

3. Find all the sets of linearly dependent rows. We do so by trying the combinations between the relations derived from all-zero rows obtained in the previous step, *e.g.,* combine $r_0+r_1+r_5 = \mathbf{0}$ and $r_0+r_2+r_6 = \mathbf{0}$ will produce $r_0+r_1+r_5+r_0+r_2+r_6 = \mathbf{0} \Rightarrow r_1+r_2+r_5+r_6 = \mathbf{0}$ which means the rows $\{r_1, r_2, r_5, r_6\}$ are linearly dependent.

$$\left\{\begin{array}{r} r_0+r_1+r_5=\mathbf{0}\\ r_0+r_2+r_6=\mathbf{0}\\ r_3+r_4+r_7=\mathbf{0}\\ r_1+r_2+r_5+r_6=\mathbf{0}\\ r_0+r_1+r_3+r_4+r_5+r_7=\mathbf{0}\\ r_0+r_2+r_3+r_4+r_6+r_7=\mathbf{0}\\ r_1+r_2+r_3+r_4+r_5+r_6+r_7=\mathbf{0} \end{array}\right.$$

4. For each set of linearly dependent rows, we derive a constraint on some bits of $\boldsymbol{v}$ enforcing any selected sub-matrix to be invertible, *e.g.,* $r_0 + r_1 + r_5 = \mathbf{0}$ means the rows $\{r_0, r_1, r_5\}$ are linearly dependent. In other words, these rows together must not be a part of any sub-matrix in order to have valid trails. Reflecting on $\boldsymbol{v}$, this means the bits v_0, v_1, v_5 cannot be 1 at the same time. We can represent this relation as a linear constrain $v_0 + v_1 + v_5 \leq 2$. The initial model for our toy linear layer includes:

$$\begin{cases} v_0 + v_1 + v_5 \leq 2 & \text{(C1)} \\ v_0 + v_2 + v_6 \leq 2 & \text{(C2)} \\ v_3 + v_4 + v_7 \leq 2 & \text{(C3)} \\ v_1 + v_2 + v_5 + v_6 \leq 3 & \text{(C4)} \\ v_0 + v_1 + v_3 + v_4 + v_5 + v_7 \leq 5 & \text{(C5)} \\ v_0 + v_2 + v_3 + v_4 + v_6 + v_7 \leq 5 & \text{(C6)} \\ v_1 + v_2 + v_3 + v_4 + v_5 + v_6 + v_7 \leq 6 & \text{(C7)} \\ v_0, \ldots, v_7 \text{ are binary variables} \end{cases}$$

5. Remove the redundancy constraints, *e.g.,* the constraint C5 is redundant because if the constraints C1 and C3 are satisfied, then the constraint C5 is satisfied. Also, if the constraints C1 and C3 are not satisfied, then the constraint C5 is not satisfied. In contrast, if one of the constraints C1 and C3 is satisfied and the other is not satisfied, the solution will be rejected even though the constraint C5 is satisfied.
   We can automate this step by checking if a set of dependent rows (A) is a sub-set of another set of dependent rows (B), then the constraint on the set B is redundant. The model for our toy linear layer is then reduced to:

$$\begin{cases} v_0 + v_1 + v_5 \leq 2 \\ v_0 + v_2 + v_6 \leq 2 \\ v_3 + v_4 + v_7 \leq 2 \\ v_1 + v_2 + v_5 + v_6 \leq 3 \\ v_0, \ldots, v_7 \text{ are binary variables} \end{cases}$$

6. Finally, add a constraint to enforce that $hw(\boldsymbol{u}) = hw(\boldsymbol{v})$. The model for our toy linear layer will be

$$\begin{cases} v_0 + v_1 + v_5 \leq 2 \\ v_0 + v_2 + v_6 \leq 2 \\ v_3 + v_4 + v_7 \leq 2 \\ v_1 + v_2 + v_5 + v_6 \leq 3 \\ v_0 + v_1 + \cdots + v_7 = 5 \\ v_0, \ldots, v_7 \text{ are binary variables} \end{cases}$$

Number of Constraints. Although we cannot count exactly the number of the required constraints before performing the procedure, we can give the upper bound of the number based on Step 3 as follows:

$$\#\mathcal{L} \leq 1 + \sum_{i=1}^{p-hw(\boldsymbol{u})} \binom{p - hw(\boldsymbol{u})}{i} = 1 + 2^{p-hw(\boldsymbol{u})} - 1 = 2^{p-hw(\boldsymbol{u})}$$

In the light of this upper bound, it is clear that the model is practically more applicable when $p - hw(\boldsymbol{u})$ is relatively small which is usually the case for the linear layer at the first round when we search for a distinguisher that covers a large number of rounds where the Hamming weight of the input division property of the distinguisher (the number of active bits) is very close to the block size.

4.2 Complete Model and Search Approach

In the previous section, we presented a model for the linear layer at the first round when its input division property is known before the modelling step. In this section, we propose a search approach allowing us to use that model even though the targeted distinguisher does not start from a linear layer. We also complete the model for the targeted distinguisher by showing how to handle the intermediate linear layers.

Intermediate Linear Layers. We use the disjointed representation described in Section 3.1 to model the intermediate linear layers. When a candidate division tail is obtained by solving the complete model, we then extract the values of the input and the output division property of each matrix multiplication in the trail. After that, we check whether $M_{\boldsymbol{v},\boldsymbol{u}}$ is invertible or not for each matrix multiplications. If one of them is not invertible, we discard the trail by updating the model through adding a special craft constraint and resolving the updated model.

Discarding Invalid Trails. Let $(u_0, \ldots, u_{n-1})$ and $(v_0, \ldots, v_{n-1})$ be the variables in the model representing the input and the output division property of a matrix multiplication where $M_{\boldsymbol{v},\boldsymbol{u}}$ is not invertible in the current solution of the model. Let I_0^u (I_1^u) be the indices of $\boldsymbol{u}$'s variables that equal to `0` (`1`) in the current solution. Similarly, let I_0^v (I_1^v) be the indices of $\boldsymbol{v}$'s variables that equal to `0` (`1`) in the current solution. We update the model based on the current solution by adding the following constraint

$$\sum_{i \in I_0^u} (u_i) + \sum_{i \in I_1^u} (1 - u_i) + \sum_{i \in I_0^v} (v_i) + \sum_{i \in I_1^v} (1 - v_i) \geq 1$$

Therefore, when we attempt to resolve the updated model, the current solution, *i.e.,* the invalid trial, will violate the new constraint and the solver will not consider it as a solution and try to obtain another solution.

Implementation. Although the models for both the first linear layer with known input division property and the intermediate linear layers with the discarding approach above are applicable using MILP and SMT/SAT, the approach to discard invalid trails is more efficient using MILP solvers via the callback function and the concept of lazy constraints [9,11] without needing to resolve the model from scratch.

Last Linear Layer. When the distinguisher ends with a linear layer, we can model it using the disjointed representation (like the intermediate linear layers) or we can efficiently model it using the model for XOR operation. Let $(u_0, \ldots, u_{n-1})$ and $(v_0, \ldots, v_{n-1})$ be the variables in the model which represent the input and the output division property of the matrix multiplication in the last linear layer. Suppose we check if there is a division trail from the input division property of the distinguisher to the unit vector e_i, *i.e.,* checking if the i-th bit of the output is balanced or not. Therefore, the variables that represent the output division property will be set to

$$\begin{cases} v_i = 1 \\ v_l = 0, \qquad 0 \leq l \leq n-1, l \neq i \end{cases}$$

Consequently, during modelling, we focus on row $r_i = M(i, *)$ of the primitive matrix M and the constraints on the input division property of the matrix multiplication will be

$$\begin{cases} \sum\limits_{\substack{j=0 \\ M(i,j)=1}}^{n-1} u_j = 1 \\ u_j = 0, \qquad 0 \leq j \leq n-1, M(i,j) = 0 \end{cases}$$

After solving the model, if there is a division trail from the input division property of the distinguisher to the unit vector e_i, we conclude that there are other division trails from the same input division property of the distinguisher to other unit vectors without creating/solving their corresponding models. The original division trial can be split into two sub-trails; from the input division property of the distinguisher to the input division property of the last linear layer $\boldsymbol{u}$, and from $\boldsymbol{u}$ to the unit vector e_i where $hw(\boldsymbol{u}) = hw(\boldsymbol{v}) = 1$, *i.e.,* only one variable from $(u_0, \ldots, u_{n-1})$ is 1 and the other are 0. Suppose this variable is u_j. Therefore, the column matrix $M_{\boldsymbol{u}}$ can be created from a single column $c_j = M(*, j)$. Based on Theorem 1, the division trail from the input division property of the distinguisher to the unit vector e_l, passing through $\boldsymbol{u}$, exists for the l-th output bit if $M(l, j) = 1$ where $0 \leq l \leq n-1$.

Search Approach. If the targeted distinguisher starts from a linear layer, the input division property of this linear layer is known and we can use the model described in Section 4.1. Hence, we create only one model for the distinguisher. Otherwise, we perform the following search approach:

1. We firstly determine all the possible values of the input division property of the first linear layer by propagating the input division property of the distinguisher through other parts of the first round, which is usually a non-linear layer of Sboxes.
2. Then, we check the i-th output bit by creating a group of sub-models starting from the first linear layer with different input division property, thereby, we

can employ the model described in Section 4.1 for the first linear layer in each sub-model.
3. Finally, we solve the sub-models independently in parallel by dividing our computational power between them. If the valid division trail that ends at the unity vector e_i exists for a sub-model, we terminate the search process for the other sub-models. If it does not exist for all sub-models, then the i-th output bit is balanced. The last two steps are repeated for all output bits.

Remark. Even though the model for the linear layer using the disjointed representation with discarding invalid trails approach is applicable to the first linear layer, we believe that modelling the first linear layer accurately from the beginning is important. Our reasoning for that is as follows. First, the Hamming weight of the input/output division property for the first linear layer is the highest compared to the successive linear layers, *i.e.,* the number of its possible propagation is high and the chance to find invalid sub-trails will increase, which leads to the second reason. Since every sub-trail in early rounds is branched to many trails in the successive rounds, invalid sub-trails in the first round have a larger effect on expanding the search space, and hence increasing the time of solving the model. We verified our hypothesis experimentally by comparing the running time to find the 4-round key-dependent integral distinguish of AES reported in [10] using the same platform in the two cases; the case when the first linear layer is modelled accurately from the beginning and the other case when we model the first linear layer using the disjointed representation with discarding approach. In the first case, the solver found the distinguisher in around 50 minutes. In contrast, the solver did not finish in the second case even after running for more than a day.

5 Applications of our New Approach

In this section, we report our findings when applying our approach to Kuznyechik and Kalyna block ciphers and a variant of PHOTON permutations. We also have reproduced the results of the 4- and 5-round dependent-key integral distinguishers of AES reported in [10].

During our experiments, We use either Gurobi[2] solver [9] or the CPLEX optimizer [11] to solve the models. Our source codes are available at https://github.com/mhgharieb/MILP_DivisionProprerty_LinearLayer

We use the following notation to present the integral property of each byte in the plaintext and ciphertext:

- $\mathcal{C}$: Each bit of the byte at the plaintext is fixed to constant.
- $\mathcal{A}$: All bits of the byte at the plaintext are active.

[2] We use the version of Groubi that has some problems reported in [6]. Therefore, when we find some balanced bits by solving a model using Gurobi and we could not verify this results by propagating the traditional integral property, we resolve the model again using the CPLEX optimizer in order to validate the results.

- $\mathcal{B}$: Each bit of the byte at the ciphertext is balanced (the XOR sum is zero).
- $\mathcal{U}$: A byte at the ciphertext with unknown status (the XOR sum is unknown).

When each bit of a byte has a different property, we use lowercase letters to present the property, *i.e.,* `c`, `a` and `b` will represent a constant bit, an active bit, and a balanced bit, respectively. For example, `caaaaaaa` represents a byte where the most significant bit is constant and the other bits are active.

In general during our experiments, when an R-round distinguisher is found, we follow two different paths in parallel as a next step; we examine whether $(R+1)$-round exists or not, and we try to find another R-round distinguisher that needs a less number of active bits, *i.e.,* less data complexity.

5.1 Application to Kuznyechik

The Russian encryption standard — Kuznyechik [4,14], also known as GOST 34.12-2015, is a 9-round SPN-based block cipher with a 128-bit block size and 256 bits of key. The encryption procedure is performed as follows. After loading a block of 128-bit plaintext to a 16-byte internal state $\boldsymbol{x} = (x_0, \ldots, x_{15})$ where x_0 is the least significant byte, the state is Xored with a whitening round key (XOR Layer (X)). Then, the state is updated 9 times using an identical round function denoted as $R = (X \circ L \circ S)$ that consists of:

- Non-linear Layer (S): Each byte of the state is mapped using 8-bit Sbox.
- Linear Layer (L): The 16-byte state is multiplied by 16×16 MDS matrix over the field $\mathbb{F}_{2^8}$ with the irreducible polynomial $X^8 + X^7 + X^6 + X + 1$.
- XOR Layer (X): The 16-bye state is Xored with the corresponding round key.

In [2], Biryukov *et al.* studied Kuznyechik security against the multiset-algebraic cryptanalysis in which they reported the 3- and 4-round integral distinguisher based on their algebraic degree.

3-round Integral Distinguishers. Biryukov *et al.* reported that the 3-round has degree at most 116 [2]. Therefore the XOR sum over a set of plaintexts with dimension 117 will be zero, *i.e.,* the 3-round integral distinguisher exists with the data complexity of 2^{117}. However, we found several 3-round integral distingushers with a much lower data complexity of 2^{56}. One of these distinguishers is as follows.

$$\begin{gathered}(\mathcal{C},\mathcal{C},\mathcal{C},\mathcal{C},\mathcal{C},\mathcal{C},\mathcal{C},\mathcal{C},\mathcal{C},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A}) \\ \Downarrow 3R \circ X \\ (\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B})\end{gathered}$$

4-round Integral Distinguishers. Biryukov *et al.* also reported a 4-round distinguisher with the data complexity of 2^{127} depending on the 4-round has degree at most 126 [2]. In our experiments, we were able to find several 4-round integral distinguishers with data complexity of 2^{120} (120 active bits). One of these distingushers is as follows.

$$(\mathcal{C}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A})$$
$$\Downarrow 4R \circ X$$
$$(\mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B})$$

Other Experiments. Biryukov *et al.* extended the 4-round key-independent integral distinguisher to a 5-round key-dependent one with the same data complexity by appending the linear layer (L) before the 4-round one. The new distinguisher depends on the least significant byte of the master key. We were able to verify the existence of this distinguisher using our model by setting one bit to a constant and the other bits to active as shown below.

$$(\texttt{caaaaaaa}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A})$$
$$\Downarrow 4R \circ X \circ L$$
$$(\mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B})$$

As a next step, we employ the search approach proposed in the previous section to check the existence of the 5-round key-independent distinguisher with a single bit constant and 127 bits active, and we confirmed that this distinguisher does not exist even with the use of the accurate propagation of the BDP.

5.2 Application to Kalyna

The Ukrainian standard Kalyna [13], also known as DSTU 7624:2014, is a family of five SPN-based block ciphers denoted as Kalyna-l/k where $l, k \in \{128, 256, 512\}$ are the block size and the key size, respectively, such that $k = l$ or $k = 2 \times l$. The number of rounds depends on the key size.

We targeted the two members with the block size of 128 bits, Kalyna-128. The encryption procedure is performed as follows. The 16 bytes of the plaintext block $\boldsymbol{x} = (x_0, \ldots, x_{15})$ where x_0 is the least significant byte, is loaded to the 8×2 16-byte state matrix in column-wise order. After that, pre-whitening round key is added to each column independently using addition modulo 2^{64}. We denote this operation as ($\boxplus_{64}$). Then, The following round function denoted as $R = (X \circ L \circ SR \circ S)$ is iterated 10 or 14 times depending on the key size:

- Non-linear Layer (S): 4 different 8-bit Sboxes $\pi_s, s \in \{0, 1, 2, 3\}$ are used to map the bytes of the state matrix where the i-th byte (x_i) is substituted by $\pi_{i \mod 4}(x_i)$.
- ShiftRows (SR): The bytes of each row in the state matrix are cyclically shifted to right by $\lfloor \frac{i}{4} \rfloor$ where $i, 0 \leq i \leq 7$ is the row index.
- Linear Layer (L): Each 8-byte column of the state matrix is independently multiplied by 8×8 MDS matrix over the field $\mathbb{F}_{2^8}$ with the irreducible polynomial $X^8 + X^4 + X^3 + X^2 + 1$.
- XOR Layer (X): the state matrix is Xored with the corresponding round key.

In the last round, the XOR Layer (X) is replaced by a post-whitening modular key addition modulo 2^{64}.

4-round Integral Distinguishers without pre-whitening. During our experiments, we found two 4-round integral distinguisher starting after the pre-whitening step with 8 active bytes as depicted below. The correctness of these distinguishers can be easily verified by propagating the integral properties though the equivalent structure of the round function. Given that, each 8-bit Sbox is reused every 4 bytes and the first (second) 4 rows of the state matrix is shifted by the same step, the state matrix can be reconstructed as 2×2 matrix such that each 4 successive bytes are concatenated in a 32-bit word and the 4 different 8-bit Sboxes build a 32-bit super Sbox. Therefore, when the diagonal (anti-diagonal) words of the new state matrix are active, *i.e.,* take all possible values from $\mathbb{F}_{2^{32}}^2$, the output after 4-rounds will be balanced similar to the 4-round integral distinguisher of AES [12].

$$\begin{bmatrix} \mathcal{C} & \mathcal{A} \\ \mathcal{C} & \mathcal{A} \\ \mathcal{C} & \mathcal{A} \\ \mathcal{C} & \mathcal{A} \\ \mathcal{A} & \mathcal{C} \\ \mathcal{A} & \mathcal{C} \\ \mathcal{A} & \mathcal{C} \\ \mathcal{A} & \mathcal{C} \end{bmatrix} \text{OR} \begin{bmatrix} \mathcal{A} & \mathcal{C} \\ \mathcal{A} & \mathcal{C} \\ \mathcal{A} & \mathcal{C} \\ \mathcal{A} & \mathcal{C} \\ \mathcal{C} & \mathcal{A} \\ \mathcal{C} & \mathcal{A} \\ \mathcal{C} & \mathcal{A} \\ \mathcal{C} & \mathcal{A} \end{bmatrix} \xRightarrow{4R} \begin{bmatrix} \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} \end{bmatrix} \xrightarrow{Appending\ \boxplus_{64}} \begin{bmatrix} \mathcal{C} & \mathcal{A} \\ \mathcal{C} & \mathcal{A} \\ \mathcal{C} & \mathcal{A} \\ \mathcal{C} & \mathcal{A} \\ \mathcal{A} & \mathcal{A} \\ \mathcal{A} & \mathcal{A} \\ \mathcal{A} & \mathcal{A} \\ \mathcal{A} & \mathcal{A} \end{bmatrix} \text{OR} \begin{bmatrix} \mathcal{A} & \mathcal{C} \\ \mathcal{A} & \mathcal{C} \\ \mathcal{A} & \mathcal{C} \\ \mathcal{A} & \mathcal{C} \\ \mathcal{A} & \mathcal{A} \\ \mathcal{A} & \mathcal{A} \\ \mathcal{A} & \mathcal{A} \\ \mathcal{A} & \mathcal{A} \end{bmatrix} \xRightarrow{4R \circ \boxplus_{64}} \begin{bmatrix} \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} \end{bmatrix}$$

4-round Integral Distinguishers with pre-whitening. We were able to extend each of the previous 4-round distinguishers to cover the pre-whitening operation. The new distinguishers need 12 active bytes as depicted above. In the following, we illustrate the way we use to select a set of plaintexts so that it satisfies the input division property of the 4-round distinguisher after applying the pre-whitening operation.

Since the pre-whitening operation is performed per column, we focus on each column independently. Suppose X, Y, and K denote a 64-bit word of the input, the output and the whitening key, respectively, such that $Y = X \boxplus_{64} K$. Each 64-bit word can be considered as the concatenation of two 32-bit words, *i.e.,* $X = X_l || X_r$, $Y = Y_l || Y_r$, and $K = K_l || K_r$. Therefore, $Y_r = X_r \boxplus_{32} K_r$ and $Y_l = X_l \boxplus_{32} K_l \boxplus_{32} C$ where $\boxplus_{32}$ denotes the addition modulo 2^{32} and C is the carry from the first addition part.

Consequently, a set of plaintexts such that X_r is fixed to constant and the 4 bytes of X_l takes all the possible values from $\mathbb{F}_{2^8}^4$, will give an output set such that Y_r will be constant and the 4 bytes of Y_r will take all the possible values from $\mathbb{F}_{2^8}^4$. This is because the whitening key is constant and the carry will be fixed over all the set's elements based on the previous two questions. As the result, we can easily satisfy one of the two column in the 4-round distinguishers.

The same method cannot be applied to the other column because if X_r takes all the possible values, Y_r will take all the possible values, but, the value of the carry will change depending on the value of the whitening key. Hence, we cannot adapt the values of X_l to enforce Y_l to be fixed over the set. To overcome this problem, we construct a set of plaintexts such that the 8 bytes

of X take all the possible values from $\mathbb{F}_{2^8}^8$, hence, the 8 bytes of Y will take all the possible values from $\mathbb{F}_{2^8}^8$. As the result, the output set Yt can be considered as 2^{32} sub-sets in which each sub-set satisfies the input division property of the other column of the 4-round distinguisher. Combining these two approaches, the 4-round distinguishers with the pre-whitening need 12 active bytes.

Using the BDP, we are able to verify the existence of these distingushers with the help of the propagation model of the BDP through modular addition with a constant proposed in [5]. Additionally, we have tried to reduce the number of active bits by iterating over the active bits one-by-one and set it to constant then check if the distinguisher still exists. Unfortunately, the distinguisher does not exist.

Other Experiments. During our experiments, we build a 4-round key-dependent distinguisher using 62 active bits. The new distinguisher depends on the 32 least significant bits of the pre-whitening key. The distinguisher starts at the linear layer of the first round with the input division property as shown below.

$$\begin{bmatrix} \texttt{ccaaaaaa} & \mathcal{C} \\ \mathcal{A} & \mathcal{C} \\ \mathcal{A} & \mathcal{C} \\ \mathcal{A} & \mathcal{C} \\ \mathcal{C} & \mathcal{A} \\ \mathcal{C} & \mathcal{A} \\ \mathcal{C} & \mathcal{A} \\ \mathcal{C} & \mathcal{A} \end{bmatrix} \xrightarrow{3R \circ X \circ L \circ SR} \begin{bmatrix} \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} \end{bmatrix}$$

5.3 Application to PHOTON

PHOTON [8] is a family of lightweight hash functions proposed by Guo *et al.* at CRYPTO 2011 and it has been standardized in ISO/IEC 29192-5:2016. PHOTON has 5 variants with 5 internal unkeyed permutations denoted as P_t where $t \in \{100, 144, 196, 256, 288\}$ is the internal state size. We target here the internal permutation P_{288}. The structure of the internal permutation follows the structure of AES where the internal state is represented as a $d \times d$ square matrix of cells. Thus, the internal state of P_{288} is a 6×6 matrix of bytes. Its round function consists of:

- AddConstants (X): Each byte of the 1st column of the state matrix is Xored with a round-dependent constant.
- SubCells (S): Each byte (x_i) of the state is substituted by $Sbox(x_i)$ where $Sbox$ is the 8-bit Sbox of AES.
- ShiftRows (SR): The bytes of each row in state are cyclically shifted to left by i where $i \in 0 \leq i \leq 5$ is the row index.
- MixColumnsSerial (L): Each column of the state is independently multiplied by 6×6 MDS matrix over $\mathbb{F}_{2^8}$ with the irreducible polynomial $X^8 + X^4 + X^3 + X + 1$.

3- and 4-round Integral Distinguishers. Since the permutation is followed the AES structure, there are 3- and 4-round distinguishers that exploit the structure itself and independent on the used Sboxes and the MDS matrix. In particular, when the state matrix has a single byte active and the other bytes are constant (the data complexity is 2^8), each output bit after 3 rounds will have zero-sum (balanced). Also, there is a 4-round distinguisher when all diagonal's bytes of the state matrix are active (the data complexity is 2^{48}). In [16], Sun *et al.* verified the existence of these 3- and 4-round distinguishers using the MILP models for the propagation of the BDP. They have modelled the linear layer using the disjointed representation.

New 4-round Integral Distinguisher. At Crypto 2016, Sun *et al.* exploited a specific property of the matrix used in AES to introduce the first 5-round key-dependent integral distinguisher [15]. This property is that each column of the matrix has two equal elements. We employ a similar property to reduce the date complexity of the 4-round distinguisher of P_{288} and build a new 5-round one.

Suppose M_P and M_P^{-1} denote the matrix and its inverse that are used in P_{288} where

$$M_P = \begin{bmatrix} 02 & 03 & 01 & 02 & 01 & 04 \\ 08 & 0e & 07 & 09 & 06 & 11 \\ 22 & 3b & 1f & 25 & 18 & 42 \\ 84 & e4 & 79 & 9b & 67 & 0b \\ 16 & 99 & ef & 6f & 90 & 4b \\ 96 & cb & d2 & 79 & 24 & a7 \end{bmatrix}, \quad M_P^{-1} = \begin{bmatrix} 15 & 50 & eb & 62 & 79 & 99 \\ 29 & a5 & c9 & c2 & fb & 2b \\ 56 & 54 & 8e & 9f & e9 & 57 \\ ae & af & 03 & 20 & c8 & ae \\ 47 & 47 & 01 & 44 & 8e & 46 \\ 8c & 8d & 01 & 8d & 02 & 8d \end{bmatrix}$$

Suppose $\boldsymbol{x} = (x_0, x_1, x_2, x_3, x_4, x_4)^T$ and $\boldsymbol{y} = (y_0, y_1, y_2, y_3, y_4, y_5)^T$ be the input and the output vectors to the matrix M_P such that $\boldsymbol{y} = M_P \times \boldsymbol{x}$. Suppose $\boldsymbol{x}$ take $2^{5\times 8=40}$ values where each of x_0, x_1, x_2, x_3 and x_4 take all the possible values from $\mathbb{F}_{2^8}$. Therefore, $\boldsymbol{y}$ will take 2^{40} values. Also, $\boldsymbol{x} = M_P^{-1} \times \boldsymbol{y}$ can be expressed as shown below

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_4 \end{bmatrix} = \begin{bmatrix} 15 & 50 & eb & 62 & 79 & 99 \\ 29 & a5 & c9 & c2 & fb & 2b \\ 56 & 54 & 8e & 9f & e9 & 57 \\ ae & af & 03 & 20 & c8 & ae \\ 47 & 47 & 01 & 44 & 8e & 46 \\ 8c & 8d & 01 & 8d & 02 & 8d \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix}$$

Hence, we can express x_4 as follows in equations (1) and (2).

$$x_4 = 47 \cdot y_0 \oplus 47 \cdot y_1 \oplus 01 \cdot y_2 \oplus 44 \cdot y_3 \oplus 8e \cdot y_4 \oplus 46 \cdot y_5 \quad (1)$$

$$x_4 = 8c \cdot y_0 \oplus 8d \cdot y_1 \oplus 01 \cdot y_2 \oplus 8d \cdot y_3 \oplus 02 \cdot y_4 \oplus 8d \cdot y_5 \quad (2)$$

$$00 = cb \cdot y_0 \oplus ca \cdot y_1 \oplus 00 \cdot y_2 \oplus c9 \cdot y_3 \oplus 8c \cdot y_4 \oplus cb \cdot y_5 \quad (3)$$

From (1) and (2), we can derive the equation (3) which implies that $\{y_0, y_1, y_3, y_4, y_5\}$ are linearly dependent, *i.e.,* they can take at most $2^{4\times 8=32}$ values. Since $\boldsymbol{y}$ takes

2^{40} values, y_2 must take 2^8 values, *i.e.,* y_2 is an active bye and takes its all possible values ($\mathcal{A}$).

Constructing 4-round Integral Distinguisher. We construct a set of 2^{40} chosen plaintexts such that the state matrix is as follows. The first 4 elements of the diagonal are active, the last two elements of the diagonal are equal and active (denoted as $\bar{\mathcal{A}}$), and the other elements of the state matrix are fixed to constant as shown below. After applying the three operations: AddConstants (X), SubCells (S), and ShiftRows (SR), the first column of the state matrix will be in the form of the vector $\boldsymbol{x}$. Therefore, the output set, after applying the MixColumnsSerial (L) operation (a full round from the input set), can be divided into 2^{32} subset so that each has one active byte and the other are constant. Consequently, after another 3 rounds, each bit of the output will have zero-sum as mentioned previously in the 3-round distinguisher section.

$$\begin{bmatrix} \mathcal{A} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} \\ \mathcal{C} & \mathcal{A} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} \\ \mathcal{C} & \mathcal{C} & \mathcal{A} & \mathcal{C} & \mathcal{C} & \mathcal{C} \\ \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{A} & \mathcal{C} & \mathcal{C} \\ \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \bar{\mathcal{A}} & \mathcal{C} \\ \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \bar{\mathcal{A}} \end{bmatrix} \xrightarrow{SR \circ S \circ X} \begin{bmatrix} \mathcal{A} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} \\ \mathcal{A} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} \\ \mathcal{A} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} \\ \mathcal{A} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} \\ \bar{\mathcal{A}} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} \\ \bar{\mathcal{A}} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} \end{bmatrix} \xRightarrow{L} 2^{32} \times \left\{ \begin{bmatrix} \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} \\ \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} \\ \mathcal{A} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} \\ \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} \\ \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} \\ \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} \end{bmatrix} \xRightarrow{3R} \begin{bmatrix} \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} \end{bmatrix} \right\}$$

MILP for the New 4-round Distinguisher. Our model can be started at the MixColumnsSerial (L) operation of the first round, therefore, we can use the accurate model for the propagation of the BDP described in Section 4.1. The first column of the state matrix (in the form of $\boldsymbol{x}$) will be multiplied by M_P. Since the last two element of the vector $\boldsymbol{x}$ are equal, we can express the multiplication operation $\boldsymbol{y} = M_P \times \boldsymbol{x}$ as $\boldsymbol{y} = \hat{M}_P \times \hat{\boldsymbol{x}}$ where $\hat{\boldsymbol{x}} = (x_0, x_1, x_2, x_3, x_4)^T$ and $\hat{M}_P$ can be derived as follows.

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix} = \begin{bmatrix} \texttt{02} & \texttt{03} & \texttt{01} & \texttt{02} & \texttt{01} & \texttt{04} \\ \texttt{08} & \texttt{0e} & \texttt{07} & \texttt{09} & \texttt{06} & \texttt{11} \\ \texttt{22} & \texttt{3b} & \texttt{1f} & \texttt{25} & \texttt{18} & \texttt{42} \\ \texttt{84} & \texttt{e4} & \texttt{79} & \texttt{9b} & \texttt{67} & \texttt{0b} \\ \texttt{16} & \texttt{99} & \texttt{ef} & \texttt{6f} & \texttt{90} & \texttt{4b} \\ \texttt{96} & \texttt{cb} & \texttt{d2} & \texttt{79} & \texttt{24} & \texttt{a7} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_4 \end{bmatrix} = \begin{bmatrix} \texttt{02} & \texttt{03} & \texttt{01} & \texttt{02} & \texttt{01} \oplus \texttt{04} \\ \texttt{08} & \texttt{0e} & \texttt{07} & \texttt{09} & \texttt{06} \oplus \texttt{11} \\ \texttt{22} & \texttt{3b} & \texttt{1f} & \texttt{25} & \texttt{18} \oplus \texttt{42} \\ \texttt{84} & \texttt{e4} & \texttt{79} & \texttt{9b} & \texttt{67} \oplus \texttt{0b} \\ \texttt{16} & \texttt{99} & \texttt{ef} & \texttt{6f} & \texttt{90} \oplus \texttt{4b} \\ \texttt{96} & \texttt{cb} & \texttt{d2} & \texttt{79} & \texttt{24} \oplus \texttt{a7} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$$= \begin{bmatrix} \texttt{02} & \texttt{03} & \texttt{01} & \texttt{02} & \texttt{05} \\ \texttt{08} & \texttt{0e} & \texttt{07} & \texttt{09} & \texttt{17} \\ \texttt{22} & \texttt{3b} & \texttt{1f} & \texttt{25} & \texttt{5a} \\ \texttt{84} & \texttt{e4} & \texttt{79} & \texttt{9b} & \texttt{6c} \\ \texttt{16} & \texttt{99} & \texttt{ef} & \texttt{6f} & \texttt{db} \\ \texttt{96} & \texttt{cb} & \texttt{d2} & \texttt{79} & \texttt{83} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \triangleq \hat{M}_P \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

Consequently, we use the primitive matrix of $\hat{M}_P$ for the first column and the primitive matrix of M_P for other columns. Regarding the intermediate linear layers, we use the disjointed representation with discarding the invalid trails approach presented at Section 4.2. The result of solving the model is that a valid division trail that ends at a unit vector does not exist for any output bits,

i.e., each output bit after 4 rounds will have zero-sum. It should be mentioned that the model of the first linear layer using the disjointed representation and not discarding the invalid trails leads some bits to be imbalanced.

5-round Integral Distinguisher. Similar to the new 4-round one, we employed the same property of the matrix M_P to build the 5-round distinguisher. We firstly construct a set of 2^{280} chosen plaintexts where the last two elements of the diagonal are active and equal (denoted as $\bar{\mathcal{A}}$), and the other elements of the state matrix are active. This set can be divided, after the first round, into 2^{232} sub-sets such that every sub-set has 6 bytes active at specific positions as shown below. Therefore, each sub-set can be considered as an input to 4-round distinguisher that exploit the structure of the round function.

$$\begin{bmatrix} \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} \\ \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} \\ \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} \\ \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} \\ \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \bar{\mathcal{A}} & \mathcal{A} \\ \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \bar{\mathcal{A}} \end{bmatrix} \xrightarrow{SR \circ S \circ X} \begin{bmatrix} \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} \\ \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} \\ \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} \\ \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} \\ \bar{\mathcal{A}} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} \\ \bar{\mathcal{A}} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} \end{bmatrix} \xRightarrow{L} 2^{232} \times \left\{ \begin{bmatrix} \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{A} & \mathcal{C} \\ \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{A} \\ \mathcal{A} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} \\ \mathcal{C} & \mathcal{A} & \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} \\ \mathcal{C} & \mathcal{C} & \mathcal{A} & \mathcal{C} & \mathcal{C} & \mathcal{C} \\ \mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{A} & \mathcal{C} & \mathcal{C} \end{bmatrix} \xRightarrow{4R} \begin{bmatrix} \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} \end{bmatrix} \right\}$$

MILP for the 5-round distinguisher. We have followed the same steps as modelling the 4-round distinguisher to model the 5-round one, where we use the primitive matrix of $\hat{M}_P$ for the first column multiplication in the first round at which the model starts and the primitive matrix of M_P for the other columns. The result of solving the model indicates that each output bit after 5 rounds is balanced.

Other Experiments. We have employed our search approach (Section 4.2) to build a regular 5-round distinguisher that does not exploit the previous property of the matrix. We verified that this kind of distinguisher does not exist even when the number of active bits are 287 bits. Also, we have tried to reduce the number of active bits in both the regular and the new 4-round distinguisher by setting one of the active bits to constant and resolving the model. We verified that a distinguisher using less number of active bits does not exist.

6 Conclusions

In this paper, we proposed a new MILP model for the propagation of the BDP through non-bit-permutation linear layers. To the best of our knowledge, this model is the most efficient one for large linear layers. With the help of our model, we improved the previous 3- and 4-round integral distinguishers of Kuznyechik block cipher and the 4-round one of PHOTON's internal permutation (P_{288}). We also found, for the first time, two 4-round integral distinguishers for Kalyna block cipher and a 5-round integral distinguisher for PHOTON's internal permutation (P_{288}).

A Gaussian Elimination for the Toy Linear Layer

$$\left[\begin{array}{ccccc|c} 1&0&0&0&0&r_0\\ 1&1&0&0&1&r_1\\ 0&1&0&0&0&r_2\\ 0&0&1&0&0&r_3\\ 0&0&0&1&0&r_4\\ 0&1&0&0&1&r_5\\ 1&1&0&0&0&r_6\\ 0&0&1&1&0&r_7 \end{array}\right] \rightarrow \left[\begin{array}{ccccc|c} 1&0&0&0&0&r_0\\ 0&1&0&0&1&r_1+r_0\\ 0&1&0&0&0&r_2\\ 0&0&1&0&0&r_3\\ 0&0&0&1&0&r_4\\ 0&1&0&0&1&r_5\\ 0&1&0&0&0&r_6+r_0\\ 0&0&1&1&0&r_7 \end{array}\right] \rightarrow \left[\begin{array}{ccccc|c} 1&0&0&0&0&r_0\\ 0&1&0&0&1&r_1+r_0\\ 0&0&0&0&1&r_2+r_1+r_0\\ 0&0&1&0&0&r_3\\ 0&0&0&1&0&r_4\\ 0&0&0&0&0&r_5+r_1+r_0\\ 0&0&0&0&1&r_6+r_1\\ 0&0&1&1&0&r_7 \end{array}\right] \rightarrow \left[\begin{array}{ccccc|c} 1&0&0&0&0&r_0\\ 0&1&0&0&1&r_1+r_0\\ 0&0&1&0&0&r_3\\ 0&0&0&0&1&r_2+r_1+r_0\\ 0&0&0&1&0&r_4\\ 0&0&0&0&0&r_5+r_1+r_0\\ 0&0&0&0&1&r_6+r_1\\ 0&0&1&1&0&r_7 \end{array}\right] \rightarrow$$

$$\left[\begin{array}{ccccc|c} 1&0&0&0&0&r_0\\ 0&1&0&0&1&r_1+r_0\\ 0&0&1&0&0&r_3\\ 0&0&0&0&1&r_2+r_1+r_0\\ 0&0&0&1&0&r_4\\ 0&0&0&0&0&r_5+r_1+r_0\\ 0&0&0&0&1&r_6+r_1\\ 0&0&0&1&0&r_7+r_3 \end{array}\right] \rightarrow \left[\begin{array}{ccccc|c} 1&0&0&0&0&r_0\\ 0&1&0&0&1&r_1+r_0\\ 0&0&1&0&0&r_3\\ 0&0&0&1&0&r_4\\ 0&0&0&0&1&r_2+r_1+r_0\\ 0&0&0&0&0&r_5+r_1+r_0\\ 0&0&0&0&1&r_6+r_1\\ 0&0&0&1&0&r_7+r_3 \end{array}\right] \rightarrow \left[\begin{array}{ccccc|c} 1&0&0&0&0&r_0\\ 0&1&0&0&1&r_1+r_0\\ 0&0&1&0&0&r_3\\ 0&0&0&1&0&r_4\\ 0&0&0&0&1&r_2+r_1+r_0\\ 0&0&0&0&0&r_5+r_1+r_0\\ 0&0&0&0&1&r_6+r_1\\ 0&0&0&0&0&r_7+r_3+r_4 \end{array}\right] \rightarrow$$

$$\left[\begin{array}{ccccc|c} 1&0&0&0&0&r_0\\ 0&1&0&0&1&r_1+r_0\\ 0&0&1&0&0&r_3\\ 0&0&0&1&0&r_4\\ 0&0&0&0&1&r_2+r_1+r_0\\ 0&0&0&0&0&r_5+r_1+r_0\\ 0&0&0&0&0&r_6+r_2+r_0\\ 0&0&0&0&0&r_7+r_3+r_4 \end{array}\right] \rightarrow \left\{\begin{array}{l} r_0+r_1+r_5=\mathbf{0}\\ r_0+r_2+r_6=\mathbf{0}\\ r_3+r_4+r_7=\mathbf{0} \end{array}\right\}$$

References

1. Banik, S., Pandey, S.K., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y.: GIFT: A Small Present. In: Fischer, W., Homma, N. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2017. LNCS, vol. 10529, pp. 321–345. Springer International Publishing, Cham (2017)
2. Biryukov, A., Khovratovich, D., Perrin, L.: Multiset-Algebraic Cryptanalysis of Reduced Kuznyechik, Khazad, and secret SPNs. IACR Transactions on Symmetric Cryptology **2016**(2), 226–247 (Feb 2017). https://doi.org/10.13154/tosc.v2016.i2.226-247, https://tosc.iacr.org/index.php/ToSC/article/view/572
3. 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: Cryptographic Hardware and Embedded Systems - CHES 2007. LNCS, vol. 4727, pp. 450–466. Berlin, Heidelberg (2007)
4. Dolmatov, V.: GOST R 34.12-2015: Block Cipher "Kuznyechik". RFC 7801, RFC Editor (3 2016), https://tools.ietf.org/html/rfc7801

5. ElSheikh, M., Tolba, M., Youssef, A.M.: Integral Attacks on Round-Reduced Bel-T-256. In: Cid, C., Jacobson Jr., M.J. (eds.) Selected Areas in Cryptography – SAC 2018. LNCS, vol. 11349, pp. 73–91. Springer International Publishing, Cham (2019)
6. ElSheikh, M., Youssef, A.M.: A cautionary note on the use of Gurobi for cryptanalysis. Cryptology ePrint Archive, Report 2020/1112 (2020), https://eprint.iacr.org/2020/1112
7. ElSheikh, M., Youssef, A.M.: Integral Cryptanalysis of Reduced-Round Tweakable TWINE. In: Krenn, S., Shulman, H., Vaudenay, S. (eds.) Cryptology and Network Security CANS 2020. LNCS, vol. 12579, pp. 485–504. Springer International Publishing, Cham (2020)
8. Guo, J., Peyrin, T., Poschmann, A.: The PHOTON Family of Lightweight Hash Functions. In: Rogaway, P. (ed.) Advances in Cryptology – CRYPTO 2011. LNCS, vol. 6841, pp. 222–239. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
9. Gurobi Optimization, L.: Gurobi Optimizer Reference Manual (2020), http://www.gurobi.com
10. Hu, K., Wang, Q., Wang, M.: Finding Bit-Based Division Property for Ciphers with Complex Linear Layers. IACR Transactions on Symmetric Cryptology **2020**(1), 396–424 (May 2020). https://doi.org/10.13154/tosc.v2020.i1.396-424, https://tosc.iacr.org/index.php/ToSC/article/view/8570
11. IBM: IBM ILOG CPLEX 12.10 User's Manual (2020), https://www.ibm.com/support/knowledgecenter/SSSA5P_12.10.0/COS_KC_home.html
12. Knudsen, L., Wagner, D.: Integral Cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 112–127. Springer (2002)
13. Oliynykov, R., Gorbenko, I., Kazymyrov, O., Ruzhentsev, V., Kuznetsov, O., Gorbenko, Y., Dyrda, O., Dolgov, V., Pushkaryov, A., Mordvinov, R., Kaidalov, D.: A New Encryption Standard of Ukraine: The Kalyna Block Cipher. Cryptology ePrint Archive, Report 2015/650 (2015), https://eprint.iacr.org/2015/650
14. Shishkin, V., Dygin, D., Lavrikov, I., Marshalko, G., Rudskoy, V., Trifonov, D.: Low-weight and hi-end: Draft Russian encryption standard. In: 3rd Workshop on Current Trends in Cryptology - CTCrypt 2014. pp. 183–188 (2014)
15. Sun, B., Liu, M., Guo, J., Qu, L., Rijmen, V.: New Insights on AES-Like SPN Ciphers. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology – CRYPTO 2016. LNCS, vol. 9814, pp. 605–624. Springer Berlin Heidelberg, Berlin, Heidelberg (2016)
16. Sun, L., Wang, W., Wang, M.Q.: MILP-aided bit-based division property for primitives with non-bit-permutation linear layers. IET Information Security **14**, 12–20(8) (January 2020), https://digital-library.theiet.org/content/journals/10.1049/iet-ifs.2018.5283
17. Todo, Y.: Structural Evaluation by Generalized Integral Property. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 287–314. Springer (2015)
18. Todo, Y.: Integral Cryptanalysis on Full MISTY1. Journal of Cryptology **30**(3), 920–959 (2017)
19. Todo, Y., Morii, M.: Bit-based division property and application to simon family. In: Peyrin, T. (ed.) FSE 2016. LNCS, vol. 9783, pp. 357–377. Springer (2016)
20. 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: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10031, pp. 648–678. Springer (2016)

21. Zhang, W., Rijmen, V.: Division cryptanalysis of block ciphers with a binary diffusion layer. IET Information Security **13**, 87–95(8) (March 2019), https://digital-library.theiet.org/content/journals/10.1049/iet-ifs.2018.5151