IET Information Security

Research Article

IET Journals
The Institution of Engineering and Technology

MILP-aided bit-based division property for primitives with non-bit-permutation linear layers

ISSN 1751-8709
Received on 8th June 2018
Revised 25th April 2019
Accepted on 19th August 2019
E-First on 24th October 2019
doi: 10.1049/iet-ifs.2018.5283
www.ietdl.org

Ling Sun[1,2], Wei Wang[1,3,4], Meiqin Q. Wang[1,3] ✉

[1]*Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan 250100, People's Republic of China*

[2]*School of Computer Science and Technology, Shandong University, Qingdao 266237, People's Republic of China*

[3]*School of Cyber Science and Technology, Shandong University, Qingdao 266237, People's Republic of China*

[4]*Qilu University of Technology (Shandong Academy of Sciences), Jinan 250353, People's Republic of China*

✉ *E-mail: mqwang@sdu.edu.cn*

Abstract: In this study, the authors settle the feasibility of mixed integer linear programming (MILP)-aided bit-based division property for ciphers with non-bit-permutation linear layers. First, they transform the complicated linear layers to their primitive representations. Then, the original Copy and exclusive OR models are generalised, and these models are exploited to depict the primitive representations. Accord- ingly, the MILP-aided bit-based division property can be applied to much more primitives with complicated linear layers. As an illus- tration, they rst evaluate the bit-based division properties of some word-oriented block ciphers. For Midori64, they obtain a 7-round integral distinguisher, which achieves one more round than the previous results. At the same time, the data requirements of some existing distinguishers are also reduced. They decrease the data complexities of 4-round and 5-round distinguishers for LED and Joltik-BC by half. Then, the bit-based division properties of some bit-oriented ciphers such as Serpent and Noekeon are considered. The data complexities of their distinguishers for short rounds are reduced. Besides, they evaluate the bit-based division properties of the internal permutations in some hash functions. An 18-round zero-sum distinguisher for SPONGENT-88 is proposed, which achieves four more rounds than the previous ones. Some integral distinguishers for PHOTON permutations are improved.

1 Introduction

The integral cryptanalysis was first introduced as a dedicate attack for the word-oriented block cipher SQUARE by Daemen *et al.* [1] at Fast software encryption (FSE) 1997. Theoretically, integral attacks can be applied to bit-oriented block ciphers. However, till FSE 2008 [2], Z'aba *et al.* first gave a specific tool to find integral distinguishers for bit-oriented block ciphers and the bit-pattern-based integral attack was successfully demonstrated on reduced-round variants of the block ciphers Noekeon [3], PRESENT [4], and Serpent [5].

At EUROCRYPT 2015, Todo [6] generalised the integral property to division property, which can precisely depict the inherent properties between traditional ALL and BALANCE properties. By propagating division property, the integral distinguishers can be constructed, efficiently. At CRYPTO 2015, Todo [7] showed that division property could be more useful if the S-box was supposed to be a public function. He detected a 6-round integral distinguisher for MISTY1 [8] by utilising the vulnerable property of S_7 and achieved the first attack against full MISTY1. At FSE 2016, Todo and Morii [9] proposed the bit-based division property and explored the 14-round integral distinguisher for SIMON32 [10]. They pointed out that the complexity of using bit-based division property was roughly 2^n for an n-bit block cipher. On the one hand, the considerable complexity restricted its application. On the other hand, whether the bit-based division property could be employed to analyse other bit-oriented block ciphers was unknown.

Many kinds of research focusing on these interesting issues have occurred in succession. At CRYPTO 2016, by introducing the notion of parity sets, Boura and Canteaut [11] presented a new approach to deal with division property. For PRESENT, they provided some low-data integral distinguishers. By replacing the Substitution rule, which managed the propagation of S-box, with a more subtle propagation table, Sun and Wang [12] worked out the table-aided bit-based division property and successfully applied it to some bit-oriented primitives such as RECTANGLE [13] and SPONGENT-88 [14]. Thus, the bit-based division property can be compatible with ciphers other than SIMON. At ASIACRYPT 2016, Xiang *et al.* [15] applied mixed integer linear programming (MILP) method to search integral distinguisher based on division property, which is called MILP-aided bit-based division property in this paper, and found some longer integral distinguishers for SIMON family, Simeck family [16], PRESENT, RECTANGLE, LBlock [17], and TWINE [18]. Their work handled the problem about the complexity and showed that bit-based division property could be efficiently applied to some ciphers whose block sizes are more significant than 32. However, the linear layers for all these analytical ciphers are restricted to only simple bit-permutations. Thus, the feasibility of MILP method to analyse ciphers with linear layers besides bit-permutations was left as a future work [15].

Our contributions: In this paper, we settle the open problem and improve some integral distinguishers for various primitives by MILP-aided bit-based division property. The contributions of this paper are summarised as follows:

i. *Construct new searching models for complicated linear layers not limited to bit-permutations*: First, we transform the complicated linear layers to the primitive representations. Then, the original Copy and exclusive OR (XOR) models are, respectively, generalised to deal with more output branches and input elements, and these generalised models are exploited to depict the primitive representations. In this way, we can model all kinds of linear layers only if we have their primitive representations, and we will find that getting the primitive representation of a linear layer is an easy task. Thus, the MILP-aided bit-based division property can be applied to much more primitives with relatively complicated linear layers.
ii. *Apply MILP-aided bit-based division property to some word-oriented ciphers including Midori64* [19], *light-emitting diode* [20], *Joltik-BC* [21], *and* Advanced Encryption Standard (AES) [22]: For Midori64, we obtain a 7-round integral

© The Institution of Engineering and Technology 2019

distinguisher, which gains one more round than the previous analysis. Moreover, the data complexity is reduced significantly for r-round distinguisher when $r \leqslant 6$. As to LED and Joltik-BC, the data requirements for 4-round and 5-round distinguishers are decreased by half. As to AES, our searching experiments show that integral distinguishers, which are based on the bit-based division property, covering more than four rounds probably do not exist.

iii. *Consider the bit-based division properties of some bit-oriented block ciphers such as Serpent* [5] *and Noekeon* [3]: Owing to their relatively complicated linear layers and large block sizes, it is difficult to perform integral cryptanalysis. At FSE 2008, Z'aba *et al.* [2] proposed 3.5-round integral distinguishers for Noekeon and Serpent. Todo [6] improved it with traditional division property. Applying the new method, we also reduce the data complexities of some short-round distinguishers for these two ciphers.

iv. *Evaluate the bit-based division properties of the internal permutations involved in some hash functions, e.g. SPONGENT* [14] *and PHOTON* [23]: The published results all focused on SPONGENT-88, that is, the 14-round zero-sum distinguishers proposed by Dong *et al.* [24], Fan *et al.* [25], and Sun and Wang [12]. The best one we obtained is an 18-round zero-sum distinguisher with complexity 2^{87}, which gains four more rounds than the previous ones. Moreover, we provide 20-round and 21-round zero-sum distinguishers for SPONGENT-128 and SPONGENT-160, respectively. For PHOTON permutations with 4 b cell, the data complexities for the 4-round distinguishers are reduced by half. Besides, we obtain a 9-round distinguisher for P_{256}, which gains one more round than the previous ones. Furthermore, for P_{288} using 8 b S-boxes, the data complexities of the distinguishers are dramatically improved.

The comparisons of the main results with previous results for some block ciphers and internal permutations of hash functions are shown in Tables 1 and 2, respectively.

Outline: The rest of this paper is organised as follows. In Section 2, we briefly review some notations and definitions about MILP-aided bit-based division property. Section 3 illustrates how to apply MILP-aided bit-based division property to ciphers with more complicated linear layers. Section 4 gives some applications of MILP-aided bit-based division property. We conclude this paper in Section 5.

2 Preliminary

2.1 Notations

In this section, we present the notations used throughout this paper.

For an n-bit string $a \in \mathbb{F}_2^n$, the ith element is expressed as $a[i]$, where the bit positions are labelled in big-endian, and the Hamming weight $wt(a)$ is calculated by $wt(a) = \sum_{i=0}^{n-1} a[i]$.

For any set $\mathbb{K}$, $|\mathbb{K}|$ denotes the number of elements in $\mathbb{K}$. Let $\varnothing$ be an empty set.

For any $\boldsymbol{a} = (a_0, a_1, \ldots, a_{m-1}) \in \mathbb{F}_2^{\ell_0} \times \mathbb{F}_2^{\ell_1} \times \cdots \times \mathbb{F}_2^{\ell_{m-1}}$, the vectorial Hamming weight of $\boldsymbol{a}$ is defined as $Wt(\boldsymbol{a}) = (wt(a_0), wt(a_1), \ldots, wt(a_{m-1})) \in \mathbb{Z}^m$. For any $\boldsymbol{k} \in \mathbb{Z}^m$ and $\boldsymbol{k}' \in \mathbb{Z}^m$, we define $\boldsymbol{k} \succeq \boldsymbol{k}'$ if $k_i \geq k_i'$ for all i. Otherwise, $\boldsymbol{k} \not\succeq \boldsymbol{k}'$.

Definition 1: : (bit product function [6]*)*: Assume $u \in \mathbb{F}_2^n$ and $x \in \mathbb{F}_2^n$. The bit product function π_u is defined as

$$\pi_u(x) = \prod_{i=0}^{n-1} x[i]^{u[i]}.$$

Table 1 Comparison of our main results for some block ciphers with previous results

Cipher	$\log_2(\#\{\text{texts}\})$				Reference
	$r = 4$	$r = 5$	$r = 6$	$r = 7$	
Midori64	4	12	45	61	Section 4.1
	28	52	60	—	[6][b]
Serpent[a]	23	83	113	124	Section 4.2
	28	84	113	124	[6]
Noekeon	27	83	113	124	Section 4.2
	28	84	113	124	[6]

$\log_2(\#\{\text{texts}\})$: the exponent of the number of required chosen plaintexts.

[a]Since Serpent uses different S-boxes, which have distinct properties, in different rounds, the starting round may influence the resulting distinguisher. Here, we refer to the case where the initial round is the first round.

[b]Corresponding distinguishers are derived with the method introduced in the literature.

Table 2 Comparison of our main results for some internal permutations of hash functions with previous results

Cipher	Round	$\log_2(\#\{\text{texts}\})$	Reference
SPONGENT-88	18	87	Section 4.3
	17	85	—
	16	84	—
	15	80	—
	14	84	[24]
	14	80	[12, 25]
SPONGENT-128	20	126	Section 4.3
SPONGENT-160	21	159	Section 4.3
P_{288} in PHOTON	3	8	Section 4.3
	3	253	[26][a]
	4	48	Section 4.3
	4	283	[26][a]

[a]Corresponding distinguishers are derived with the method introduced in the literature.

© The Institution of Engineering and Technology 2019

For $\boldsymbol{u} = (u_0, u_1, \ldots, u_{m-1}) \in \mathbb{F}_2^{\ell_0} \times \mathbb{F}_2^{\ell_1} \times \cdots \times \mathbb{F}_2^{\ell_{m-1}}$, let $\boldsymbol{x} = (x_0, x_1, \ldots, x_{m-1}) \in \mathbb{F}_2^{\ell_0} \times \mathbb{F}_2^{\ell_1} \times \cdots \times \mathbb{F}_2^{\ell_{m-1}}$ be the input, the bit product function π_u is defined as

$$\pi_u(\boldsymbol{x}) = \prod_{i=0}^{m-1} \pi_{u_i}(x_i).$$

2.2 Division property and bit-based division property

Traditional integral distinguisher is usually constructed by evaluating the propagation of integral properties such as ALL and BALANCE properties. Division property, which was first proposed in [6], is a generalisation of integral property. It can precisely depict the inherent properties between ALL and BALANCE properties, which makes division property an efficient tool to construct integral distinguisher. Bit-based division property [9] handles a particular case of division property, where the space under consideration is restricted to the direct product of a series of binary fields. Comparing to traditional division property, bit-based division property traced the division property at the bit level and showed its power by finding longer integral distinguisher for SIMON32. In this section, we will briefly review division property and bit-based division property and list some propagation rules of bit-based division property.

Definition 2: (*division property* [6]): Let $\mathbb{X}$ be a multi-set whose elements take values from $\mathbb{F}_2^{\ell_0} \times \mathbb{F}_2^{\ell_1} \times \cdots \times \mathbb{F}_2^{\ell_{m-1}}$. When the multi-set $\mathbb{X}$ has the division property $\mathcal{D}_{\mathbb{K}}^{\ell_0, \ell_1, \ldots, \ell_{m-1}}$, where $\mathbb{K}$ denotes a set of m-dimensional vectors whose ith element takes a value between 0 and ℓ_i; it fulfils the following conditions:

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

Remark 1: Note that $\ell_0, \ell_1, \ldots, \ell_{m-1}$ are restricted to 1 when we consider bit-based division property.

2.2.1 Propagation rules of bit-based division property: Todo [6] proved some propagation rules for general division property and these rules were summarised into five rules in [7], which were Substitution, Copy, XOR, Split, and Concatenation, respectively. Among the five rules, only Copy and XOR are necessary for bit-based division property. They are restated in a bit-based look as follows:

Rule 1 (Copy [7]*)*: Let F be a Copy function, where the input x *takes a value of* $\mathbb{F}_2$ *and the output is calculated as* $(y_0, y_1) = (x, x)$. Let $\mathbb{X}$ and $\mathbb{Y}$ be the input multi-set and output multi-set, respectively. Assuming that $\mathbb{X}$ has the division property $\mathcal{D}_{\{1\}}^{1}$, the division property of $\mathbb{Y}$ is $\mathcal{D}_{\mathbb{K}'}^{1\times 1}$. *There are only two possible cases for the propagation*

$$\begin{cases} \mathbb{K}' = \{(0, 0)\}, & \text{if } k = 0 \\ \mathbb{K}' = \{(0, 1), (1, 0)\}, & \text{if } k = 1 \end{cases}.$$

Rule 2 (XOR [7]*)***:** Let F be a function composed of XOR operation, where the input (x_0, x_1) *takes a value of* $\mathbb{F}_2 \times \mathbb{F}_2$ *and the output is calculated as* $y = x_0 \oplus x_1$. Let $\mathbb{X}$ and $\mathbb{Y}$ be the input multi-set and output multi-set, respectively. Assuming that $\mathbb{X}$ has division property $\mathcal{D}_{\{1\}}^{1\times 1}$, the division property of $\mathbb{Y}$ is $\mathcal{D}_{\mathbb{K}'}^{1}$. *There are only three possible cases for the propagation*

$$\begin{cases} \mathbb{K}' = \{(0)\}, & \text{if } \boldsymbol{k} = (0, 0) \\ \mathbb{K}' = \{(1)\}, & \text{if } \boldsymbol{k} = (0, 1) \text{ or } (1, 0). \\ \mathbb{K}' = \varnothing, & \text{if } \boldsymbol{k} = (1, 1) \end{cases}$$

For some bit-oriented block ciphers such as SIMON, AND is another non-linear operation. The propagation for AND is given in [9], and we summarise it as follows.

Rule 3 (AND [9]*)***:** Let F be a function composed of AND operation, where the input (x_0, x_1) *takes a value of* $\mathbb{F}_2 \times \mathbb{F}_2$ *and the output is calculated as* $y = x_0 \wedge x_1$. Let $\mathbb{X}$ and $\mathbb{Y}$ be the input multi-set and output multi-set, respectively. Assuming that $\mathbb{X}$ has division property $\mathcal{D}_{\{1\}}^{1\times 1}$, the division property of $\mathbb{Y}$ is $\mathcal{D}_{\mathbb{K}'}^{1}$. *There are only two possible cases for the propagation*

$$\begin{cases} \mathbb{K}' = \{(0)\}, & \text{if } \boldsymbol{k} = (0, 0) \\ \mathbb{K}' = \{(1)\}, & \text{otherwise} \end{cases}.$$

Propagating the bit-based division property of S-box: There are many different methods [11, 12, 15] to propagate bit-based division property through S-box. By analysing the algebraic normal form, the propagation can be summed up as a propagation table. The table has two columns. The first column is filled with input division property $\boldsymbol{k}$ while the second column is filled with output division property $\mathbb{K}$ corresponding to $\boldsymbol{k}$. Then, the propagation of bit-based division property for S-box becomes a simple look-up table.

2.3 MILP-aided bit-based division property

Although bit-based division property is proved to be a powerful tool to find integral distinguishers, the complexity of utilising this method is roughly 2^n for an n-bit block cipher. Owing to this restriction, searching integral distinguishers for some primitives whose sizes are more significant than 32 bits is almost impossible. At ASIACRYPT 2016, Xiang *et al.* [15] proposed the method of describing the bit-based division property with the MILP model. With the help of some openly available MILP optimisers such as Gurobi [http://www.gurobi.com/.], the complexities of employing bit-based division property can dramatically decrease, and the workload of designers and cryptanalysts is significantly reduced. In this section, we will give a brief review of MILP-aided bit-based division property.

The main idea of MILP-aided bit-based division property is modelling the propagation rules of bit-based division property with a series of linear inequalities [We do not distinguish linear equality and linear inequality in this paper since MILP model can include linear inequality as well as linear equality.].

2.3.1 Modelling Copy, AND, XOR, and S-box: Corresponding to Rule 1–Rule 3, the following models are proposed to describe three basic bit-wise operations with linear inequalities.

Model 1: : (Copy [15]*)*: Denote $(a) \xrightarrow{\text{Copy}} (b_0, b_1)$ as a division trail of Copy function, and the following inequalities are sufficient to describe the division propagation of Copy:

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

Model 2: : (AND [15]*)*: Denote $(a_0, a_1) \xrightarrow{\text{AND}} (b)$ as a division trail of AND function, and the following linear inequalities are sufficient to describe the division propagation of AND:

$$\begin{cases} b - a_0 \geqslant 0 \\ b - a_1 \geqslant 0 \\ b - a_0 - a_1 \leqslant 0 \\ a_0, a_1, b \text{ are binaries} \end{cases}.$$

Model 3: : (XOR [15]*)*: Denote $(a_0, a_1) \xrightarrow{\text{XOR}} (b)$ as a division trail through XOR function, and the following inequalities can describe the division trail through XOR function:

© The Institution of Engineering and Technology 2019

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

Modelling S-box: To deduce the linear inequality system of S-box, we first generate its propagation table. After that, by invoking the *inequality_generator()* function in the Sage [http://www.sagemath.org/.] software, a set of linear inequalities will be returned. Sometimes, the number of linear inequalities in the set is substantial such that adding all these inequalities into the MILP model will make the problem computationally infeasible. Thus, Sun *et al.* [27] proposed an algorithm called the greedy algorithm (Algorithm 1 in [15]) to reduce this set. Since the greedy algorithm is not deterministic, the resulting linear inequality systems are not unique for one S-box.

Up to now, for block ciphers based on the three operations and (or) S-box, we can construct a set of linear inequalities characterising one round division property propagation. Iterating this process r times, we can get a linear inequality system $\mathscr{L}$ describing r rounds division property propagation. All feasible solutions of $\mathscr{L}$ correspond to all r-round division trails, which are defined below.

Definition 3: : (division trail [15]): Denote $\{\boldsymbol{k}\} \triangleq \mathbb{K}_0 \to \mathbb{K}_1 \to \mathbb{K}_2 \to \cdots \to \mathbb{K}_r$ as a chain of division property propagation through r rounds of encryption. For any vector $\boldsymbol{k}_i^* \in \mathbb{K}_i$ ($i \geqslant 1$), 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 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.

2.3.2 Initial division property and stopping rule: Denote $(a_0^0, a_1^0, \ldots, a_{n-1}^0) \to \cdots \to (a_0^r, a_1^r, \ldots, a_{n-1}^r)$ as an r-round division trail $\mathscr{L}$ is a linear inequality system defined on variables a_i^j ($i = 0, 1, \ldots, n-1$, $j = 0, 1, \ldots, r$) and some auxiliary variables. Let $\mathscr{D}_{\{k\}}^{1^n}$ denote the initial input division property with $\boldsymbol{k} = (k_0, k_1, \ldots, k_{n-1})$, we need to add $a_i^0 = k_i$ ($i = 0, 1, \ldots, n-1$) into $\mathscr{L}$, and all feasible solutions of $\mathscr{L}$ are division trails which start from vector $\boldsymbol{k}$.

By applying the definition of division property, the existence of any vector with Hamming weight larger than two indicates that all bits of the state satisfy the zero-sum property. Moreover, the presence of a unit vector tells that the bit located at the position of the unique non-zero element does not follow the zero-sum property. Thus, the objective function is set as

$$\text{Obj}: \text{Min}\{a_0^r + a_1^r + \cdots + a_{n-1}^r\}.$$

Let $\mathscr{D}_{\mathbb{K}_i}^{1^n}$ denote the output division property after i rounds of encryption and the input division property is denoted by $\mathscr{D}_{\mathbb{K}_0}^{1^n}$. If $\mathbb{K}_{r+1}$ contains all the n unit vectors for the first time, the division property propagation should stop, and an r-round distinguisher can be derived from $\mathscr{D}_{\mathbb{K}_r}^{1^n}$.

Note that we only recall some key points here. For more details, please see [6, 7, 9, 11, 12, 15].

3 MILP-aided bit-based division property for primitives with non-bit-permutation linear layers

Even though MILP-aided bit-based division property illustrated by Xiang *et al.* handles the enormous complexities of bit-based division property, the primitives with non-bit-permutation linear layers are not considered. To settle this problem, the critical point is to transform the complex linear layer to an equivalent representation with only Copy and XOR operations and to generalise the original Copy and XOR models to handle it. The invocations of the generalised models introduce some intermediate variables, which are reorganised according to the equivalent representation, and the linear inequality system for the division property propagation of linear layer is obtained. Finally, MILP-aided bit-based division property becomes more powerful and can be applied to more primitives with relatively complicated linear layers.

3.1 Generalising Copy and XOR models

Note that we have many different ways to define a linear transformation. However, we always can represent the linear transformation as a matrix over $\mathbb{F}_2$. We call this kind of representation the primitive representation as in [28] and always denote $M_*^{\mathscr{PR}}$ the primitive representation of a linear transformation. A simple observation implies the following claim.

Claim 1: No matter how complicated the linear layer is, it can always be split into Copy and XOR operations according to the primitive representation.

To work for more complicated linear layers, we generalise Model 1 and Model 3 in Section 2.3 to fit for more output branches and input elements, respectively.

Model 4: : (generalised Copy): Denote $(a) \xrightarrow{\text{Copy}} (b_0, b_1, \ldots, b_{m-1})$ as a division trail of a Copy function, and the following inequalities are sufficient to describe the division propagation of Copy:

$$\begin{cases} a - b_0 - b_1 - \cdots - b_{m-1} = 0 \\ a, b_0, b_1, \ldots, b_{m-1} \text{ are binaries} \end{cases}.$$

Model 5: : (Generalised XOR): Denote $(a_0, a_1, \ldots, a_{m-1}) \xrightarrow{\text{XOR}} (b)$ as a division trail through XOR function, and the following inequalities can describe the division trail through XOR function:

$$\begin{cases} a_0 + a_1 + \cdots + a_{m-1} - b = 0 \\ a_0, a_1, \ldots, a_{m-1}, b \text{ are binaries} \end{cases}.$$

With Model 4 and Model 5, we can depict the division property propagation of any linear layer by introducing some intermediate variables according to the primitive representation of the linear layer.

3.2 Modelling the primitive representation of a linear layer

Let $\boldsymbol{M}^{\mathscr{PR}}$ be an $n \times n$ matrix, which is the primitive representation of a (or a part of) linear layer, and denote

$$\boldsymbol{M}^{\mathscr{PR}} = \begin{pmatrix} m_{0,0} & m_{0,1} & \cdots & m_{0,n-1} \\ m_{1,0} & m_{1,1} & \cdots & m_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n-1,0} & m_{n-1,1} & \cdots & m_{n-1,n-1} \end{pmatrix},$$

where $m_{i,j} \in \{0, 1\}$. We suppose that the Hamming weight of the ith column of $M^{\mathscr{PR}}$ is c_i, and the Hamming weight of the jth row is r_j. Let $c_M = \sum_{i=0}^{n-1} c_i = \sum_{j=0}^{n-1} r_j$ be the number of non-zero elements in $M^{\mathscr{PR}}$.

By analysing the primitive representation, the ith input bit occurs in c_i output branches and the ith input bit of $M^{\mathscr{PR}}$ needs to be copied c_i times. Thus, c_i intermediate variables need to be introduced to represent the division properties of these copies. To propagate the division properties for all input bits, c_M intermediate variables $t_0 - t_{c_{M-1}}$ are required in total.

Denote $(x_0, x_1, \ldots, x_{n-1}) \to (y_0, y_1, \ldots, y_{n-1})$ as a division trail of $M^{\mathscr{PR}}$. We may allocate the first c_0 intermediate variables to x_0, and allocate the next c_1 variables to x_1, and so forth. Then, by utilising Model 4, we list the linear inequalities to describe the Copy operations for all input bits as follows:

© The Institution of Engineering and Technology 2019

$$\begin{cases} x_0 - t_0 - t_1 - \cdots - t_{c_0-1} = 0 \\ x_1 - t_{c_0} - t_{c_0+1} - \cdots - t_{c_0+c_1-1} = 0 \\ \cdots\cdots \\ x_{n-1} - t_{c_M-c_{n-1}} - t_{c_M-c_{n-1}+1} - \cdots - t_{c_M-1} = 0 \\ x_0, x_1, \ldots, x_{n-1}, t_0, t_1, \ldots, t_{c_M-1} \text{ are binaries} \end{cases} \quad (1)$$

To propagate the XOR operations, those intermediate variables should be allocated according to the arrangement of non-zero elements in $M^{\mathscr{PR}}$. For example, since $t_0 - t_{c_0-1}$ are assigned to depict the division properties of the output copies for the first input bit, they are put in those positions of the first column's non-zero elements in order. Let $I^{(i)} = \left\{I_0^{(i)}, I_1^{(i)}, \ldots, I_{r_i-1}^{(i)}\right\}$ be the index set of the ith row, whose elements are the indexes of intermediate variables in the ith row. According to Model 5, the linear inequalities to describe the XOR operations for all output bits are obtained, that is

$$\begin{cases} t_{I_0^{(0)}} + t_{I_1^{(0)}} + \cdots + t_{I_{r_0-1}^{(0)}} - y_0 = 0 \\ t_{I_0^{(1)}} + t_{I_1^{(1)}} + \cdots + t_{I_{r_1-1}^{(1)}} - y_1 = 0 \\ \cdots\cdots \\ t_{I_0^{(n-1)}} + t_{I_1^{(n-1)}} + \cdots + t_{I_{r_{n-1}-1}^{(n-1)}} - y_{n-1} = 0 \\ y_0, y_1, \ldots, y_{n-1}, t_0, t_1, \ldots, t_{c_M-1} \text{ are binaries} \end{cases} . \quad (2)$$

Combining (1) and (2), we construct the linear inequality system used to trace the division property propagation of the linear operation $M^{\mathscr{PR}}$.

3.3 Application to the MixColumns of LED

To illustrate the above model, we take the MixColumns operation of LED [20] as an example. MixColumns is a part of linear operations for LED's round function, and it works like the MixColumn operation for AES [22]. It multiplies each column of the internal state by the same 4×4 maximum distance separable matrix $\boldsymbol{M}_{\text{LED}}$ over the field $\mathbb{F}_2^4$, where

$$\boldsymbol{M}_{\text{LED}} = \begin{pmatrix} 0\times4 & 0\times1 & 0\times2 & 0\times2 \\ 0\times8 & 0\times6 & 0\times5 & 0\times6 \\ 0\times b & 0\times e & 0\times a & 0\times9 \\ 0\times2 & 0\times2 & 0\times f & 0\times b \end{pmatrix}.$$

The underlying polynomial for the field multiplication is $x^4 + x + 1$. The primitive representation $M_{\text{LED}}^{\mathscr{PR}}$ of M_{LED} is given by

$$\begin{pmatrix} 0&0&1&0&1&0&0&0&0&1&0&0&0&1&0&0 \\ 1&0&0&1&0&1&0&0&0&0&1&0&0&0&1&0 \\ 1&1&0&0&0&0&1&0&1&0&0&1&1&0&0&1 \\ 0&1&0&0&0&0&0&1&1&0&0&0&1&0&0&0 \\ 1&0&0&1&0&1&1&0&1&0&1&0&0&1&1&0 \\ 1&1&0&0&1&0&1&1&1&1&0&1&1&0&1&1 \\ 0&1&1&0&0&1&0&1&1&1&1&0&0&1&0&1 \\ 0&0&1&0&1&1&0&0&0&1&0&1&1&1&0&0 \\ 0&1&0&1&1&1&1&1&1&1&0&1&0&0&0&1 \\ 1&0&1&0&0&1&1&1&1&1&1&0&1&0&0&0 \\ 1&1&0&1&0&0&1&1&1&1&1&1&0&1&0&0 \\ 1&0&1&1&1&1&1&0&1&0&1&0&0&0&1&1 \\ 0&1&0&0&0&1&0&0&0&1&1&1&0&1&0&1 \\ 0&0&1&0&0&0&1&0&0&0&1&1&1&0&1&0 \\ 1&0&0&1&1&0&0&1&0&0&0&1&1&1&0&1 \\ 1&0&0&0&1&0&0&0&1&1&1&1&1&0&1&1 \end{pmatrix}. \quad (3)$$

Note that there are four columns in the internal state of the LED, and M_{LED} operates on each column, independently. Thus, we list the linear inequalities for one column as follows.

Denote $(x_0, x_1, \ldots, x_{15}) \to (y_0, y_1, \ldots, y_{15})$ as a division trail of $M_{\text{LED}}^{\mathscr{PR}}$. There are 124 non-zero elements in $M_{\text{LED}}^{\mathscr{PR}}$. So that 124 intermediate variables $(t_0 - t_{123})$ are introduced, and they are arranged according to the positions of '1's in $M_{\text{LED}}^{\mathscr{PR}}$. Please see the equation below: (see (4)) . On the one hand, the variables located in the same column are precisely the variables used to describe the Copy operation for the corresponding input bit. Thus, the linear inequality system (5) is sufficient to describe the Copy operations. On the other hand, the variables located in the same row are involved in the XOR operation of the corresponding output bit. So the propagations of the XOR operations turn into linear inequality system (6). Thereby, we combine linear inequality systems (5) and (6) as a whole linear inequality system, which can trace the propagation of division property for $M_{\text{LED}}^{\mathscr{PR}}$.

3.4 Sketch of MILP-aided bit-based division property for primitives with complicated linear layers

In the remaining of this section, we give an overview of applying the MILP-aided bit-based division property to primitives with complicated linear layers. All the analyses of primitives provided in Section 4 follow the procedures given below:

Generating the linear inequality system for S-box:

(a) We deduce the propagation table of the S-box.
(b) All the elements in the propagation table are put into *inequality_generator()* to generate the linear inequalities used to describe the S-box.
(c) Greedy algorithm is invoked to simplify the above linear inequality system.

Generating the linear inequality system for linear layer:

(a) The linear layer is transformed into the primitive representation.
(b) The intermediate variables are introduced and arranged according to the non-zero elements in the primitive representation of the linear layer, and the linear inequality system is obtained.

Constructing the linear inequality system for $\boldsymbol{r}$ rounds division property propagation:

(a) The linear inequality system used to propagate r rounds division property is constructed by combining the above two linear inequality systems following the structure of the specific cipher.

Searching integral distinguishers with different initial division properties:

(a) To obtain various distinguishers, we can change the initial division property of the MILP model
(see (5))
(see (6))

4 Applications of MILP-aided bit-based division property

In this section, we show some applications of MILP-aided bit-based division property. First, we present the applications of MILP-aided bit-based division property to some word-oriented block ciphers such as Midori64, LED, Joltik-BC, and AES. Then, we evaluate some bit-oriented block ciphers including Serpent and Noekeon. At last, the bit-based division properties of the internal permutations used in some hash functions are concerned. We do not list the details for space limitation, and please see the full version of this paper [29] for more information.

4.1 Applications to word-oriented block ciphers

4.1.1 Application to Midori64: A brief introduction of Midori64 [19]*:* Midori64 is a block cipher with the 64 b block and the 128 b key. The 64 b state S is arranged in a 4×4 matrix of 4 b cells

© The Institution of Engineering and Technology 2019

$$\left(\begin{array}{cccc|cccc|cccc|cccc}
0 & 0 & t_{16} & 0 & t_{28} & 0 & 0 & 0 & 0 & t_{68} & 0 & 0 & 0 & t_{103} & 0 & 0 \\
t_0 & 0 & 0 & t_{22} & 0 & t_{35} & 0 & 0 & 0 & 0 & t_{77} & 0 & 0 & 0 & t_{110} & 0 \\
t_1 & t_9 & 0 & 0 & 0 & 0 & t_{43} & 0 & t_{58} & 0 & 0 & t_{86} & t_{95} & 0 & 0 & t_{116} \\
0 & t_{10} & 0 & 0 & 0 & 0 & 0 & t_{51} & t_{59} & 0 & 0 & 0 & t_{96} & 0 & 0 & 0 \\
\hline
t_2 & 0 & 0 & t_{23} & 0 & t_{36} & t_{44} & 0 & t_{60} & 0 & t_{78} & 0 & 0 & t_{104} & t_{111} & 0 \\
t_3 & t_{11} & 0 & 0 & t_{29} & 0 & t_{45} & t_{52} & t_{61} & t_{69} & 0 & t_{87} & t_{97} & 0 & t_{112} & t_{117} \\
0 & t_{12} & t_{17} & 0 & 0 & t_{37} & 0 & t_{53} & t_{62} & t_{70} & t_{79} & 0 & 0 & t_{105} & 0 & t_{118} \\
0 & 0 & t_{18} & 0 & t_{30} & t_{38} & 0 & 0 & 0 & t_{71} & 0 & t_{88} & t_{98} & t_{106} & 0 & 0 \\
\hline
0 & t_{13} & 0 & t_{24} & t_{31} & t_{39} & t_{46} & t_{54} & t_{63} & t_{72} & 0 & t_{89} & 0 & 0 & 0 & t_{119} \\
t_4 & 0 & t_{19} & 0 & 0 & t_{40} & t_{47} & t_{55} & t_{64} & t_{73} & t_{80} & 0 & t_{99} & 0 & 0 & 0 \\
t_5 & t_{14} & 0 & t_{25} & 0 & 0 & t_{48} & t_{56} & t_{65} & t_{74} & t_{81} & t_{90} & 0 & t_{107} & 0 & 0 \\
t_6 & 0 & t_{20} & t_{26} & t_{32} & t_{41} & t_{49} & 0 & t_{66} & 0 & t_{82} & 0 & 0 & 0 & t_{113} & t_{120} \\
\hline
0 & t_{15} & 0 & 0 & 0 & t_{42} & 0 & 0 & 0 & t_{75} & t_{83} & t_{91} & 0 & t_{108} & 0 & t_{121} \\
0 & 0 & t_{21} & 0 & 0 & 0 & t_{50} & 0 & 0 & 0 & t_{84} & t_{92} & t_{100} & 0 & t_{114} & 0 \\
t_7 & 0 & 0 & t_{27} & t_{33} & 0 & 0 & t_{57} & 0 & 0 & 0 & t_{93} & t_{101} & t_{109} & 0 & t_{122} \\
t_8 & 0 & 0 & 0 & t_{34} & 0 & 0 & 0 & t_{67} & t_{76} & t_{85} & t_{94} & t_{102} & 0 & t_{115} & t_{123}
\end{array}\right) \tag{4}$$

$$\begin{cases}
x_0 - t_0 - t_1 - t_2 - t_3 - t_4 - t_5 - t_6 - t_7 - t_8 = 0 \\
x_1 - t_9 - t_{10} - t_{11} - t_{12} - t_{13} - t_{14} - t_{15} = 0 \\
x_2 - t_{16} - t_{17} - t_{18} - t_{19} - t_{20} - t_{21} = 0 \\
x_3 - t_{22} - t_{23} - t_{24} - t_{25} - t_{26} - t_{27} = 0 \\
x_4 - t_{28} - t_{29} - t_{30} - t_{31} - t_{32} - t_{33} - t_{34} = 0 \\
x_5 - t_{35} - t_{36} - t_{37} - t_{38} - t_{39} - t_{40} - t_{41} - t_{42} = 0 \\
x_6 - t_{43} - t_{44} - t_{45} - t_{46} - t_{47} - t_{48} - t_{49} - t_{50} = 0 \\
x_7 - t_{51} - t_{52} - t_{53} - t_{54} - t_{55} - t_{56} - t_{57} = 0 \\
x_8 - t_{58} - t_{59} - t_{60} - t_{61} - t_{62} - t_{63} - t_{64} - t_{65} - t_{66} - t_{67} = 0 \\
x_9 - t_{68} - t_{69} - t_{70} - t_{71} - t_{72} - t_{73} - t_{74} - t_{75} - t_{76} = 0 \\
x_{10} - t_{77} - t_{78} - t_{79} - t_{80} - t_{81} - t_{82} - t_{83} - t_{84} - t_{85} = 0 \\
x_{11} - t_{86} - t_{87} - t_{88} - t_{89} - t_{90} - t_{91} - t_{92} - t_{93} - t_{94} = 0 \\
x_{12} - t_{95} - t_{96} - t_{97} - t_{98} - t_{99} - t_{100} - t_{101} - t_{102} = 0 \\
x_{13} - t_{103} - t_{104} - t_{105} - t_{106} - t_{107} - t_{108} - t_{109} = 0 \\
x_{14} - t_{110} - t_{111} - t_{112} - t_{113} - t_{114} - t_{115} = 0 \\
x_{15} - t_{116} - t_{117} - t_{118} - t_{119} - t_{120} - t_{121} - t_{122} - t_{123} = 0 \\
x_0,\ x_1,\ \ldots,\ x_{15},\ t_0,\ t_1,\ \ldots,\ t_{123} \text{ are binaries}
\end{cases} \tag{5}$$

$$\begin{cases}
t_{16} + t_{28} + t_{68} + t_{103} - y_0 = 0 \\
t_0 + t_{22} + t_{35} + t_{77} + t_{110} - y_1 = 0 \\
t_1 + t_9 + t_{43} + t_{58} + t_{86} + t_{95} + t_{116} - y_2 = 0 \\
t_{10} + t_{51} + t_{59} + t_{96} - y_3 = 0 \\
t_2 + t_{23} + t_{36} + t_{44} + t_{60} + t_{78} + t_{104} + t_{111} - y_4 = 0 \\
t_3 + t_{11} + t_{29} + t_{45} + t_{52} + t_{61} + t_{69} + t_{87} + t_{97} + t_{112} + t_{117} - y_5 = 0 \\
t_{12} + t_{17} + t_{37} + t_{53} + t_{62} + t_{70} + t_{79} + t_{105} + t_{118} - y_6 = 0 \\
t_{18} + t_{30} + t_{38} + t_{71} + t_{88} + t_{98} + t_{106} - y_7 = 0 \\
t_{13} + t_{24} + t_{31} + t_{39} + t_{46} + t_{54} + t_{63} + t_{72} + t_{89} + t_{119} - y_8 = 0 \\
t_4 + t_{19} + t_{40} + t_{47} + t_{55} + t_{64} + t_{73} + t_{80} + t_{99} - y_9 = 0 \\
t_5 + t_{14} + t_{25} + t_{48} + t_{56} + t_{65} + t_{74} + t_{81} + t_{90} + t_{107} - y_{10} = 0 \\
t_6 + t_{20} + t_{26} + t_{32} + t_{41} + t_{49} + t_{66} + t_{82} + t_{113} + t_{120} - y_{11} = 0 \\
t_{15} + t_{42} + t_{75} + t_{83} + t_{91} + t_{108} + t_{121} - y_{12} = 0 \\
t_{21} + t_{50} + t_{84} + t_{92} + t_{100} + t_{114} - y_{13} = 0 \\
t_7 + t_{27} + t_{33} + t_{57} + t_{93} + t_{101} + t_{109} + t_{122} - y_{14} = 0 \\
t_8 + t_{34} + t_{67} + t_{76} + t_{85} + t_{94} + t_{102} + t_{115} + t_{123} - y_{15} = 0 \\
y_0,\ y_1,\ \ldots,\ y_{15},\ t_0,\ t_1,\ \ldots,\ vt_{123} \text{ are binaries}
\end{cases} \tag{6}$$

© The Institution of Engineering and Technology 2019

$$\boldsymbol{S} = \begin{bmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{bmatrix}.$$

Our searching algorithm related to the data processing part MidoriCore$_{(16)}$, which is a 16-round substitution permutation-network, and each round takes the following four operations, and the final round omits ShuffleCell and MixColumn operations. Before the first round, there is a key whitening operation. For more details, please see [19].

• *SubCell*: A 4 b S-box is applied to each cell of the state *S*.

• *ShuffleCell*: Each cell of the state is permuted as follows:

$$(s_0, s_1, \ldots, s_{15}) \leftarrow (s_0, s_{10}, s_5, s_{15}, s_{14}, s_4, s_{11}, s_1, s_9, s_3, s_{12}, s_6, s_7, s_{13}, s_2, s_8)$$

• *MixColumn*: Multiplying each column by a 4×4 matrix $\boldsymbol{M}_{\text{Midori64}}$ over $\mathbb{F}_2^4$, where

$$\boldsymbol{M}_{\text{Midori64}} = \begin{pmatrix} 0\times0 & 0\times1 & 0\times1 & 0\times1 \\ 0\times1 & 0\times0 & 0\times1 & 0\times1 \\ 0\times1 & 0\times1 & 0\times0 & 0\times1 \\ 0\times1 & 0\times1 & 0\times1 & 0\times0 \end{pmatrix}.$$

• *KeyAdd*: The ith 64 b round key RK_i is XORed to the state *S*.

Since KeyAdd operation does not affect the propagation of division property, we do not consider it in our analysis. Thus, the key schedule is omitted. For more details, please see [19].

Applying MILP-aided bit-based division property to Midori64 [19]*:*

• *Generating the linear inequality system for S-box:* The propagation table for S_{Midori64} has 48 vectors, and 54 linear inequalities are returned by invoking *inequality_generator()*. After using the greedy algorithm, only five linear inequalities are left.

• *Generating the linear inequality system for MixColumns operation:* There are 48 non-zero elements in the primitive representation of $\boldsymbol{M}_{\text{Midori64}}$. Thus, $48 \times 4 = 192$ intermediate variables $t_0 - t_{191}$ are required for one round of encryption since there are four columns in the state.

Experimental results of Midori64: We put different initial division properties into the MILP model, and a 7-round integral distinguisher is obtained, which gains one more round than the previous cryptanalysis. Besides, the data complexity is reduced significantly for *r*-round distinguisher, where $r \leqslant 6$. Our results are following the expectation of the designers that the length of the integral distinguisher is bounded by 7. The concrete number of chosen plaintexts to construct *r*-round distinguisher are listed in Table 1.

4.1.2 Applications to some AES-like block ciphers: We also test some AES-like block ciphers, e.g. LED [20], Joltik-BC [21], and AES [22]. For LED and Joltik-BC, the data requirements for 4-round and 5-round distinguishers are reduced by half. As to AES, even the initial division property $\boldsymbol{k} = [0 \times ffffffff, 0 \times ffffffff, 0 \times ffffffff, 0, \text{of which } \times fffffffe]$ the Hamming weight is 127, is put into AES's MILP model. However, no bit satisfies zero-sum property after five-round encryption. Our experimental results indicate that integral distinguishers built on bit-based division property, covering more than four rounds are unlikely to exist. The comparison and the numbers of chosen plaintexts to construct *r*-round integral distinguishers for these ciphers are given in Table 3.

Comparing to the dedicated attack to search integral distinguishers for AES-like ciphers in [6], we propagate the S-box at the bit level and consider the concrete form of the linear layer. Thus, the bit-based division properties for specific ciphers should be better than or at least equal to the more general search in [6]. From Table 3, the data requirements of *r*-round distinguishers for LED and Joltik-BC are improved where $3 < r < 6$, which follows the former claim. When we directly searched for 6-round distinguishers, we observed that Gurobi gradually ran out of memory and no solution was returned.

To settle this issue, we note that Todo *et al.* [31] proposed the idea of adding dummy objective function to accelerate the solving time of MILP at CRYPTO 2017. The primary purpose is to modify the objective function as

$$\text{Obj}: \text{Max}\{a_0^{\text{r}} + a_1^{\text{r}} + \cdots + a_{n-1}^{\text{r}}\},$$

which is meaningless, and we only care whether the problem is feasible or not. We adopt this method in search of 6-round distinguishers. For the initial division property $\boldsymbol{k} = [0 \times ffff, 0 \times ff0f, 0 \times fff0, 0 \times 0fff]$ with Hamming weight 52, Gurobi still cannot return any results. However, the experimental result illustrates that there is no zero-sum bit when the initial division property only has 51 non-zero bits. Since the bit-based division property for a specific cipher is no worst than the general search, we conclude that the data requirements for 6-round distinguishers are 2^{52}.

4.2 Applications to bit-oriented block ciphers

4.2.1 Application to the Serpent: A brief introduction of Serpent [5]*:* Serpent is a block cipher which was one of the five finalists for AES. It is a 32-round substitution permutation network (SPN) structure operating on four 32 b words (X_0, X_1, X_2, and X_3), thus giving a block size of 128 b. The round function consists of alternating layers of key mixing, S-boxes, and linear transformation. The Serpent has eight S-boxes ($S_0 - S_7$), and the set of eight S-boxes is used four times. Each round function uses a single S-box 32 times in parallel. The first round uses S_0 and the second round uses S_1. After using S_7 in the eighth round, S_0 is used

Table 3 Comparison of data requirements of integral distinguishers for some AES-like block ciphers

Cipher	$\log_2(\#\{\text{texts}\})$				Reference
	$r = 3$	$r = 4$	$r = 5$	$r = 6$	
LED and Joltik-BC	4	11	31	52[a]	Section 4.1.2
	4	12	32	52	[6]
	12	28	52	60	[6]
	28	52	60	63	[26][b]
	4	16	—	—	[1, 30][b]
AES	8	32	—	—	Section 4.1.2
	56	120	—	—	[6]
	117	127	—	—	[26][b]
	8	32	—	—	[1, 30][b]

[a]Results are obtained under the dummy objective function.

[b]Corresponding distinguishers are derived with the method introduced in the literatures.

© The Institution of Engineering and Technology 2019

again in the ninth round. We do not expand on the bit-wise linear transformation of Serpent for space limitation. Please see [5] for more information.

Applying MILP-aided bit-based division property to Serpent:

• *Generating the linear inequality systems for eight S-boxes:* For space limitation, we do not give the propagation tables and the linear inequality systems for the eight S-boxes of Serpent.

• *Generating the linear inequality system for linear transformation:* We treat the linear layer as a 128×128 matrix, and there are 610 non-zero elements in the primitive representation of Serpent's linear layer. Thus, 610 intermediate variables $t_0 - t_{609}$ are needed for one round of encryption.

Experimental results for Serpent: Since different rounds use different S-boxes, the starting round may influence the length of the resulting integral distinguisher. After analysing all possible cases, we find that the data requirements are different for different initial rounds, and the experimental results are shown in Table 4. Comparing to the results given by Todo [6], we improve the data complexities of some distinguishers for shorter rounds ($r < 6$). For $r \geqslant 6$, the data requirements are the same as the previous results.

4.2.2 Application to Noekeon: The bit-based division property for Noekeon [3] is also considered. Note that Noekeon follows permutation substitution permutation structure. Since Step 2 of Algorithm 2 in Todo's work [6] deals with division property propagation of non-linear layer first, we conjecture that Todo transformed Noekeon into the SPN structure, so that we also do the transformations to compare with the results of [6]. The experimental results can be found in Table 1.

Table 4 Comparison of data requirements for Serpent with different initial rounds

Initial Round	$\log_2$(#{texts})				Reference
	$r = 4$	$r = 5$	$r = 6$	$r = 7$	
0	23	83	113	124	Section 4.2
1, 2, and 6	24	83	113	124	
3, 4, 5, and 7	24	84	113	124	
all	28	84	113	124	[6]

4.3 Applications to other primitives

4.3.1 Application to SPONGENT: A brief introduction of SPONGENT [14]*:* SPONGENT is a family of lightweight hash functions with different hash sizes and similar round functions. There are five variants of SPONGENT, and we only analyse SPONGENT-88, SPONGENT-128, and SPONGENT-160 with hash sizes 88, 128, and 160, respectively. SPONGENT uses SP-network and utilises a PRESENT-type permutation which iterates 45, 70, and 90 times for the above-mentioned three variants. The non-linear layer applies one 4 bit S-box (S_{SPONTENT}) in parallel. For more details about SPONGENT, please see [14].

Experimental results for SPONGENT: For SPONGENT-88, we find four zero-sum distinguishers, and the general information is listed in Table 2. One of them is a 15-round zero-sum distinguisher with data complexity 2^{80}. This newly obtained distinguisher achieves one more round than the one proposed by Fan and Duan [25] while keeps the same complexity. The best one is an 18-round zero-sum distinguisher with data complexity 2^{87}, which gains four more rounds than the previous ones. Besides, some experimental results for SPONGENT-128 and SPONGENT-160 are also provided in Table 2.

4.3.2 Applications to PHOTON permutations: We also analyse the internal permutation P_t of PHOTON [23], which is a family of hash functions, where $t \in \{100, 144, 196, 256, 288\}$. All P_ts adopt the AES-like structure, and the cell sizes of the first four variants are all 4 b while the last one has 8 b cell size. The experimental results for different variants can be found in Table 5.

Similarly to the cases as mentioned above for LED and Joltik-BC, for some initial division properties, the direct search is out of operation. To handle this problem, we introduce the dummy objective, and the results obtained under this method are marked

Table 5 Comparison of data requirements of integral distinguishers for PHOTON's internal permutations

Cipher	$\log_2$(#{texts})							Reference
	$r = 3$	$r = 4$	$r = 5$	$r = 6$	$r = 7$	$r = 8$	$r = 9$	
P_{100}	4	11	20	$\leqslant 72$	97[a]	—	—	Section 4.3
	4	12	20	72	97	—	—	[6]
	12	28	76	92	—	—	—	[6]
	28	76	92	98	—	—	—	[26][b]
	4	20	—	—	—	—	—	[1, 30][b]
P_{144}	4	11	24	$\leqslant 84$	131[a]	—	—	Section 4.3
	4	12	24	84	132	—	—	[6]
	12	28	84	124	140	—	—	[6]
	28	82	124	138	142	—	—	[26][b]
	4	24	—	—	—	—	—	[1, 30][b]
P_{196}	4	11	24[a]	84[a]	164[a]	192	—	Section 4.3
	4	12	24	84	164	192	—	[6]
	12	28	84	160	184	192	—	[6]
	28	82	158	184	192	195	—	[26][a]
	4	28	—	—	—	—	—	[1, 30][a]
P_{256}	4	11	28	92[a]	$\leqslant 204$	249[a]	252	Section 4.3
	4	12	28	92	204	249	—	[6]
	12	28	84	200	237	252	—	[6]
	28	82	198	237	250	254	—	[26][a]
	4	32	—	—	—	—	—	[1, 30][a]
P_{288}	8	48	—	—	—	—	—	Section 4.3
	253	283	—	—	—	—	—	[26][a]

[a]Results are obtained under dummy objective function.

[b]Corresponding distinguishers are derived with the method introduced in the literatures.

© The Institution of Engineering and Technology 2019

with '*' in Table 5. For the cases marked with '⩽' in Table 5, Gurobi cannot give any results even though we apply the dummy objective function. Thus, we only claim an upper-bound for the data requirement.

From the observation of Table 5, the advantages can be summarised into three points. First, we improve the data complexities of 4-round integral distinguishers for variants with 4 b cell size, and the data complexity of 7-round distinguisher for P_{144} is reduced by half. Second, we significantly reduce the data requirements of distinguishers for P_{288}, whose cell size is 8 b. Besides, we obtain a 9-round distinguisher for P_{256} whose initial division property has a sizeable Hamming weight and extend the length of the integral distinguisher for P_{256} by one round.

5 Conclusion

In this paper, we answer the open question proposed by Xiang *et al.* at ASIACRYPT 2016, and construct new models to illustrate that the MILP technique can be applied to primitives with non-bit-permutation linear layers. The critical point is to transform the complicated linear layers to the primitive representations and generalise the original Copy and XOR models to depict the primitive representations. Accordingly, the MILP-aided bit-based division property can be performed. As an illustration, we adopt MILP-aided bit-based division property to detect integral distinguishers for various primitives and improved some previous results.

6 Acknowledgments

The research leading to these results has received funding from the National Natural Science Foundation of China (Grant no. 61572293), the Science and Technology on Communication Security Laboratory of China (Grant no. 9140c110207150c11050), the Key Science Technology Project of Shandong Province (Grant no. 2015GGX101046), and the Chinese Major Program of National Cryptography Development Foundation (Grant no. MMJJ20170102). Wei Wang is partially supported by the Open Research Fund from the Shandong provincial Key Laboratory of Computer Network (Grant no. SDKLCN-2017-04).

7 References

[1] Daemen, J., Knudsen, L.R., Rijmen, V.: 'The block cipher square'. Fourth Int. workshop Fast Software Encryption, FSE '97, Haifa, Israel, 20–22 January 1997, pp. 149–165

[2] Z'aba, M.R., Raddum, H., Henricksen, M., *et al.*: 'Bit-pattern based integral attack'. 15th Int. Workshop Fast Software Encryption, FSE 2008, Lausanne, Switzerland, 10–13 February 2008, pp. 363–381

[3] Daemen, J., Peeters, M., Van Assche, G., *et al.*: 'Nessie proposal: Noekeon'. First Open NESSIE Workshop, 2000, pp. 213–230

[4] Bogdanov, A., Knudsen, L. R., Leander, G., *et al.*: 'PRESENT: an ultra-lightweight block cipher'. Ninth Int. Workshop Cryptographic Hardware and Embedded Systems – CHES 2007, Vienna, Austria, 10–13 September 2007, pp. 450–466

[5] Biham, E., Anderson, R.J., Knudsen, L.R.: 'Serpent: a new block cipher proposal'. Fast Software Encryption, Fifth Int. Workshop FSE '98, Paris, France, 23–25 March 1998, pp. 222–238

[6] Todo, Y.: 'Structural evaluation by generalized integral property'. Advances in Cryptology – EUROCRYPT 2015–34th Annual Int. Conf. Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, 26–30 April 2015, pp. 287–314

[7] Todo, Y.: 'Integral cryptanalysis on full MISTY1'. Proc. Part I Advances in Cryptology – CRYPTO 2015–35th Annual Cryptology Conf., Santa Barbara, CA, USA, 16–20 August 2015, vol. **2015**, pp. 413–432

[8] Matsui, M.: 'New block encryption algorithm MISTY'. Fourth Int. Workshop Fast Software Encryption FSE '97, Haifa, Israel, 20–22 January 1997, pp. 54–68

[9] Todo, Y., Morii, M.: 'Bit-based division property and application to SIMON family'. 23rd Int. Conf. Fast Software Encryption – FSE 2016, Bochum, Germany, 20–23 March 2016, pp. 357–377

[10] Beaulieu, R., Shors, D., Smith, J., *et al.*: 'The SIMON and SPECK lightweight block ciphers'. Proc. 52nd Annual Design Automation Conf., San Francisco, CA, USA, 7–11 June 2015, pp. 1–6

[11] Boura, C., Canteaut, A.: 'Another view of the division property'. Advances in Cryptology – CRYPTO 2016–36th Annual Int. Cryptology Conf., Santa Barbara, CA, USA, 14–18 August 2016, pp. 654–682

[12] Sun, L., Wang, M.: 'Towards a further understanding of bit-based division property', IACR Cryptology ePrint Archive, 2016o:392

[13] Zhang, W., Bao, Z., Lin, D., *et al.*: 'RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms', *Sci. China Inf. Sci.*, 2015, **58**, (12), pp. 1–15

[14] Bogdanov, A., Knezevic, M., Leander, G., *et al.*: 'SPONGENT: the design space of lightweight cryptographic hashing', *IEEE Trans. Comput.*, 2013, **62**, (10), pp. 2041–2053

[15] Xiang, Z., Zhang, W., Bao, Z., *et al.*: 'Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers'. Advances in Cryptology – ASIACRYPT 2016–22nd Int. Conf. Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, 4–8 December 2016, pp. 648–678

[16] Yang, G., Zhu, B., Suder, V., *et al.*: 'The Simeck family of lightweight block ciphers'. Cryptographic Hardware and Embedded Systems – CHES 2015–17th Int. Workshop, Saint-Malo, France, 13–16 September 2015, pp. 307–329

[17] Wu, W., Zhang, L.: 'LBlock: a lightweight block cipher'. Ninth Int. Conf. Applied Cryptography and Network Security – ACNS 2011, Nerja, Spain, 7–10 June 2011, pp. 327–344

[18] Suzaki, T., Minematsu, K., Morioka, S., *et al.*: 'TWINE: a lightweight block cipher for multiple platforms'. 19th Int. Conf. Selected Areas in Cryptography SAC 2012, Windsor, ON, Canada, 15–16 August 2012, pp. 339–354

[19] Banik, S., Bogdanov, A., Isobe, T., *et al.*: 'Midori A block cipher for low energy'. Advances in Cryptology – ASIACRYPT 2015–21st Int. Conf. Theory and Application of Cryptology and Information Security, Auckland, New Zealand, 29 November–3 December 2015, pp. 411–436

[20] Guo, J., Peyrin, T., Poschmann, A., *et al.*: 'The LED block cipher'. Cryptographic Hardware and Embedded Systems – CHES 2011–13th Int. Workshop, Nara, Japan, 28 September–1 October 2011, pp. 326–341

[21] Jean, J., Nikolić, I., Peyrin, T.: 'Joltik v1. 3', CAESAR Round, 2, 2015

[22] NIST FIPS Pub. 197.: 'Advanced encryption standard (AES)', *Fed. Inf. Process. Stand. Publ.*, 2001, **197**, (441), pp. 1–51

[23] Guo, J., Peyrin, T., Poschmann, A.: 'The PHOTON family of lightweight hash functions'. Advances in Cryptology – CRYPTO 2011–31st Annual Cryptology Conf., Santa Barbara, CA, USA, 14–18 August 2011, pp. 222–239

[24] Dong, L., Wu, W.-L., Wu, S., *et al.*: 'Another look at the integral attack by the higher-order differential attack', *Jisuanji Xuebao (Chin. J. Comput.)*, 2012, **35**, (9), pp. 1906–1917

[25] Fan, S., Duan, M.: 'Improved zero-sum distinguisher for SPONGENT-88'. Int. Conf. Electromechanical Control Technology and Transportation – ICECTT 2015 – Zhuhai City, Guangdong Province, China, 31 October–1 November 2015, pp. 582–587

[26] Boura, C., Canteaut, A., Cannière, C.D.: 'Higher-order differential properties of Keccak and *Luffa*'. Fast Software Encryption – 18th Int. Workshop FSE 2011, Lyngby, Denmark, 13–16 February 2011, pp. 252–269

[27] Sun, S., Hu, L., Wang, M., *et al.*: 'Towards finding the best characteristics of some bit-oriented block ciphers and automatic enumeration of (related-key) differential and linear characteristics with predefined properties', Technical Report Cryptology ePrint Archive, Report 2014/747, 2014

[28] Sun, B., Liu, Z., Rijmen, V., *et al.*: 'Links among impossible differential, integral and zero correlation linear cryptanalysis'. Advances in Cryptology – CRYPTO 2015–35th Annual Cryptology Conf., Santa Barbara, CA, USA, 16–20 August 2015, pp. 95–115

[29] Sun, L., Wang, W., Wang, M.: 'MILP-aided bit-based division property for primitives with non-bit-permutation linear layers', IACR Cryptology ePrint Archive, 2016o:811

[30] Knudsen, L.R., Wagner, D.: 'Integral cryptanalysis'. Ninth Int. Workshop Fast Software Encryption FSE 2002, Leuven, Belgium, 4–6 February 2002, vol. **2002**, pp. 112–127

[31] Todo, Y., Isobe, T., Hao, Y., *et al.*: 'Cube attacks on non-blackbox polynomials based on division property'. Advances in Cryptology – CRYPTO 2017–37th Annual Int. Cryptology Conf., Santa Barbara, CA, USA, Leuven, Belgium, 20–24 August 2017, pp. 250–279

© The Institution of Engineering and Technology 2019