MILP-Aided Bit-Based Division Property for Primitives with Non-Bit-Permutation Linear Layers

Ling Sun[1], Wei Wang[1], Meiqin Wang[1,2]

[1] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, 250100, China
[2] State Key Laboratory of Cryptology, P.O. Box 5159, Beijing, 100878, China
`lingsun@mail.sdu.edu.cn; weiwangsdu@sdu.edu.cn; mqwang@sdu.edu.cn`

Abstract. At ASIACRYPT 2016, Xiang *et al.* applied MILP method to search integral distinguisher based on division property. This method handled the huge time and memory complexities which had constituted the main restriction of the bit-based division property proposed by Todo and Morri, and showed its strength through finding some longer integral distinguishers for various primitives. Although MILP-aided bit-based division property has given many interesting results for some ciphers, the linear layers of these cipher are simple bit-permutations. Thus, the feasibility of MILP method applying to ciphers with linear layers which are not bit-permutations was left as a future work. In this paper, we handle this problem. Following this way, MILP-aided bit-based division property can operate on more primitives. As an illustration, we apply MILP-aided bit-based division property to find integral distinguishers for AES, LED, `Joltik-BC`, PHOTON, Serpent, Noekeon, SM4, and SPONGENT-88. We can not find any integral distinguisher whose length is longer than four rounds for AES. But for LED and `Joltik-BC`, which are AES-like block ciphers, we obtain 6-round integral distinguishers. For PHOTON permutations, which are also AES-like permutations, we obtain some better integral distinguishers comparing with those provided in its design paper. Based on these observations, the security of these AES-like block ciphers may need to be reconsidered and directly copying AES-like security proofs for some attacks seems to be less reasonable. We also find 7-round integral distinguishers for Serpent and Noekeon, which attain 3.5 more rounds than the previous distinguishers found by Z'aba *et al.* at FSE 2008. For SM4, we find a 12-round integral distinguisher, which attains four more rounds than the previous distinguisher found by Liu *et al.* at ACISP 2007. A 16-round higher-order integral distinguisher for SPONGENT-88 is proposed and this newly found distinguisher attains two more rounds than the previously known distinguishers.

Keywords: MILP-aided bit-based division property, AES, LED, PHOTON, `Joltik-BC`, Serpent, Noekeon, SM4, SPONGENT-88

1 Introduction

The integral cryptanalysis was first introduced as a dedicate attack for the block cipher SQUARE [8]. In principle, integral attacks can be applied to bit-oriented block ciphers. However, for a fairly long time, there do not exist specific tools to define the attack on these ciphers. At FSE 2008 [30], Z'aba *et al.* firstly gave an explicit 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 [9], PRESENT [5], and Serpent [1].

At EUROCRYPT 2015, Todo [25] generalized the integral property and firstly proposed division property. By using division property, we are allowed to effectively construct integral distinguishers even if block ciphers have non-bijective functions, bit-oriented structures, and low-degree functions. But in [25], the S-box was restricted to be a secret black box and only the algebraic degree of the S-box was supposed to be known. At CRYPTO 2015, Todo [24] showed that division property could be more useful if an S-box was supposed to be a public function. He found a 6-round integral distinguisher for MISTY1 [18] by utilizing the vulnerable property of S_7 and gave the first attack against full MISTY1.

At FSE 2016, Todo and Morri [26] proposed bit-based division property and applied it to find 14-round integral distinguisher for SIMON32 [3]. They pointed that the complexity requirements for the bit-based division property were roughly 2^n if an n-bit block cipher was under consideration. On the one hand, the huge complexities restricted the applications of bit-based division property. On the other hand, whether the bit-based division property could be used to analyse other bit-oriented block ciphers other than SIMON was not known.

Based on these unsolvable questions, many further researches have occurred in succession. At CRYPTO 2016, by introducing the notion of parity sets, Boura and Canteaut [6] gave a new approach to deal with division property. For the analysis of 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 [22] worked out table-aided bit-based division property, and successfully applied it to some bit-oriented primitives such as RECTANGLE [31] and SPONGENT-88 [4]. In this way, the problem about the usability of bit-based division property for ciphers other than SIMON was settled. At ASIACRYPT 2016, Xiang [28] *et al.* applied MILP method to search integral distinguisher based on division property[3], and found some long integral distinguishers for SIMON family of block ciphers, PRESENT, and RECTANGLE. For one thing, this work handled the problem about the complexities and bit-based division property could be efficiently applied to some ciphers whose block sizes are larger than 32. For another thing, they gave a different idea to trace the propagation through S-box.

Indeed, MILP-aided bit-based division property has given some interesting results for some ciphers, such as SIMON, SIMECK [29], PRESENT, RECTANGLE, LBlock [27], and TWINE [23]. But the linear layers for these ciphers are

[3] We name it *MILP-aided bit-based division property* in this paper.

simple bit-permutations. Thus, the feasibility of MILP method applying to ciphers with linear layers which are not bit-permutations was left as a future work in [28].

1.1 Our Contributions

The contributions of this paper are concluded as follows.

1. By introducing some intermediate variables among the linear layer, we settle the feasibility of MILP method applying to ciphers with more linear layers which are not bit-permutations. Following this way, MILP-aided bit-based division property can be applied to more primitives. We apply it to find integral distinguishers for AES [20], LED [15], `Joltik-BC` [16], PHOTON [14], Serpent [1], Noekeon [9], SM4 [11], and SPONGENT-88 [4].
2. The security evaluation of AES is always the key focus of many cryptographers because of its important and sensitive status. As we all know, AES has some 4-round integral distinguishers with data complexity 2^{32}. The researches to find new integral distinguishers of AES that cover more rounds have never stopped. But after applying MILP-aided bit-based division property, we find that there does not exist any integral distinguisher that covers more than four rounds even though we traverse the first 127 bits of the input multi-set. By no means do we judge that there is no integral distinguisher more than four rounds for AES. We only say that we can not find better integral distinguisher based on bit-based division property. `Joltik-BC` and LED are AES-like ciphers. Profiting from the structural similarity, cryptographers are able to drive simple yet interesting AES-like security proofs for `Joltik-BC` and LED regarding related- or single-key attacks. In this paper, we also take effort to search integral distinguishers for `Joltik-BC` and LED. Surprisingly, we find 6-round integral distinguishers for these two ciphers both with data complexity 2^{60}. For PHOTON permutations, which are also AES-like permutations, we find some better integral distinguishers comparing with those provided in [14]. Owing to the difference lying in the lengths of integral distinguishers for AES and these AES-like ciphers, the security of these AES-like block ciphers may need to be reconsidered and directly copying AES-like security proofs for some attacks seems to be less reasonable.
3. Serpent and Noekeon are two bit-oriented block ciphers. Due to their relatively complicated linear layers and large block sizes, the works about the integral cryptanalysis for these two ciphers are rarely seen. At FSE 2008 [30], Z'aba *et al.* proposed 3.5-round integral distinguishers for Noekeon and Serpent. In this paper, we observe that there is a 7-round integral distinguisher for Noekeon if the last 127 bits of the input multi-set are traversed. For Serpent, we also find some 7-round integral distinguishers. Comparing with the former results, these newly found integral distinguishers are longer. We expect that better integral cryptanalytic results for these two ciphers may be deduced with these newly proposed distinguishers.

4. SM4 (formerly SMS4) is a block cipher used in the Chinese National Standard for Wireless LAN WAPI (Wired Authentication and Privacy Infrastructure). In this paper, we find a 12-round integral distinguisher, which attains four more rounds than the one proposed by Liu [17] *et al.* at ACISP 2007.
5. SPONGENT is a family of lightweight hash functions. In this paper, we aim to find higher-order integral distinguishers for SPONGENT-88. The existing results are the 14-round higher-order integral distinguisher with complexity 2^{84} found in [12] and the 14-round higher-order integral distinguisher with complexity 2^{80} found in [13]. Sun and Wang also provided a 14-round higher-order integral distinguisher with complexity 2^{80} in [22]. In this paper, one of our resulting distinguishers attains one more round than the one in [13] while keeping the complexity. The other one is a 16-round higher-order integral distinguisher with complexity 2^{84}.

The resulting integral distinguishers in this paper and some former results are listed in **Table 1**.

Outline of the Paper. The remainder of this paper is organized as follows. In **Section 2**, we briefly review some notations, division property, bit-based division property, and 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 the paper in **Section 5**. Some auxiliary materials are given in **Appendix**.

2 Priliminary

2.1 Notations

In this subsection, we present the notations used throughout this paper. In order to simplify the representation, we will express a bit-string in hexadecimal format and make it `verbatim` font. For an n-bit bit-string, the bit positions are labeled as 0 to $n-1$ from left to right. The following notations was also used in [25].

For any $a \in \mathbb{F}_2^n$, the i-th element is expressed as $a[i]$ $(i = 0, 1, \ldots, n-1)$ 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 ϕ 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, $\boldsymbol{k} \nsucceq \boldsymbol{k'}$. $\mathbb{K} \leftarrow \boldsymbol{k}$ means that $\mathbb{K}$ turns into $\mathbb{K} \bigcup \{\boldsymbol{k}\}$.

Table 1: Summarization of Results.

Cipher	Block Size	Length	Data Requirement	#{Zero-Sum Bits}	Ref.
AES	128	4	2^{32}	128	[10]†
		4	$\mathbf{2^{32}}$	**128**	**Section 4.1**
		4	$\mathbf{2^{127}}$	**128**	
LED	64	**4**	$\mathbf{2^{16}}$	**64**	**Section 4.3**
		6	$\mathbf{2^{60}}$	**64**	
`Joltik-BC`	64	**4**	$\mathbf{2^{16}}$	**64**	**Section 4.2**
		6	$\mathbf{2^{60}}$	**64**	
PHOTON	100	5	2^{92}	100	[14]
		5	$\mathbf{2^{20}}$	**100**	**Section 4.4**
		6	2^{98}	100	[14]
		6	$\mathbf{2^{92}}$	**100**	**Section 4.4**
	144	5	2^{124}	144	[14]
		5	$\mathbf{2^{24}}$	**144**	**Section 4.4**
	196	5	2^{158}	196	[14]
		5	$\mathbf{2^{28}}$	**196**	**Section 4.4**
Serpent‡	128	3.5	2^{10}	128	[30]
		7	$\mathbf{2^{124}}$	**128**	**Section 4.5**
Noekeon	128	3.5	2^{16}	128	[30]
		6	$\mathbf{2^{124}}$	**128**	**Section 4.6**
		7	$\mathbf{2^{127}}$	**128**	
SM4	128	8	2^{8}	32	[17]
		10	$\mathbf{2^{94}}$	**32**	**Section 4.7**
		11	$\mathbf{2^{104}}$	**32**	
		12	$\mathbf{2^{125}}$	**32**	
SPONGENT-88	88	14	2^{84}	-	[12]
		14	2^{80}	-	[13]
		14	2^{80}	-	[22]
		15	$\mathbf{2^{80}}$	-	**Section 4.8**
		16	$\mathbf{2^{84}}$	-	

† Note that there is a 3-round integral distinguisher with complexity 2^8 in [10], and this distinguisher can be easily extended into a 4-round integral distinguisher with data complexity 2^{32}.

‡ For Serpent, since different rounds utilize different S-boxes, the lengths of the integral distinguishers starting from different rounds may be different. But after trying all possible cases of starting rounds, the lengths of the drawn integral distinguishers are all equal.

Definition 1 (Bit Product Function [25]). *Assume $u \in \mathbb{F}_2^n$ and $x \in \mathbb{F}_2^n$. The Bit Product Function π_u is defined as*

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

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 $\pi_{\boldsymbol{u}}$ is defined as

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

2.2 Division Property and Bit-Based Division Property

The division property is a new method to find integral distinguishers, which was firstly proposed in [25]. It is defined for a multi-set, and is calculated by using the bit product function.

Definition 2 (Division Property [25]). *Let $\mathbb{X}$ be a multi-set whose elements takes a value of $\mathbb{F}_2^{\ell_0} \times \mathbb{F}_2^{\ell_1} \times \cdots \times \mathbb{F}_2^{\ell_{m-1}}$, and $\boldsymbol{k}^{(j)} (j = 0, 1, \ldots, q-1)$ are m-dimensional vectors whose i-th element takes a value between 0 and ℓ_i. When the multi-set $\mathbb{X}$ has the division property $\mathcal{D}^{\ell_0, \ell_1, \ldots, \ell_{m-1}}_{\{\boldsymbol{k}^{(0)}, \boldsymbol{k}^{(1)}, \ldots, \boldsymbol{k}^{(q-1)}\}}$, it fulfills the following conditions: The parity of $\pi_{\boldsymbol{u}}(\boldsymbol{x})$ over all $x \in \mathbb{X}$ is always even when*

$$\boldsymbol{u} \in \left\{ \boldsymbol{u} = (u_0, u_1, \ldots, u_{m-1}) \middle| Wt(\boldsymbol{u}) \not\succeq \boldsymbol{k}^{(0)}, Wt(\boldsymbol{u}) \not\succeq \boldsymbol{k}^{(1)}, \ldots, Wt(\boldsymbol{u}) \not\succeq \boldsymbol{k}^{(q-1)} \right\}.$$

Moreover, the parity becomes unknown when $\boldsymbol{u}$ is used such that there exists an i $(0 \leqslant i \leqslant q-1)$ satisfying $Wt(\boldsymbol{u}) \succeq \boldsymbol{k}^{(i)}$.

Remark 1 *Note that ℓ_0, ℓ_1, …, ℓ_{m-1} are restricted to 1 when we consider **bit-based division property**.*

Propagation Rules of Bit-Based Division Property [25] proves some propagation rules for conventional division property and these rules are summarized into five rules in [24], which are **Substitution**, **Copy**, **Compression by `XOR`**, **Split**, and **Concatenation**, respectively. Among the five rules, only **Copy** and **Compression by `XOR`** are necessary for bit-based division property. For the propagation of S-box, the **Substitution** rule should be replaced with a subtle propagation table which is introduced in [22]. We do not review the method introduced in [22], please refer to [22] for more details. The two necessary rules are restated in a bit-based look in the following.

Rule 1 (Copy) *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 the multi-set $\mathbb{X}$ has the division property $\mathcal{D}^1_k$, the division property of the multi-set $\mathbb{Y}$ is $\mathcal{D}^{1\times 1}_{\mathbb{K}'}$. Then the propagation only have two possible cases:*

$$\begin{cases} k = 0 \rightarrow \mathbb{K}' = \{(0,0)\} \\ k = 1 \rightarrow \mathbb{K}' = \{(0,1), (1,0)\} \end{cases}$$

Rule 2 (`XOR`) *Let F be a function compressed by an* `XOR`*, where the input (x_1, x_2) takes a value of $\mathbb{F}_2 \times \mathbb{F}_2$ and the output is calculated as $y = x_1 \oplus x_2$. Let $\mathbb{X}$ and $\mathbb{Y}$ be the input multi-set and output multi-set, respectively. Assuming that the multi-set $\mathbb{X}$ has division property $\mathcal{D}_{\boldsymbol{k}}^{1\times 1}$, the division property of the multi-set $\mathbb{Y}$ is $\mathcal{D}_{\mathbb{K}'}^{1}$. Then the propagation only have four possible cases:*

$$\begin{cases} \boldsymbol{k} = (0,0) \rightarrow \mathbb{K}' = \{(0)\} \\ \boldsymbol{k} = (0,1) \rightarrow \mathbb{K}' = \{(1)\} \\ \boldsymbol{k} = (1,0) \rightarrow \mathbb{K}' = \{(1)\} \\ \boldsymbol{k} = (1,1) \rightarrow \mathbb{K}' = \emptyset \end{cases}$$

For some bit-oriented block ciphers such as SIMON, `AND` is another non-linear operation. The propagation for `AND` is given in [26] and we summarize it as follows.

Rule 3 (`AND`) *Let F be a function compressed by an* `AND`*, where the input (x_1, x_2) takes a value of $\mathbb{F}_2 \times \mathbb{F}_2$ and the output is calculated as $y = x_1 \wedge x_2$. Let $\mathbb{X}$ and $\mathbb{Y}$ be the input multi-set and output multi-set, respectively. Assuming that the multi-set $\mathbb{X}$ has division property $\mathcal{D}_{\boldsymbol{k}}^{1\times 1}$, the division property of the multi-set $\mathbb{Y}$ is $\mathcal{D}_{\mathbb{K}'}^{1}$. Then the propagation only have four possible cases:*

$$\begin{cases} \boldsymbol{k} = (0,0) \rightarrow \mathbb{K}' = \{(0)\} \\ \boldsymbol{k} = (0,1) \rightarrow \mathbb{K}' = \{(1)\} \\ \boldsymbol{k} = (1,0) \rightarrow \mathbb{K}' = \{(1)\} \\ \boldsymbol{k} = (1,1) \rightarrow \mathbb{K}' = \{(1)\} \end{cases}$$

2.3 MILP-Aided Bit-Based Division Property

At ASIACRYPT 2016, Xiang *et al.* proposed the method of characterising the bit-based division property with the MILP model. In this subsection, we will give a brief review.

When we want to utilize MILP method to solve a problem, a stopping rule is necessary. To describe the stopping rule, Xiang *et al.* introduced the following definition in [28].

Definition 3 (Division Trail). *Let f_r denote the round function of an iterated block cipher. Assume that the input multi-set to the block cipher has initial division property $\mathcal{D}_{\boldsymbol{k}}^{1^n}$, and denote the division property after i-round propagation through f_r by $\mathcal{D}_{\mathbb{K}_i}^{1^n}$. Thus we have the following chain of division property propagations:*

$$\{\boldsymbol{k}\} \triangleq \mathbb{K}_0 \xrightarrow{f_r} \mathbb{K}_1 \xrightarrow{f_r} \mathbb{K}_2 \xrightarrow{f_r} \cdots .$$

Moreover, for any vector $\boldsymbol{k}_i^*$ in $\mathbb{K}_i$ $(i \geqslant 1)$, there must exist an 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.

Remark 2 *The considered division property for the above definition is restricted to $\mathcal{D}_{*}^{1^n}$, i.e., only the bit-based division property for an n-bit block cipher is considered. However, the* ***Definition 2*** *of division trail given in [28] is defined for conventional division property. Since we only consider bit-based division property, the above definition is sufficient.*

Then the following proposition can help us to describe the stopping rule.

Proposition 1. *Denote the division property of the input multi-set to an iterated block cipher by $D_{\boldsymbol{k}}^{1^n}$, lef f_r be the round function. Denote*

$$\{\boldsymbol{k}\} \triangleq \mathbb{K}_0 \xrightarrow{f_r} \mathbb{K}_1 \xrightarrow{f_r} \mathbb{K}_2 \xrightarrow{f_r} \cdots \xrightarrow{f_r} \mathbb{K}_r$$

the r-round division property propagation. Thus the set of the last vectors of all r-round division trails which start with $\boldsymbol{k}$ is equal to $\mathbb{K}_r$.

Remark 3 *Again, the proposition given above holds for bit-based division property. But the* ***Proposition 5*** *of [28] is given for conventional division property. Since we only consider bit-based division property, the above proposition is sufficient.*

According to **Proposition 1**, checking whether there exists useful zero-sum property is equivalent to find all r-round division trails which start with $\boldsymbol{k}$, and check the last vectors in the division trails to judge if any available distinguisher can be extracted.

Modeling Copy, `AND` and `XOR` In the following, we briefly review the method of modeling bit-wise **Copy**, `AND`, and `XOR` operations by linear inequalities introduced in [28].

Model 1 (Copy [28]) *Denote $(a) \xrightarrow{Copy} (b_0, b_1)$ a division trail of Copy function, 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` [28]) *Denote $(a_0, a_1) \xrightarrow{AND} (b)$ a division trail of `AND` function, 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` [28]) *Denote $(a_0, a_1) \xrightarrow{XOR} (b)$ a division trail through `XOR` function, the following inequalities can describe the division trail through `XOR` function.*

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

Modelling S-box For the propagation of S-box, we firstly apply table-aided bit-based division property introduced in [22] to generate the propagation table of the S-box.[4] After that, just as what has been introduced in [28], by using the `inequality_generator()` function in the Sage[5] software, a set of linear inequalities will be returned. Furthermore, this set can be reduced by **Algorithm 1** (**Greedy Algorithm**) in [28].

Initial Division Property Integral distinguisher searching algorithm often has a given initial division property $\mathcal{D}^{1^n}_{\boldsymbol{k}}$. To model the division trails starting from the given initial division property, we have to model the initial division property into the linear inequality system. Denote $(a^0_0, a^0_1, \ldots, a^0_{n-1}) \rightarrow \cdots \rightarrow (a^r_0, a^r_1, \ldots, a^r_{n-1})$ an r-round division trail, $\mathcal{L}$ is a linear inequality system defined on variables a^j_i $(i = 0, 1, \cdots, n-1, j = 0, 1, \cdots, r)$ and some auxiliary variables. Let $\mathcal{D}^{1^n}_{\boldsymbol{k}}$ denote the initial input division property with $\boldsymbol{k} = (k_0, k_1, \ldots, k_{n-1})$, we need to add $a^0_i = k_i$ $(i = 0, 1, \ldots, n-1)$ into $\mathcal{L}$, and all feasible solutions of $\mathcal{L}$ are division trails which start from vector $\boldsymbol{k}$.

Stopping Rule

Proposition 2 (Proposition 6, [28]). *Assume $\mathbb{X}$ is a multi-set with division property $\mathcal{D}^{1^n}_{\mathbb{K}}$, then $\mathbb{X}$ does not have integral property if and only of $\mathbb{K}$ contains all the n unit vectors.*

Let $\mathcal{D}^{1^n}_{\mathbb{K}_i}$ denote the output division property after i rounds of encryption and the input division property is denoted by $\mathcal{D}^{1^n}_{\mathbb{K}_0}$. 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 $\mathcal{D}^{1^n}_{\mathbb{K}_r}$.

Based on this observation, we only need to detect whether $\mathbb{K}_r$ contains all unit vectors. According to **Proposition 1**, checking the vectors in $\mathbb{K}_r$ is equivalent to check the last vectors of all r-round division trails. Let $(a^0_0, a^0_1, \ldots, a^0_{n-1}) \rightarrow \cdots \rightarrow (a^r_0, a^r_1, \ldots, a^r_{n-1})$ be an r-round division trail, and let $\mathcal{L}$ be a linear inequality system whose feasible solutions are all division trails which start from the given input division property. Then the objective function is set as

$$Obj : Min\{a^r_0 + a^r_1 + \cdots + a^r_{n-1}\}.$$

Note that we only review some key points here. For more details about division property, bit-based division property, table-aided bit-based division property, and MILP-aided bit-based division property, please refer to [22, 24, 25, 28].

[4] Another method to generate the propagation of the S-box was introduced in [28]. Both of these two methods consider the Boolean function expressions of the S-box. Although the starting points of them are different, the resulting propagations are exactly the same.

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

3 Modelling Complicated Linear Layers

In [28], the feasibility of MILP method applying to ciphers with more complicated linear layers was left as a future work. In this paper, we settle this problem by introducing some intermediate variables among the linear layer. Following this way, MILP-aided bit-based division property can be applied to more primitives.

3.1 Generalizing the Original Models for Copy and `XOR`

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* of the linear transformation.

No matter how complicated the linear layer is, the linear layer can always be splitted into **Copy** and `XOR` operations. To establish the method of modelling complicated linear layers, we need to generalize **Model 1** and **Model 3**.

Model 4 (Generalizaion of Copy) *Denote* $(a) \xrightarrow{Copy} (b_0, b_1, \ldots, b_m)$ *a division trail of Copy function, the following inequalities are sufficient to describe the division propagation of copy.*

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

Model 5 (Generalization of `XOR`) *Denote* $(a_0, a_1, \ldots, a_m) \xrightarrow{XOR} (b)$ *a division trail through `XOR` function, the following inequalities can describe the division trail through `XOR` function.*

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

With **Model 4** and **Model 5**, we can characteristic the propagation of the linear layer by introducing some intermediate variables.

3.2 Modelling the `MixNibbles` of `Joltik-BC`

In the following, we use `Joltik-BC`'s linear layer [16] as an illustration. `Joltik-BC` is an AES-like block ciphers. `MixNibbles` is a linear operation of `Joltik-BC`'s round function, and it is like the `MixColumn` operation for AES [20]. It operates by multiplying every column of the internal state by a 4×4 constant MDS matrix $M_{\texttt{Joltik-BC}}$.

For `Joltik-BC`, the primitive representation of $M_{\texttt{Joltik-BC}}$ is

$$M_{\texttt{Joltik-BC}} = \left(\begin{array}{cccc|cccc|cccc|cccc}
1&0&0&0&0&0&1&0&0&0&0&1&0&0&1&1\\
0&1&0&0&1&0&0&1&1&0&0&0&0&0&0&1\\
0&0&1&0&1&1&0&0&0&1&0&0&1&0&0&0\\
0&0&0&1&0&1&0&0&0&0&1&1&0&1&1&1\\
\hline
0&0&1&0&1&0&0&0&0&0&1&1&0&0&0&1\\
1&0&0&1&0&1&0&0&0&0&0&1&1&0&0&0\\
1&1&0&0&0&0&1&0&1&0&0&0&0&1&0&0\\
0&1&0&0&0&0&0&1&0&1&1&1&0&0&1&1\\
\hline
0&0&0&1&0&0&1&1&1&0&0&0&0&0&1&0\\
1&0&0&0&0&0&0&1&0&1&0&0&1&0&0&1\\
0&1&0&0&1&0&0&0&0&0&1&0&1&1&0&0\\
0&0&1&1&0&1&1&1&0&0&0&1&0&1&0&0\\
\hline
0&0&1&1&0&0&0&1&0&0&1&0&1&0&0&0\\
0&0&0&1&1&0&0&0&1&0&0&1&0&1&0&0\\
1&0&0&0&0&1&0&0&1&1&0&0&0&0&1&0\\
0&1&1&1&0&0&1&1&0&1&0&0&0&0&0&1
\end{array}\right). \tag{1}$$

Suppose that the input multi-set of $M_{\texttt{Joltik-BC}}$ satisfies division property $\mathcal{D}_{\boldsymbol{x}}^{1^{16}}$, where $\boldsymbol{x} = (x_0, x_1, \ldots, x_{15})$. From the first column of (1), we known that x_0 is copied five times, and these copies should `XOR` with different branches of the output to generate the division property of output multi-set. The remaining x_i, $1 \leqslant i \leqslant 15$, are also handled with similar operations.

There are 88 non-zero elements in $M_{\texttt{Joltik-BC}}$. By introducing intermediate variables $t_0 \sim t_{87}$, $M_{\texttt{Joltik-BC}}$ is transformed into the following form

$$\left(\begin{array}{cccc|cccc|cccc|cccc}
t_0&0&0&0&0&0&t_{32}&0&0&0&0&t_{59}&0&0&t_{76}&t_{81}\\
0&t_5&0&0&t_{22}&0&0&t_{37}&t_{44}&0&0&0&0&0&0&t_{82}\\
0&0&t_{10}&0&t_{23}&t_{27}&0&0&0&t_{49}&0&0&t_{66}&0&0&0\\
0&0&0&t_{15}&0&t_{28}&0&0&0&0&t_{54}&t_{60}&0&t_{71}&t_{77}&t_{83}\\
\hline
0&0&t_{11}&0&t_{24}&0&0&0&0&0&t_{55}&t_{61}&0&0&0&t_{84}\\
t_1&0&0&t_{16}&0&t_{29}&0&0&0&0&0&t_{62}&t_{67}&0&0&0\\
t_2&t_6&0&0&0&0&t_{33}&0&t_{45}&0&0&0&0&t_{72}&0&0\\
0&t_7&0&0&0&0&0&t_{38}&0&t_{50}&t_{56}&t_{63}&0&0&t_{78}&t_{85}\\
\hline
0&0&0&t_{17}&0&0&t_{34}&t_{39}&t_{46}&0&0&0&0&0&t_{79}&0\\
t_3&0&0&0&0&0&0&t_{40}&0&t_{51}&0&0&t_{68}&0&0&t_{86}\\
0&t_8&0&0&t_{25}&0&0&0&0&0&t_{57}&0&t_{69}&t_{73}&0&0\\
0&0&t_{12}&t_{18}&0&t_{30}&t_{35}&t_{41}&0&0&0&t_{64}&0&t_{74}&0&0\\
\hline
0&0&t_{13}&t_{19}&0&0&0&t_{42}&0&0&t_{58}&0&t_{70}&0&0&0\\
0&0&0&t_{20}&t_{26}&0&0&0&t_{47}&0&0&t_{65}&0&t_{75}&0&0\\
t_4&0&0&0&0&t_{31}&0&0&t_{48}&t_{52}&0&0&0&0&t_{80}&0\\
0&t_9&t_{14}&t_{21}&0&0&t_{36}&t_{43}&0&t_{53}&0&0&0&0&0&t_{87}
\end{array}\right). \tag{2}$$

Thus, to describe the **Copy** operations stated above, the following 16 linear inequalities are sufficient by applying **Model 4**.

$$\begin{cases} x_0 - t_0 - t_1 - t_2 - t_3 - t_4 = 0 \\ x_1 - t_5 - t_6 - t_7 - t_8 - t_9 = 0 \\ x_2 - t_{10} - t_{11} - t_{12} - t_{13} - t_{14} = 0 \\ x_3 - t_{15} - t_{16} - t_{17} - t_{18} - t_{19} - t_{20} - t_{21} = 0 \\ x_4 - t_{22} - t_{23} - t_{24} - t_{25} - t_{26} = 0 \\ x_5 - t_{27} - t_{28} - t_{29} - t_{30} - t_{31} = 0 \\ x_6 - t_{32} - t_{33} - t_{34} - t_{35} - t_{36} = 0 \\ x_7 - t_{37} - t_{38} - t_{39} - t_{40} - t_{41} - t_{42} - t_{43} = 0 \\ x_8 - t_{44} - t_{45} - t_{46} - t_{47} - t_{48} = 0 \\ x_9 - t_{49} - t_{50} - t_{51} - t_{52} - t_{53} = 0 \\ x_{10} - t_{54} - t_{55} - t_{56} - t_{57} - t_{58} = 0 \\ x_{11} - t_{59} - t_{60} - t_{61} - t_{62} - t_{63} - t_{64} - t_{65} = 0 \\ x_{12} - t_{66} - t_{67} - t_{68} - t_{69} - t_{70} = 0 \\ x_{13} - t_{71} - t_{72} - t_{73} - t_{74} - t_{75} = 0 \\ x_{14} - t_{76} - t_{77} - t_{78} - t_{79} - t_{80} = 0 \\ x_{15} - t_{81} - t_{82} - t_{83} - t_{84} - t_{85} - t_{86} - t_{87} = 0 \\ x_i, t_i \text{ are binaries} \end{cases} \tag{3}$$

On the other hand, the copies of x_i should `XOR` with the corresponding output bit potisitons. Denote $(x_0, x_1, \ldots, x_{15}) \xrightarrow{M_{\texttt{Joltik-BC}}} (y_0, y_1, \ldots, y_{15})$ a division trail of $M_{\texttt{Joltik-BC}}$ operation. From (2), we know that the variables located at the same line are those variables `XOR`ing to the same branch of the output. Thus, to describe the `XOR` operations, the following 16 linear inequalities are sufficient by using **Model 5**.

$$\begin{cases} t_0 + t_{32} + t_{59} + t_{76} + t_{81} - y_0 = 0 \\ t_5 + t_{22} + t_{37} + t_{44} + t_{82} - y_1 = 0 \\ t_{10} + t_{23} + t_{27} + t_{49} + t_{66} - y_2 = 0 \\ t_{15} + t_{28} + t_{54} + t_{60} + t_{71} + t_{77} + t_{83} - y_3 = 0 \\ t_{11} + t_{24} + t_{55} + t_{61} + t_{84} - y_4 = 0 \\ t_1 + t_{16} + t_{29} + t_{62} + t_{67} - y_5 = 0 \\ t_2 + t_6 + t_{33} + t_{45} + t_{72} - y_6 = 0 \\ t_7 + t_{38} + t_{50} + t_{56} + t_{63} + t_{78} + t_{85} - y_7 = 0 \\ t_{17} + t_{34} + t_{39} + t_{46} + t_{79} - y_8 = 0 \\ t_3 + t_{40} + t_{51} + t_{68} + t_{86} - y_9 = 0 \\ t_8 + t_{25} + t_{57} + t_{69} + t_{73} - y_{10} = 0 \\ t_{12} + t_{18} + t_{30} + t_{35} + t_{41} + t_{64} + t_{74} - y_{11} = 0 \\ t_{13} + t_{19} + t_{42} + t_{58} + t_{70} - y_{12} = 0 \\ t_{20} + t_{26} + t_{47} + t_{65} + t_{75} - y_{13} = 0 \\ t_4 + t_{31} + t_{48} + t_{52} + t_{80} - y_{14} = 0 \\ t_9 + t_{14} + t_{21} + t_{36} + t_{43} + t_{53} + t_{87} - y_{15} = 0 \\ y_i, t_i \text{ are binaries} \end{cases} \tag{4}$$

In summary, to characterize the division property propagation of $M_{\texttt{Joltik-BC}}$, we only need to combine linear inequality systems (3) and (4) and make it a bigger linear inequality system.

4 Applications of MILP-Aided Bit-Based Division Property

In this section, we show some applications of MILP-aided bit-based division property.

4.1 Application to AES

AES [20] AES, also known as Rijndael [10], is a SPN based symmetric cipher. AES has a fixed block size of 128 bits, and a key size of 128, 192 or 256 bits. The 128-bit plaintext and the intermediate state are commonly treated as byte matrices of size 4×4. Each round is composed of four operations applied to the internal state in the order specified below:

– `SubByte`(SB): Applying the 8-bit S-box on each byte.
– `ShiftRow`(SR): The i-th row is shifted by i bytes to the left, $i = 0, 1, 2, 3$.
– `MixColumn`(MC): Multiplying each column by a constant 4×4 matrix M_{AES} over the field $\mathbb{F}_{2^8}$, where M_{AES} is

$$M_{\mathrm{AES}} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}.$$

And the base field is defined by the irreducible polynomial $x^8+x^4+x^3+x+1$.
– `AddRoundKey`(AK): `XOR` the state and a 128-bit subkey.

Before the first round, an additional `AddRoundKey` operation is performed to deal with the plaintext.

Since `XOR`ing with constants does not influence the division property, we do not consider `AddRoundKey` in our analysis. Thus the key schedule of AES is not introduced. For more details about AES, please refer to [20].

Former Results of AES According to the structural property of AES, there exist some 4-round integral distinguishers if we traverse 32 specific bits of the input.

Applying MILP-Aided Bit-Based Division Property to AES The propagation table for the S-box of AES has 2001 elements. By using the function called `inequality_generator()` in the Sage software, 179657 linear inequalities are returned. After using **Greedy Algorithm**, we get 32 linear inequalities.

These linear inequalities are sufficient to describe the division property propagation of the AES's S-box and they are given in **Appendix A**. There are 184 non-zero elements in the primitive representation of M_{AES} and $184 \times 4 = 736$ intermediate variables ($t_0 \sim t_{735}$) are required for one round of encryption. The naming rule of variables for one round of encryption is illustrated in **Fig. 1**. The naming rule determines the order of elements in division property. For example, if we say that the input multi-set of AES satisfies the division property $\mathcal{D}_{\boldsymbol{k}}^{1^{128}}$, where $\boldsymbol{k} = (k_0, k_1, \ldots, k_{127})$, we mean that $a_0 = k_0$, $a_1 = k_1$, …, $a_{127} = k_{127}$. In the following, the naming rule has exactly the same meaning.

$$\left[\begin{array}{c|c|c|c} a_0 \sim a_7 & a_{32} \sim a_{39} & a_{64} \sim a_{71} & a_{96} \sim a_{103} \\ \hline a_8 \sim a_{15} & a_{40} \sim a_{47} & a_{72} \sim a_{79} & a_{104} \sim a_{111} \\ \hline a_{16} \sim a_{23} & a_{48} \sim a_{55} & a_{80} \sim a_{87} & a_{112} \sim a_{119} \\ \hline a_{24} \sim a_{31} & a_{56} \sim a_{63} & a_{88} \sim a_{95} & a_{120} \sim a_{127} \end{array}\right] \xrightarrow{\text{SB}} \left[\begin{array}{c|c|c|c} b_0 \sim b_7 & b_{32} \sim b_{39} & b_{64} \sim b_{71} & b_{96} \sim b_{103} \\ \hline b_8 \sim b_{15} & b_{40} \sim b_{47} & b_{72} \sim b_{79} & b_{104} \sim b_{111} \\ \hline b_{16} \sim b_{23} & b_{48} \sim b_{55} & b_{80} \sim b_{87} & b_{112} \sim b_{119} \\ \hline b_{24} \sim b_{31} & b_{56} \sim b_{63} & b_{88} \sim b_{95} & b_{120} \sim b_{127} \end{array}\right]$$

$$\xrightarrow{\text{SR}} \left[\begin{array}{c|c|c|c} b_0 \sim b_7 & b_{32} \sim b_{39} & b_{64} \sim b_{71} & b_{96} \sim b_{103} \\ \hline b_{40} \sim b_{47} & b_{72} \sim b_{79} & b_{104} \sim b_{111} & b_8 \sim b_{15} \\ \hline b_{80} \sim b_{87} & b_{112} \sim b_{119} & b_{16} \sim b_{23} & b_{48} \sim b_{55} \\ \hline b_{120} \sim b_{127} & b_{24} \sim b_{31} & b_{56} \sim b_{63} & b_{88} \sim b_{95} \end{array}\right]$$

$$\xrightarrow[t_0 \sim t_{735}]{\text{MC}} \left[\begin{array}{c|c|c|c} a_{128} \sim a_{135} & a_{160} \sim a_{167} & a_{192} \sim a_{199} & a_{224} \sim a_{231} \\ \hline a_{136} \sim a_{143} & a_{168} \sim a_{175} & a_{200} \sim a_{207} & a_{232} \sim a_{239} \\ \hline a_{144} \sim a_{151} & a_{176} \sim a_{183} & a_{208} \sim a_{215} & a_{240} \sim a_{247} \\ \hline a_{152} \sim a_{159} & a_{184} \sim a_{191} & a_{216} \sim a_{223} & a_{248} \sim a_{255} \end{array}\right]$$

Fig. 1: Variables for One Round of AES.

4-Round Integral Distinguisher Let the division property of the input multi-set be $\mathcal{D}^{1^{128}}_{\{\texttt{[ff000000,00ff0000,0000ff00,000000ff]}\}}$, *i.e.*, we traverse the 32 bits located at the diagonal of the matrix. We find that the objective function is equal to 2 after four rounds of encryption, which indicates that all the 128 bits satisfy zero-sum property after four rounds of encryption. The objective function is equal to 1 after five rounds of encryption and the experimental results show that all the 128 unit vectors exist under this situation. This fact indicates that there does not exist any bit satisfying zero-sum property after five rounds of encryption if we only traverse the 32 bits located at the specific position of the input. The experimental results are also in accordance with the theoretical deduction.

4-Round Integral Distinguisher Let the division property of the input multi-set be $\mathcal{D}^{1^{128}}_{\{\texttt{[ffffffff,ffffffff,ffffffff,ffffff00]}\}}$, *i.e.*, we traverse the first 120 bits. We find that the objective function is equal to 4 after four rounds of encryption, which indicates that all the 128 bits satisfy zero-sum property after four rounds of encryption. The objective function is equal to 1 after five rounds of encryption and the experimental results show that all the 128 unit vectors occur under this setting. This fact tells that there does not exist any bit satisfying zero-sum property after five rounds of encryption if we only traverse the specific 120 bits at the input.

4-Round Integral Distinguisher Let the division property of the input multiset be $\mathcal{D}^{1^{128}}_{\{[\texttt{ffffffff},\texttt{ffffffff},\texttt{ffffffff},\texttt{fffffffe}]\}}$, *i.e.*, we traverse the first 127 bits. We find that the objection function is equal to 1 after five rounds of encryption and the experimental results show that all the 128 unit vectors exist under this setting. This fact indicates that there does not exist any bit satisfying zero-sum property after five rounds of encryption even though we traverse the specific 127 bits at the input.

4.2 Application to `Joltik-BC`

`Joltik-BC` [**16**] `Joltik-BC` is a 64-bit lightweight tweakable block cipher that uses an AES-like round function as a building block. It is a building block of `Joltik`, which is a new authenticated encryption disign. As most AES-like designs, the state of `Joltik-BC` is seen as 4×4 matrix of nibbles. The base field is $\mathbb{F}_{2^4}$ and is defined by the irreducible polynomial $x^4 + x + 1$. `Joltik-BC` has two versions depending on the length of the tweak. The number r of rounds is 24 for `Joltik-BC-128` and 32 for `Joltik-BC-192`. One round has the following four transformations applied to the internal state in the order specified below:

- `AddRoundTweakey`(AT): `XOR` the 64-bit round subtweakey to the internal state.
- `SubNibble`(SN): Apply the 4-bit S-box to the 16 nibbles of the internal state.
- `ShiftRows`(SR): Rotate the 4-nibble i-th row left by $\rho[i]$ positions, where $\rho = 0, 1, 2, 3$.
- `MixNibbles`(MN): Multiply the internal state by the 4×4 constant MDS matrix $M_{\texttt{Joltik-BC}}$.

After the last round, a final `AddRoundTweakey` operation is performed to produce the ciphertext.

The 4-bit S-box $S_{\texttt{Joltik-BC}}$ is the one selected for Piccolo [21], and is given in **Table 2**. The MDS matrix $M_{\texttt{Joltik-BC}}$ with coefficients in $\mathbb{F}_{2^4}$ used in `Joltik-BC` is non-circulant, MDS and involutory:

$$M_{\texttt{Joltik-BC}} = \begin{pmatrix} \texttt{1} & \texttt{4} & \texttt{9} & \texttt{d} \\ \texttt{4} & \texttt{1} & \texttt{d} & \texttt{9} \\ \texttt{9} & \texttt{d} & \texttt{1} & \texttt{4} \\ \texttt{d} & \texttt{9} & \texttt{4} & \texttt{1} \end{pmatrix} = M^{-1}_{\texttt{Joltik-BC}}.$$

Since `XOR`ing with constant does not influence the division property, we do not consider `AddRoundTweakey` in our analysis. For more details about `Joltik` and `Joltik-BC`, please refer to [16].

Table 2: `Joltik-BC`'s S-Box $S_{\texttt{Joltik-BC}}$ [16]

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S[x]$	e	4	b	2	3	8	0	9	1	a	7	f	6	c	5	d

$$\begin{cases} x_0+x_1+x_2+x_3-y_0-y_1-y_2-y_3 \geqslant 0 \\ -2x_0-x_1-x_2-3x_3+2y_1+y_2+2y_3 \geqslant -3 \\ -x_0-x_1-x_3+y_0-y_1+y_2 \geqslant -2 \\ x_3-y_0+y_1-y_2-y_3 \geqslant -1 \\ 3x_0-y_0-y_1-y_2-2y_3 \geqslant -2 \\ -x_0-x_2+2x_3+y_0-2y_1-y_2-y_3 \geqslant -3 \\ -x_0-x_1-2x_3+2y_0+y_1+2y_2+3y_3 \geqslant 0 \\ -x_2-y_0+y_1 \geqslant -1 \\ x_1+x_2-y_0-y_2-y_3 \geqslant -1 \\ x_0+x_1+x_3-y_0-2y_1-2y_2 \geqslant -2 \\ x_2+2x_3-y_0-y_1-y_2-y_3 \geqslant -1 \\ -x_0-x_2-x_3+y_0+y_1+2y_2+2y_3 \geqslant 0 \\ x_0+x_3-y_0-y_1-y_3 \geqslant -1 \\ x_i, y_i \text{ are binaries} \end{cases} \tag{5}$$

Applying MILP-Aided Bit-Based Division Property to `Joltik-BC` The naming rule of variables for one round of encryption is illustrated in **Fig. 2**. From **Section 3**, We know that there are 88 non-zero elements in the primitive representation of $M_{\texttt{Joltik-BC}}$. Since there are four columns in the internal state of `Joltik-BC`, $88 \times 4 = 352$ intermediate variables ($t_0 \sim t_{351}$) are needed for one round of encryption.

The propagation table for `Joltik-BC`'s S-box have 46 elements, which are given in **Table 3**. By using the `inequality_generator()` in the Sage software, 100 linear inequalities are returned. After utilizing **Greedy Algorithm** introduced in [28], we get 13 linear inequalities, which are given in linear inequality system (5), where $(x_0, x_1, x_2, x_3) \to (y_0, y_1, y_2, y_3)$ denotes a division trail for the S-box.

4-Round Integral Distinguisher Let the division property of the input multiset be $\mathcal{D}^{1^{64}}_{\{[\texttt{f000,0f00,00f0,000f}]\}}$, *i.e.*, we traverse the 16 bits which are located at the diagonal of the matrix. We find that the objective function is equal to 2 after four rounds of encryption, which indicates that all 64 bits satisfy zero-sum property after four rounds of encryption. The objective function is equal to 1 after five rounds of encryption and the experimental results show that all the 64 unit vectors exist under this setting. This fact indicates that there does not exist

$$\begin{bmatrix} a_0 \sim a_3 & a_{16} \sim a_{19} & a_{32} \sim a_{35} & a_{48} \sim a_{51} \\ a_4 \sim a_7 & a_{20} \sim a_{23} & a_{36} \sim a_{39} & a_{52} \sim a_{55} \\ a_8 \sim a_{11} & a_{24} \sim a_{27} & a_{40} \sim a_{43} & a_{56} \sim a_{59} \\ a_{12} \sim a_{15} & a_{28} \sim a_{31} & a_{44} \sim a_{47} & a_{60} \sim a_{63} \end{bmatrix} \xrightarrow{\text{SN}} \begin{bmatrix} b_0 \sim b_3 & b_{16} \sim b_{19} & b_{32} \sim b_{35} & b_{48} \sim b_{51} \\ b_4 \sim b_7 & b_{20} \sim b_{23} & b_{36} \sim b_{39} & b_{52} \sim b_{55} \\ b_8 \sim b_{11} & b_{24} \sim b_{27} & b_{40} \sim b_{43} & b_{56} \sim b_{59} \\ b_{12} \sim b_{15} & b_{28} \sim b_{31} & b_{44} \sim b_{47} & b_{60} \sim b_{63} \end{bmatrix}$$

$$\xrightarrow{\text{SR}} \begin{bmatrix} b_0 \sim b_3 & b_{16} \sim b_{19} & b_{32} \sim b_{35} & b_{48} \sim b_{51} \\ b_{20} \sim b_{23} & b_{36} \sim b_{39} & b_{52} \sim b_{55} & b_4 \sim b_7 \\ b_{40} \sim b_{43} & b_{56} \sim b_{59} & b_8 \sim b_{11} & b_{24} \sim b_{27} \\ b_{60} \sim b_{63} & b_{12} \sim b_{15} & b_{28} \sim b_{31} & b_{44} \sim b_{47} \end{bmatrix}$$

$$\xrightarrow[t_0 \sim t_{351}]{\text{MN}} \begin{bmatrix} a_{64} \sim a_{67} & a_{80} \sim a_{83} & a_{96} \sim a_{99} & a_{112} \sim a_{115} \\ a_{68} \sim a_{71} & a_{84} \sim a_{87} & a_{100} \sim a_{103} & a_{116} \sim a_{119} \\ a_{72} \sim a_{75} & a_{88} \sim a_{91} & a_{104} \sim a_{107} & a_{120} \sim a_{123} \\ a_{76} \sim a_{79} & a_{92} \sim a_{95} & a_{108} \sim a_{111} & a_{124} \sim a_{127} \end{bmatrix}$$

Fig. 2: Variables for One Round of `Joltik-BC`.

Table 3: Propagation of Division Property for `Joltik-BC`'s S-box

Input $\mathcal{D}^{1^4}_{\boldsymbol{k}}$	Output $\mathcal{D}^{1^4}_{\mathbb{K}'}$
$\boldsymbol{k} = [0,0,0,0]$	$\mathbb{K}' = \{[0,0,0,0]\}$
$\boldsymbol{k} = [0,0,0,1]$	$\mathbb{K}' = \{[0,0,0,1],[0,0,1,0],[1,0,0,0]\}$
$\boldsymbol{k} = [0,0,1,0]$	$\mathbb{K}' = \{[0,0,0,1],[0,0,1,0],[0,1,0,0]\}$
$\boldsymbol{k} = [0,0,1,1]$	$\mathbb{K}' = \{[0,0,0,1],[0,0,1,0],[1,1,0,0]\}$
$\boldsymbol{k} = [0,1,0,0]$	$\mathbb{K}' = \{[0,0,0,1],[0,0,1,0],[0,1,0,0],[1,0,0,0]\}$
$\boldsymbol{k} = [0,1,0,1]$	$\mathbb{K}' = \{[0,0,0,1],[0,1,1,0],[1,0,1,0],[1,1,0,0]\}$
$\boldsymbol{k} = [0,1,1,0]$	$\mathbb{K}' = \{[0,0,0,1],[0,0,1,0],[0,1,0,0]\}$
$\boldsymbol{k} = [0,1,1,1]$	$\mathbb{K}' = \{[0,0,0,1],[0,1,1,0],[1,1,0,0]\}$
$\boldsymbol{k} = [1,0,0,0]$	$\mathbb{K}' = \{[0,0,0,1],[0,0,1,0],[0,1,0,0],[1,0,0,0]\}$
$\boldsymbol{k} = [1,0,0,1]$	$\mathbb{K}' = \{[0,0,0,1],[0,1,1,0],[1,1,0,0]\}$
$\boldsymbol{k} = [1,0,1,0]$	$\mathbb{K}' = \{[0,0,0,1],[0,0,1,0],[1,1,0,0]\}$
$\boldsymbol{k} = [1,0,1,1]$	$\mathbb{K}' = \{[0,0,1,1],[0,1,1,0],[1,1,0,1]\}$
$\boldsymbol{k} = [1,1,0,0]$	$\mathbb{K}' = \{[0,0,0,1],[0,0,1,0],[1,0,0,0]\}$
$\boldsymbol{k} = [1,1,0,1]$	$\mathbb{K}' = \{[0,0,1,1],[1,1,1,0]\}$
$\boldsymbol{k} = [1,1,1,0]$	$\mathbb{K}' = \{[0,0,0,1],[0,0,1,0],[1,1,0,0]\}$
$\boldsymbol{k} = [1,1,1,1]$	$\mathbb{K}' = \{[1,1,1,1]\}$

any bit satisfying zero-sum property after five rounds of encryption if we only traverse the 16 bits located at the specific positions. The experimental results are also in accordance with the theoretical deduction.

6-Round Integral Distinguisher Let the division property of the input multiset be $\mathcal{D}^{1^{64}}_{\{[\texttt{0fff,ffff,ffff,ffff}]\}}$, *i.e.*, we traverse the last 60 bits of the input. We find that the objective function is equal to 2 after six rounds of encryption, which indicates that all the 64 bits satisfy zero-sum property after six rounds of encryption. The objective function is equal to 1 after seven rounds of encryption and the experimental results show that all the 64 unit vectors occur. This fact indicates that there does not exist any bit satisfying zero-sum property after seven rounds of encryption if we only traverse the last 60 bits at the input.

We also try to propagate the input division property $\mathcal{D}^{1^{64}}_{\{[\texttt{7fff,ffff,ffff,ffff}]\}}$, but the objective function is equal to 1 after seven rounds of encryption and the experimental results tell that all 64 unit vectors exist. This observation illustrates that we can not go on extending the length of the distinguisher even though we traverse more bits at the input.

4.3 Application to LED

LED [15] LED, which is an AES-like block cipher, has 64-bit block size with two primary instances taking 64 and 128 bit keys. The cipher state is conceptually arranged in a 4×4 matrix where each nibble represents an element from $\mathbb{F}_{2^4}$ with the underlying polynomial for field multiplication given by $x^4 + x + 1$. The basic structure of LED consists of alternating operations of `AddRoundKey` and `Step`. The number of `Step`'s s during encryption depends on the key size. For 64-bit key, $s = 8$. For bigger key sizes up to 128 bits, $s = 12$. The illustration for the encryption of LED is given in **Fig. 3**.

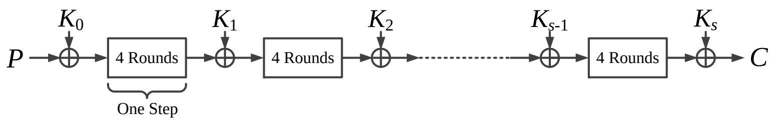

Fig. 3: An Illustration of LED.

Each round is composed of the following four operations applied to the internal state in the order specified below:

- `AddConstants`(AC): The state is combined with the pre-defined constant matrix, using bitwise `XOR`.
- `SubCells`(SC): Each nibble in the state is replaced by the nibble generated after using the PRESENT's [5] S-box.

- `ShiftRow`(SR): Row i of the state is rotated i nibbles to the left, for $i = 0, 1, 2, 3$.
- `MixColumns`(MC): Multiplying each column by a constant 4×4 matrix M_{LED} over the field $\mathbb{F}_{2^4}$, where M_{LED} is

$$M_{\text{LED}} = \begin{pmatrix} \texttt{4} & \texttt{1} & \texttt{2} & \texttt{2} \\ \texttt{8} & \texttt{6} & \texttt{5} & \texttt{6} \\ \texttt{b} & \texttt{e} & \texttt{a} & \texttt{9} \\ \texttt{2} & \texttt{2} & \texttt{f} & \texttt{b} \end{pmatrix}.$$

Since `AddRoundKey` and `AddConstants` operations does not affect the propagation of division property, we do not consider these operations in our analysis. Thus the way of generating constants and the key schedule are left behind. For more details about LED, please refer to [15].

Former results of LED There are several known 4-round integral distinguishers for LED. For example, we know that all 64 bits satisfy zero-sum property after four rounds of encryption if we traverse the 16 bits located at the diagonal of the matrix.

Applying MILP-Aided Bit-Based Division Property to LED Since LED takes PRESENT's S-box as its S-box and the propagation table and linear inequality system for PRESENT's S-box are already given in [28], these details are not covered again here. There are 124 non-zero elements in the primitive representation of M_{LED} and $124 \times 4 = 496$ intermediate variables ($t_0 \sim t_{495}$) are required for one round of encryption. The naming rule of variables for LED is exactly the same to the one for `Joltik-BC` except for the number of the intermediate variables.

4-Round Integral Distinguisher Let the division property of the input multiset be $\mathcal{D}^{1^{64}}_{\{\texttt{[f000,0f00,00f0,000f]}\}}$, *i.e.*, we traverse the 16 bits located at the diagonal of the matrix. We find that the objective function is equal to 3 after four rounds of encryption. The objective function is equal to 1 after five rounds of encryption and the experimental results show that all the 64 unit vectors occur under this setting. This fact indicates that there does not exist any bit satisfying zero-sum property after five rounds of encryption if we only traverse the corresponding 16 bits. The experimental results are also in accordance with the theoretical deduction.

6-Round Integral Distinguisher Let the division property of the input multiset be $\mathcal{D}^{1^{64}}_{\{\texttt{[ffff,ffff,ffff,fff0]}\}}$, *i.e.*, we traverse the first 60 bits of the input. We observe that the objective function is equal to 2 after six rounds of encryption, which indicates that all 64 bits satisfying zero-sum property after six rounds of encryption. The objective function is equal to 1 after seven rounds of encryption and the experimental results show that all 64 unit vectors occur under this setting. This fact shows that there does not exist any bit satisfying zero-sum property after seven rounds of encryption if we traverse the corresponding 60 bits.

We also try to propagate the input division property $\mathcal{D}^{1^{64}}_{\{[\texttt{ffff},\texttt{ffff},\texttt{ffff},\texttt{fff7}]\}}$, but the objective function is equal to 1 after seven rounds of encryption and all 64 unit vectors occur. This observation indicates that we can not continue extending the length of the distinguisher even though we traverse more bits at the input.

4.4 Application to PHOTON

PHOTON [14] PHOTON is a family of hash function. In this paper we discuss the integral property of its internal permutations. The internal permutation of PHOTON is defined by P_t, where $t \in \{100, 144, 196, 256, 288\}$. The internal state of the N_r-round permutation is viewed as a $d \times d$ matrix of s-bit cells. The parameters of the internal permutations P_t are listed in **Table 4**.

Table 4: The Parameters of the Internal Permutations P_t.

Permutation	t	d	s	N_r	**Irreducible Polynomial**
P_{100}	100	5	4	12	x^4+x+1
P_{144}	144	6	4	12	x^4+x+1
P_{196}	196	7	4	12	x^4+x+1
P_{256}	256	8	4	12	x^4+x+1
P_{288}	288	6	8	12	$x^8+x^4+x^3+x+1$

Irreducible Polynomial: The irreducible polynomials used to define the base field.

P_t's are also AES-like permutation. One round has the following four layers applied to the internal state in the order stated below:

- `AddConstant`(AC): `XOR` two $(d \times s)$-bit constants to the first column of the internal state.
- `SubCell`(SC): Apply an s-bit S-box to each of the cells of the internal state. In the case of 4-bit cells, we use PRESENT's S-box while for the 8-bit cells we use AES's S-box.
- `ShiftRows`(SR): Rotate all cells to the left by i column positions for each row i.
- `MixColumnsSerial`(MCS): Multiply the internal state by a $d \times d$ MDS matrix M_{P_t}. The irreducible polynomials used to define the base field $\mathbb{F}_2^s$ are given in **Table 4**.

Since XOR with constants does not influence the propagation of division property, we do not introduce the constants used in the permutations. For space limitation, the MDS matrices used in different variants are also left behind. For more information, please refer to [14].

Former Results of PHOTON In the design paper of PHOTON [14], the authors applied the most recent developed cube testers [2] and its zero-sum distinguishers to the PHOTON permutations, the best they could find within practical time complexity is at most 3 rounds for all PHOTON variants. And the better bounds on the algebraic degree were recently published [7] and the results for PHOTON permutations were summarized in [14]. These bounds are listed in **Table 5**. For more information, please refer to [14].

Table 5: Bounds on the Algebraic degree of the PHOTON Internal Permutation.

Rounds	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
P_{100}	3	9	27	75	91	97	99	99	91
P_{144}	3	9	27	81	123	137	141	143	143
P_{196}	3	9	27	81	157	183	191	194	195
P_{256}	3	9	27	81	197	236	249	253	255
P_{288}	7	42	252	282	287	287	287	287	287

Applying MILP-Aided Bit-Based Division Property to PHOTON Since PHOTON permutations use the S-boxes of PRESENT and AES, and the propagation tables and the linear inequality systems for these two S-boxes are already given in [28] and **Section 4.1** of these paper, there details are not covered again here. In this paper, we analysis the integral property of the first three PHOTON permutations. For P_{100}, there are 194 non-zero elements in the primitive representation of its MDS matrix and $194 \times 5 = 970$ intermediate variables are required for one round of encryption. For P_{144}, the number of non-zero elements in the primitive representation of the corresponding MDS matrix is 269 and $269 \times 6 = 1614$ intermediate variables are needed for one round of encryption. As to P_{96}, there are 406 non-zero elements in the primitive representation of its corresponding MDS matrix and $406 \times 7 = 2842$ intermediate variables are required for one round of encryption.

For space limitation, we only state the resulting integral distinguishers in the following.

5-Round Integral Distinguisher for P_{100} Let the division property of the input multi-set be $\mathcal{D}^{1^{100}}_{\{[\texttt{f0000,0f000,00f00,000f0,0000f}]\}}$, *i.e.*, we traverse the 20 bits located at the diagonal of the internal state. We find that the objective function is equal to 2 after five rounds of encryption. The objective function is equal to 1 after six rounds of encryption and the experimental results show that all 100 unit vectors occur under this setting. This fact indicates that there does not exist any bit satisfying zero-sum property after six rounds of encryption if we only traverse the corresponding 20 bits.

From **Table 5**, the bound on the algebraic degree for five rounds P_{100} is equal to 91. In other word, this bound indicates that all the 100 bits satisfy zero-sum property after five rounds of encryption if we traverse 92 bits of the input state. Note that our newly found integral distinguisher is much better than the results deduced by the bounds on the algebraic degree.

6-Round Integral Distinguisher for P_{100} Let the division property of the input multi-set be $\mathcal{D}^{1^{100}}_{\{[\texttt{00fff,fffff,fffff,fffff,fffff}]\}}$, *i.e.*, we traverse the last 92 bits. We find that the objective function is equal to 2 after six rounds of encryption. The objective function is equal to 1 after seven rounds of encryption and the experimental results show that all 100 unit vectors exist under this setting. This fact indicates that there does not exist any bit satisfying zero-sum property after seven rounds of encryption if we only traverse the corresponding 92 bits.

From **Table 5**, we known that the bound on the algebraic degree for six rounds P_{100} is equal to 97, which indicates that all the 100 bits satisfy zero-sum property after six rounds of encryption if we traverse 98 bits of the input state. Note that our newly found 6-round integral distinguisher is better than the results deduced by the bounds on the algebraic degree.

5-Round Integral Distinguisher for P_{144} Let the division property of the input multi-set be $\mathcal{D}^{1^{144}}_{\{[\texttt{f00000,0f0000,00f000,000f00,0000f0,00000f}]\}}$, *i.e.*, we traverse the 24 bits located at the diagonal of the internal state. We find that the objective function is equal to 2 after five rounds of encryption. The objective function is equal to 1 after six rounds of encryption, and the experimental results also show that all 144 unit vectors occur under this setting. This fact indicates that there does not exist any bit satisfying zero-sum property after six rounds of encryption if we only traverse the corresponding 24 bits.

Note **Table 5** shows that the bounds on the algebraic degree for five rounds P_{144} is equal to 123, which indicates that that all the 144 bits satisfy zero-sum property after five rounds of encryption if we traverse 124 bits of the input state. Again, our newly found 5-round integral distinguisher is much better than the results deduced by the bounds on the algebraic degree.

5-Round Integral Distinguisher for P_{196} Let the division property of the input multi-set be $\mathcal{D}^{1^{196}}_{\{[\texttt{f000000,0f00000,00f0000,000f000,0000f00,00000f0,000000f}]\}}$, *i.e.*, we traverse the 28 bits located at the diagonal of the internal state. We find that the objective function is equal to 2 after five rounds of encryption. The objective function is equal to 1 after six rounds of encryption, and the experimental results also show that all 196 unit vectors occur under this setting. This fact indicates that there does not exist any bit satisfying zero-sum property after six rounds of encryption if we only traverse the corresponding 28 bits.

From **Table 5**, we know that the bounds on the algebraic degree for five rounds P_{196} is equal to 157, which tells that all the 196 bits satisfy zero-sum property after five rounds of encryption if we traverse 158 bits of the input state. Also note that our newly found 5-round integral distinguisher is much better that the results deduced by the bounds on the algebraic degree.

Introspection The AES-like design principle is convenient for designers and this kind of design principle allows designers to derive very simple bounds on the number of active S-boxes during the encryption. Profiting from the structural similarity, cryptographers are able to drive simple yet interesting AES-like security proofs for these AES-like block ciphers regarding related- or single-key attacks. But in this paper, we find that the lengths of the integral distinguishers for AES, `Joltik-BC`, and LED are different, which indicates that directly copying AES-like security proofs for some kinds of attacks seems to be less reasonable and the security of these AES-like design may need to be reconsidered.

4.5 Application to Serpent

Serpent [1] Serpent is a block cipher which was selected to be among the five finalists for Advanced Encryption Standard [19]. Serpent is a 32-round SPN structure operating on four 32-bit words (X_0, X_1, X_2, and X_3), thus giving a block size of 128 bits. Its round function consists of alternating layers of key mixing, S-boxes, and linear transformation. Serpent has eight S-boxes ($S_0 \sim S_7$) 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 again in the ninth round. In the linear transformation, $X_0 \sim X_3$ are linearly mixed by

$$\begin{aligned}
&X_0 \coloneqq X_0 \lll 13\\
&X_2 \coloneqq X_2 \lll 3\\
&X_1 \coloneqq X_1 \oplus X_0 \oplus X_2\\
&X_3 \coloneqq X_3 \oplus X_2 \oplus (X_0 \ll 3)\\
&X_1 \coloneqq X_1 \lll 1\\
&X_3 \coloneqq X_3 \lll 7\\
&X_0 \coloneqq X_0 \oplus X_1 \oplus X_3\\
&X_2 \coloneqq X_2 \oplus X_3 \oplus (X_1 \ll 7)\\
&X_0 \coloneqq X_0 \lll 5\\
&X_2 \coloneqq X_2 \lll 22
\end{aligned}$$

where '$\lll$' denotes rotation, and '$\ll$' denotes shift.

Since XORing with a constant does not influence the division property of a multi-set, we do not consider key mixing in our analysis. Also the key schedule is left behind. For more informations about Serpent, please refer to [1].

Former Results of Serpent At FSE 2008, Z'aba [30] *et al.* proposed bit-pattern based integral attack which was a generic method of finding integral distinguishers for bit-oriented block ciphers. They successfully utilized this method to find 3.5-round integral distinguisher for Serpent.

Applying MILP-Aided Bit-Based Division Property to Serpent For space limitation, we do not give the propagation tables and the linear inequality systems for the eight S-boxes of Serpent. For the linear layer, we treat it as a large 128×128 matrix and there are 610 non-zero elements in the primitive representation of Serpent's linear layer. Thus 610 intermediate variables ($t_0 \sim t_{609}$) are needed for one round of encryption. The naming rule of variables for one round of encryption is illustrated in **Fig. 4**. When across the linear layer, the first line of the variables represents X_0 and the last line of the variables represents X_3.

$$\begin{bmatrix} a_{124} & a_{120} & a_{116} & \cdots & a_4 & a_0 \\ a_{125} & a_{121} & a_{117} & \cdots & a_5 & a_1 \\ a_{126} & a_{122} & a_{118} & \cdots & a_6 & a_2 \\ a_{127} & a_{123} & a_{119} & \cdots & a_7 & a_3 \end{bmatrix} \xrightarrow{\text{S-box}} \begin{bmatrix} b_{124} & b_{120} & b_{116} & \cdots & b_4 & b_0 \\ b_{125} & b_{121} & b_{117} & \cdots & b_5 & b_1 \\ b_{126} & b_{122} & b_{118} & \cdots & b_6 & b_2 \\ b_{127} & b_{123} & b_{119} & \cdots & b_7 & b_3 \end{bmatrix} \xrightarrow[t_0 \sim t_{609}]{\text{Linear}} \begin{bmatrix} a_{252} & a_{248} & a_{244} & \cdots & a_{132} & a_{128} \\ a_{253} & a_{249} & a_{245} & \cdots & a_{133} & a_{129} \\ a_{254} & a_{250} & a_{246} & \cdots & a_{134} & a_{130} \\ a_{255} & a_{251} & a_{247} & \cdots & a_{135} & a_{131} \end{bmatrix}$$

Fig. 4: Variables for One Round of Serpent.

7-Round Integral Distinguisher Let the division property of the input multi-set be $\mathcal{D}^{1^{128}}_{\{[\texttt{0fffffff,ffffffff,ffffffff,ffffffff}]\}}$, *i.e.*, we traverse the last 124 bits. Since different rounds use different S-boxes, the starting round may influence the length of the resulting integral distinguisher. But after trying all possible cases of starting round, we find that the objective functions are all equal to 2 after seven rounds of encryption, which indicates that all the 128 bits satisfy zero-sum property after seven rounds of encryption. And the objective functions are all equal to 1 after eight rounds of encryption and the experimental results show that all the 128 unit vectors occur under this setting. This fact tells that there does not exist any bit satisfying zero-sum property after eight rounds of encryption if we only traverse the last 124 bits of the input multi-set.

We also try to propagate $\mathcal{D}^{1^{128}}_{\{[\texttt{ffffffff,ffffffff,ffffffff,fffffffe}]\}}$, but the objective function is also equal to 1 after eight rounds of encryption and all the 128 unit vectors occur. This observation indicates that we can not continue extending the length of the distinguisher even though we traverse more bits at the input.

4.6 Application to Noekeon

Noekeon [9] Noekeon accepts a 128-bit plaintext, and a 128-bit master key K. It runs in 16 rounds and each round consists of a linear function called L_0, three rotations called L_1, S-box layer, and three rotations L_1^{-1}. The S-box layer of Noekeon uses a single S-box 32 times in parallel, which is given in **Table 6**. Let $P = X^0 = (X_0^0, X_1^0, X_2^0, X_3^0)$ denote the 128-bit plaintext block formed by the concatenation of four 32-bit wrods X_i^0. Denote the input block of the r-th round by $X^r = (X_0^r, X_1^r, X_2^r, X_3^r)$. The round function of Noekeon is illustrated in **Fig. 5**. When across the S-box layer, every word contributes one bit to the input of the S-box. Denote the input block for the S-box layer of the r-th round by $Y^r = (Y_0^r, Y_1^r, Y_2^r, Y_3^r)$. Then the 4-bit nibbles $(Y_0^r[i], Y_1^r[i], Y_2^r[i], Y_3^r[i])$, $0 \leqslant i \leqslant 31$, constitute the inputs of the S-boxes.

Table 6: Noekeon's S-Box S_{Noekeon} [9]

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S[x]$	7	a	2	c	4	8	f	0	5	9	1	e	3	d	b	6

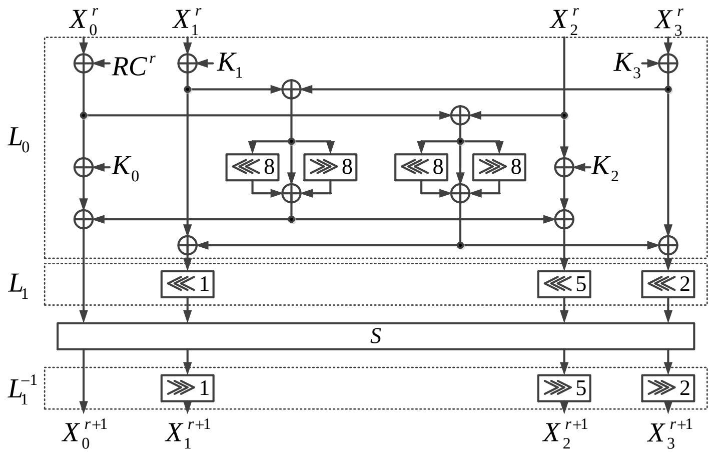

Fig. 5: Round Function of Noekeon.

Former Results of Noekeon At FSE 2008, Z'aba [30] *et al.* proposed bit-pattern based integral attack which was a generic method of finding integral distinguishers for bit-oriented block ciphers. They successfully utilized this method to find 3.5-round integral distinguisher for Noekeon.

Applying MILP-Aided Bit-Based Division Property to Noekeon The propagation table for Noekeon's S-box has 46 vectors. 147 linear inequalities are returned by using `inequality_generator()`. After using **Greedy Algorithm**, we get 12 linear inequalities. These linear inequalities are given in **Appendix B**. For the linear layer, we combine L_0 and L_1 and treat it as a large 128×128 matrix. There are 896 non-zero elements in the primitive representation of $L_1 \circ L_0$. Thus 896 intermediate variables ($t_0 \sim t_{895}$) are required for one round of encryption. Since L_1^{-1} is composed of three rotations, we do not introduce new intermediate variable and all we need to do is rotating the variables as needed. The naming rule for one round of encryption is illustrated in **Fig. 6**.

$$\left[a_0|a_1|a_2|\cdots|a_{126}|a_{127}\right] \xrightarrow[t_0\sim t_{895}]{L_1 \circ L_0} \left[b_0|b_1|b_2|\cdots|b_{126}|b_{127}\right] \xrightarrow{\text{Rearrange}} \left[\begin{array}{c|c|c|c|c|c} b_0 & b_1 & b_2 & \cdots & b_{30} & b_{31} \\ \hline b_{32} & b_{33} & b_{34} & \cdots & b_{62} & b_{63} \\ \hline b_{64} & b_{65} & b_{66} & \cdots & b_{94} & b_{95} \\ \hline b_{96} & b_{97} & b_{98} & \cdots & b_{136} & b_{127} \end{array}\right]$$

$$\xrightarrow{\text{S-box}} \left[\begin{array}{c|c|c|c|c|c} a_{128} & a_{129} & a_{130} & \cdots & a_{158} & a_{159} \\ \hline a_{161} & a_{162} & a_{163} & \cdots & a_{191} & a_{160} \\ \hline a_{197} & a_{198} & a_{199} & \cdots & a_{195} & a_{196} \\ \hline a_{226} & a_{227} & a_{228} & \cdots & a_{224} & a_{225} \end{array}\right] \xrightarrow{L_1^{-1}} \left[\begin{array}{c|c|c|c|c|c} a_{128} & a_{129} & a_{130} & \cdots & a_{158} & a_{159} \\ \hline a_{160} & a_{161} & a_{162} & \cdots & a_{190} & a_{191} \\ \hline a_{192} & a_{193} & a_{194} & \cdots & a_{222} & a_{223} \\ \hline a_{224} & a_{225} & a_{226} & \cdots & a_{254} & a_{255} \end{array}\right]$$

$$\xrightarrow{\text{Rearrange}} \left[a_{128}|a_{129}|a_{130}|\cdots|a_{254}|a_{255}\right]$$

Fig. 6: Variables for One Round of Noekeon.

6-Round Integral Distinguisher Let the division property of the input multi-set be $\mathcal{D}^{1^{128}}_{\{[\texttt{0fffffff,ffffffff,ffffffff,ffffffff}]\}}$, *i.e.*, we traverse the last 124 bits. We find that the objective function is equal to 2 after six rounds of encryption, which indicates that all the 128 bits satisfy zero-sum property after six rounds of encryption. The objective function is equal to 1 after seven rounds of encryption and the experimental results show that all the 128 unit vectors occur under this setting. This fact tells that there does not exist any bit satisfying zero-sum property after seven rounds of encryption if we traverse the 124 corresponding bits.

7-Round Integral Distinguisher Let the division property of the input multi-set be $\mathcal{D}^{1^{128}}_{\{[\texttt{7fffffff,ffffffff,ffffffff,ffffffff}]\}}$, *i.e.*, we traverse the last 127 bits. We find that the objective function is equal to 2 after seven rounds of encryption, which indicates that all the 128 bits satisfy zero-sum property after seven rounds of encryption. The objective function is equal to 1 after eight rounds of encryption and the experimental results show that all the 128 unit vectors occur under this setting. This fact tells that there does not exist any bit satisfying zero-sum property after eight rounds of encryption even though we traverse 127 bit of the input multi-set.

4.7 Application to SM4

SM4 [11] SM4 (formerly SMS4) is a block cipher used in the Chinese National Standard for Wireless LAN WAPI (Wired Authentication and Privacy Infrastructure). SM4 is a block cipher with 128-bit block size and 128-bit key size. It is a kind of unbalanced Feistel network. Let (X_0, X_1, X_2, X_3) and $(Y_0, Y_1, Y_2, Y_3) \in (\mathbb{F}_2^{32})^4$ denote the 128-bit plaintext P and the 128-bit ciphertext C, respectively. The i-th round of SM4 can be expressed as:

$$(X_i, X_{i+1}, X_{i+2}, X_{i+3}) \to (X_{i+1}, X_{i+2}, X_{i+3}, X_{i+4}),$$

where $X_{i+4} = X_i \oplus (L \circ S)(X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \oplus RK_i)$, and RK_i is the subkey of the i-th round. The round function of SM4 is illustrated in **Fig. 7**. The function S uses a single 8-bit S-box 4 times in parallel. The linear function L, which operates on 32-bit word, is given by

$$L(x) = x \oplus (x \lll 2) \oplus (x \lll 10) \oplus (x \lll 18) \oplus (x \lll 24).$$

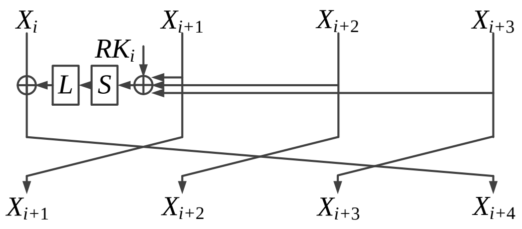

Fig. 7: The Round Function of SM4.

Applying MILP-Aided Bit-Based Division Property to SM4 The propagation table for SM4's S-box has 2009 vectors, and 46409 linear inequalities are returned by using `inequality_generator()`. After using **Greedy Algorithm**, we get 27 linear inequalities. These linear inequalities are given in **Appendix D**.

10-Round Integral Distinguisher Let the division property of the input multi-set be $\mathcal{D}^{1^{128}}_{\{[\texttt{ffffffff, ffffffff, fffffffc, 00000000}]\}}$, *i.e.*, we traverse the first 94 bits of the input. The values of the objective functions for different number of rounds are given in **Table 7**.

Combining with the results of **Table 7**, we know that there still are 32 bits satisfying zero-sum property after ten rounds of encryption if we traverse the first 94 bits of the input. But there does not exist any bit meeting zero-sum

Table 7: Zero-Sum Property for SM4 with Input Division Property $\mathcal{D}^{1^{128}}_{\{[\texttt{ffffffff, ffffffff, fffffffc, 00000000}]\}}$

Round	Values of Objective Function	#{Unit Vectors}	#{Zero-Sum Bits}
7	2	0	128
8	1	32	96
9	1	64	64
10	1	96	32
11	1	128	0

property after eleven rounds of encryption. The 10-round integral distinguisher can be expressed as:

$$\begin{aligned} &(\mathcal{A}^8\mathcal{A}^8\mathcal{A}^8\mathcal{A}^8, \mathcal{A}^8\mathcal{A}^8\mathcal{A}^8\mathcal{A}^8, \mathcal{A}^8\mathcal{A}^8\mathcal{A}^8\mathcal{A}^6, \mathcal{C}^8\mathcal{C}^8\mathcal{C}^8\mathcal{C}^8) \\ \xrightarrow{\text{10 Rounds}} &(\mathcal{B}^8\mathcal{B}^8\mathcal{B}^8\mathcal{B}^8, \mathcal{U}^8\mathcal{U}^8\mathcal{U}^8\mathcal{U}^8, \mathcal{U}^8\mathcal{U}^8\mathcal{U}^8\mathcal{U}^8, \mathcal{U}^8\mathcal{U}^8\mathcal{U}^8\mathcal{U}^8), \end{aligned}$$

where '$\mathcal{A}^8$' represents an active byte, '$\mathcal{A}^6$' represents a byte whose first six bits and last two bits are respectively active and constant, '$\mathcal{C}^8$' represents a constant byte, '$\mathcal{B}^8$' represents a byte with every bit satisfying zero-sum property, and '$\mathcal{U}^8$' represents an unknown byte.

11-Round Integral Distinguisher Let the division property of the input multi-set be $\mathcal{D}^{1^{128}}_{\{[\texttt{ffffffff, ffffffff, ffffffff, ff000000}]\}}$, *i.e.*, we traverse the first 104 bits of the input. The values of the objective functions for different number of rounds are listed in **Table 8**.

Table 8: Zero-Sum Property for SM4 with Input Division Property $\mathcal{D}^{1^{128}}_{\{[\texttt{ffffffff, ffffffff, ffffffff, ff000000}]\}}$

Round	Values of Objective Function	#{Unit Vectors}	#{Zero-Sum Bits}
8	2	0	128
9	1	32	96
10	1	64	64
11	1	96	32
12	1	128	0

Combining with the results of **Table 8**, we know that there still are 32 bits satisfying zero-sum property after eleven rounds of encryption if we traverse the first 104 bits of the input. But there does not exist any bit meeting zero-sum property after twelve rounds of encryption. The 11-round integral distinguisher can be expressed as:

$$(\mathcal{A}^8\mathcal{A}^8\mathcal{A}^8\mathcal{A}^8, \mathcal{A}^8\mathcal{A}^8\mathcal{A}^8\mathcal{A}^8, \mathcal{A}^8\mathcal{A}^8\mathcal{A}^8\mathcal{A}^8, \mathcal{A}^8\mathcal{C}^8\mathcal{C}^8\mathcal{C}^8)$$
$$\xrightarrow{\text{11 Rounds}} (\mathcal{B}^8\mathcal{B}^8\mathcal{B}^8\mathcal{B}^8, \mathcal{U}^8\mathcal{U}^8\mathcal{U}^8\mathcal{U}^8, \mathcal{U}^8\mathcal{U}^8\mathcal{U}^8\mathcal{U}^8, \mathcal{U}^8\mathcal{U}^8\mathcal{U}^8\mathcal{U}^8),$$

where '$\mathcal{A}^8$' represents an active byte, '$\mathcal{C}^8$' represents a constant byte, '$\mathcal{B}^8$' represents a byte with every bit satisfying zero-sum property, and '$\mathcal{U}^8$' represents an unknown byte.

12-Round Integral Distinguisher Let the division property of the input multi-set be $\mathcal{D}^{1^{128}}_{\{[\texttt{ffffffff, ffffffff, ffffffff, fffffff8}]\}}$, *i.e.*, we traverse the first 125 bits of the input. The values of the objective functions for different number of rounds are listed in **Table 9**.

Table 9: Zero-Sum Property for SM4 with Input Division Property $\mathcal{D}^{1^{128}}_{[\texttt{ffffffff, ffffffff, ffffffff, fffffff8}]}$

Round	**Values of Objective Function**	**#{Unit Vectors}**	**#{Zero-Sum Bits}**
9	2	0	128
10	1	32	96
11	1	64	64
12	1	96	32
13	1	128	0

Combining with the results of **Table 9**, we know that there still are 32 bits satisfying zero-sum property after twelve rounds of encryption if we traverse the first 125 bits of the input. But there does not exist any bit meeting zero-sum property after thirteen rounds of encryption. The 12-round integral distinguisher can be expressed as:

$$(\mathcal{A}^8\mathcal{A}^8\mathcal{A}^8\mathcal{A}^8, \mathcal{A}^8\mathcal{A}^8\mathcal{A}^8\mathcal{A}^8, \mathcal{A}^8\mathcal{A}^8\mathcal{A}^8\mathcal{A}^8, \mathcal{A}^8\mathcal{A}^8\mathcal{A}^8\mathcal{A}^5)$$
$$\xrightarrow{\text{12 Rounds}} (\mathcal{B}^8\mathcal{B}^8\mathcal{B}^8\mathcal{B}^8, \mathcal{U}^8\mathcal{U}^8\mathcal{U}^8\mathcal{U}^8, \mathcal{U}^8\mathcal{U}^8\mathcal{U}^8\mathcal{U}^8, \mathcal{U}^8\mathcal{U}^8\mathcal{U}^8\mathcal{U}^8),$$

where '$\mathcal{A}^8$' represents an active byte, '$\mathcal{A}^5$' represents a byte whose first five bits and last three bits are respectively active and constant, '$\mathcal{B}^8$' represents a byte with every bit satisfying zero-sum property, and '$\mathcal{U}^8$' represents an unknown byte.

4.8 Application to SPONGENT-88

SPONGENT [4] SPONGENT is a family of lightweight hash functions with different hash sizes. In this paper, we only analyze the variant with hash size 88 which is denoted by SPONGENT-88. SPONGENT-88 also uses SP-network and utilizes a PRESENT-type permutation which iterates 45 times. The non-linear layer uses a 4-bit S-box which is given in **Table 10**. An illustration of the round function is given in **Fig. 8**. For more details about SPONGENT, please refer to [4].

Table 10: SPONGENT's S-Box S_{SPONGENT} [4]

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S[x]$	e	d	b	0	2	1	4	f	7	a	8	5	9	c	3	6

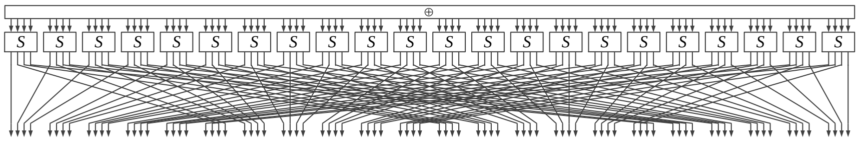

Fig. 8: Round Function of SPONGENT-88.

Former Results of SPONGENT-88 In 2012, Dong *et al.* [12] combined the higher-order differential attack and integral attack to construct a zero-sum distinguisher for 14-round SPONGENT-88 in computational complexity 2^{84}. Firstly, they found two short zero-sum distinguishers going to opposite directions by tracking the algebraic degree round by round. Then they tried to combine these two distinguishers by expanding them under a considerable data requirement. In 2015, Fan and Duan [13] utilized the same idea to construct an improved zero-sum distinguisher for 14-round SPONGENT-88 in computational complexity 2^{80}, which is 2^4 less than the one in [12]. By integrating table-aided bit-based division property and higher-order differential attack, Sun and Wang [22] provided a 14-round higher-order integral distinguisher with complexity 2^{80}.

Applying MILP-Aided Bit-Based Division Property to SPONGENT-88 Since we need to find zero-sum distinguishers in opposite directions, the propagation table for S_{SPONGENT} and S^{-1}_{SPONGENT} are required. When we need to find the zero-sum distinguishers in the forward direction, we use the S_{SPONGENT}'s propagation table. If we turn to search the zero-sum distinguishers in the backward direction, we apply the propagation table of S^{-1}_{SPONGENT}.

There are 48 vectors in the propagation table of S_{SPONGENT}. By using the `inequality_generator()` in the Sage software, 197 inequalities are returned.

After utilizing **Greedy Algorithm**, we get 10 inequalities. For the propagation table of S^{-1}_{SPONGENT}, 48 elements return 180 inequalities. After utilizing **Greedy Algorithm**, 11 inequalities are gotten. The linear inequality systems for S_{SPONGENT} and S^{-1}_{SPONGENT} are given in **Appendix C**. The naming rule of variables for SPONGENT-88 is simply $a_0 \sim a_{87}$, from left to right. Since the linear layer of SPONGENT-88 is just a permutation, $88 \times 2 = 176$ variables (including the 88 variables representing the division property for the input bits of the next round) are enough to constitute the linear inequality system for one round.

15-Round Zero-Sum Distinguisher Let the division property for the input multi-set of the seventh round be $\mathcal{D}^{1^{88}}_{\{[\texttt{ffffffffff,ffffffffff00}]\}}$, *i.e.*, we traverse the first 80 bits.

- In the forward direction, we find that the objective function is equal to 1 after eight rounds of encryption. But there are only 59 unit vectors. The absence of the other 29 unit vectors indicates that there are 29 bits satisfying zero-sum property after eight rounds of encryption. Those 29 bits satisfying zero-sum property are labeled as 10, 18 $\sim$ 21, 38, 41 $\sim$ 43, 60, 63 $\sim$ 65, 68, 70, 72 $\sim$ 73, 75 $\sim$ 76, and 78 $\sim$ 87.
- In the backward direction, we find that the objective function is equal to 1 after seven rounds of decryption. But there are only 68 unit vectors. The absence of the other 20 unit vectors indicates that there are 20 bits satisfying zero-sum property after seven rounds of decryption. Those 20 bits satisfying zero-sum property are labeled as 0, 8, 16, 20, 24, 28, 32, 36, 40, 44, 48, 56, 60, 64, 68, 72, 80 $\sim$ 82, and 84.

Combining these short integral distinguishers in different directions, we get a 15-round higher-order integral distinguisher for SPONGENT-88 with complexity 2^{80}. Comparing with the former results, our newly found distinguisher achieves one more round than the one proposed in [13] while keeping the data complexity.

16-Round Zero-Sum Distinguisher Let the division property for the input multi-set of the eighth round be $\mathcal{D}^{1^{88}}_{\{[\texttt{ffffffffff,fffffffff0}]\}}$, *i.e.*, we traverse the first 84 bits.

- In the forward direction, we find that the objection function is equal to 1 after eight rounds of encryption. But there are only 55 unit vectors. The absence of the other 33 unit vectors tells that there are 33 bits satisfying zero-sum property after eight rounds of encryption. Those 33 bits satisfying zero-sum property are labeled as 10, 18 $\sim$ 21, 38 $\sim$ 39, 41 $\sim$ 43, 60 $\sim$ 61, 63 $\sim$ 65, 68 $\sim$ 73, 75 $\sim$ 76, and 78 $\sim$ 87.
- In the backward direction, we observe that the objective function is equal to 1 after eight rounds of decryption. But there are only 82 unit vectors. The absence of the other 6 unit vectors indicates that there are 6 bits satisfying zero-sum property after eight rounds of decryption. Those 6 bits satisfying zero-sum property are labeled as 0, 16, 24, 40, 56, and 72.

Combining the above two short integral distinguishers in different directions, we get a 16-round higher-order integral distinguisher for SPONGENT-88 with complexity 2^{84}. Note that this newly found distinguisher achieves two more rounds than the one given in [13].

5 Conclusion

In this paper, we settle the feasibility of MILP method applying to ciphers with linear layers which are not bit-permutations by introducing some intermediate variables among the linear layer and make the MILP-aided bit-based division to be more helpful. MILP-aided bit-based division property is also applied to find integral distinguishers for AES, LED, `Joltik-BC`, PHOTON, Serpent, Noekeon, SM4, and SPONGENT-88. By no means do we judge that there are no longer integral distinguishers for these primitives. We only say that we can not find longer integral distinguishers based on bit-based division property.

References

1. R. Anderson, E. Biham, and L. Knudsen. *Serpent: A proposal for the advanced encryption standard*, volume 174, pages 1–23. 1998.
2. J. P. Aumasson, I. Dinur, W. Meier, and A. Shamir. *Cube Testers and Key Recovery Attacks on Reduced-Round MD6 and Trivium*, pages 1–22. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
3. R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers. The simon and speck lightweight block ciphers. In *Proceedings of the 52nd Annual Design Automation Conference*, page 175. ACM, 2015.
4. A. Bogdanov, M. Knezevic, G. Leander, D. Toz, K. Varici, and I. Verbauwhede. Spongent: The design space of lightweight cryptographic hashing. *IEEE Transactions on Computers*, 62(10):2041–2053, 2013.
5. A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, and C. Vikkelsoe. *PRESENT: An Ultra-Lightweight Block Cipher*, pages 450–466. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.
6. C. Boura and A. Canteaut. Another view of the division property. Cryptology ePrint Archive, Report 2016/554, 2016. http://eprint.iacr.org/2016/554.
7. C. Boura, A. Canteaut, and C. De Cannière. *Higher-Order Differential Properties of Keccak and Luffa*, pages 252–269. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
8. J. Daemen, L. Knudsen, and V. Rijmen. *The block cipher Square*, pages 149–165. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997.
9. J. Daemen, M. Peeters, G. Van Assche, and V. Rijmen. Nessie proposal: Noekeon. In *First Open NESSIE Workshop*, pages 213–230, 2000.
10. J. Daemen and V. Rijmen. Aes proposal: Rijndael. 1999.
11. W. Diffie and G. Ledin. Sms4 encryption algorithm for wireless networks. *IACR Cryptology ePrint Archive*, 2008:329, 2008.
12. L. Dong, W.L. Wu, S. Wu, and J. Zou. Another look at the integral attack by the higher-order differential attack. *Jisuanji Xuebao(Chinese Journal of Computers)*, 35(9):1906–1917, 2012.

13. S. Fan and M. Duan. Improved zero-sum distinguisher for spongent-88. 2015.
14. J. Guo, T. Peyrin, and A. Poschmann. *The PHOTON Family of Lightweight Hash Functions*, pages 222–239. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
15. J. Guo, T. Peyrin, A. Poschmann, and M. Robshaw. *The LED Block Cipher*, pages 326–341. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
16. J. Jean, I. Nikolić, and T. Peyrin. Joltik v1. 3. *CAESAR Round*, 2, 2015.
17. F. Liu, W. Ji, L. Hu, J. Ding, S. Lv, A. Pyshkin, and R. P. Weinmann. *Analysis of the SMS4 Block Cipher*, pages 158–170. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.
18. M. Matsui. *New block encryption algorithm MISTY*, pages 54–68. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997.
19. A. NIST. Request for candidate algorithm nominations for the aes. *Available on-line at http://www. nist. gov/aes.*
20. NIST FIPS Pub. 197: Advanced encryption standard (aes). *Federal Information Processing Standards Publication*, 197:441–0311, 2001.
21. K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai. *Piccolo: An Ultra-Lightweight Blockcipher*, pages 342–357. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
22. L. Sun and M. Wang. Towards a further understanding of bit-based division property. Cryptology ePrint Archive, Report 2016/392, 2016. http://eprint.iacr.org/2016/392.
23. T. Suzaki, K. Minematsu, S. Morioka, and E. Kobayashi. *TWINE: A Lightweight Block Cipher for Multiple Platforms*, pages 339–354. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
24. Y. Todo. Integral cryptanalysis on full misty1. In *Annual Cryptology Conference*, pages 413–432. Springer, 2015.
25. Y. Todo. *Structural Evaluation by Generalized Integral Property*, pages 287–314. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.
26. Y. Todo and M. Morii. Bit-based division property and application to simon family. *Pre-Proceedings of FSE*, 2016.
27. W. Wu and L. Zhang. *LBlock: A Lightweight Block Cipher*, pages 327–344. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
28. Z. Xiang, W. Zhang, Z. Bao, and D. Lin. Applying milp method to search integral distinguishers based on division property for 6 lightweight block ciphers. To appear at ASIACRYPT 2016.
29. G. Yang, B. Zhu, V. Suder, M. D. Aagaard, and G. Gong. *The Simeck Family of Lightweight Block Ciphers*, pages 307–329. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.
30. M.R. Z'aba, H. Raddum, M. Henricksen, and E. Dawson. *Bit-Pattern Based Integral Attack*, pages 363–381. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
31. W. Zhang, Z. Bao, D. Lin, V. Rijmen, B. Yang, and I. Verbauwhede. Rectangle: a bit-slice lightweight block cipher suitable for multiple platforms. *Science China Information Sciences*, 58(12):1–15, 2015.

A Linear Inequalities of AES's S-Box

Denote $(x_0, x_1, \ldots, x_7) \xrightarrow{S_{\text{AES}}} (y_0, y_1, \ldots, y_7)$ a division trail of AES's S-box. The 32 linear inequalities used to describe the propagation is given in **Table 11**. For simplicity, we only list the coefficients of the linear inequalities. For example, the first line of **Table 11** refers to as:

$$x_0 + x_1 + x_2 + x_3 + 41x_4 + x_5 + x_6 + x_7 - 6y_0 - 7y_1 - 7y_2 - 6y_3 - \\ 7y_4 - 7y_5 - 7y_6 - 7y_7 \geqslant -6.$$

Table 11: Linear Inequalities Used to Describe S_{AES}.

Number	**Linear Inequalities Used to Describe $S_{\mathbf{AES}}$**
1	$(1, 1, 1, 1, 41, 1, 1, 1, -6, -7, -7, -6, -7, -7, -7, -7, -6)$
2	$(1, 1, 45, 1, 1, 1, 1, 1, -6, -6, -7, -8, -8, -8, -8, -8, -7)$
3	$(0, 0, 0, 0, 0, 11, 0, 0, -2, -2, -1, -1, -2, -2, -2, -1, -2)$
4	$(0, 0, 0, 14, 0, 0, 0, 0, -2, -1, -3, -1, -3, -3, -3, -1, -3)$
5	$(44, 1, 1, 2, 1, 1, 1, 1, -7, -1, -8, -9, -10, -10, -8, -8, -9)$
6	$(0, 0, 0, 0, 0, 0, 5, 0, -1, -1, 0, -1, -1, -1, 0, -1, -1)$
7	$(2, 49, 2, 2, 1, 1, 1, 1, -9, -9, -9, -11, -11, -9, -9, -2, -10)$
8	$(-4, -3, -5, -6, -8, -11, -11, -1, 38, 41, 43, 46, 45, 48, 44, 44, 0)$
9	$(-2, -2, -3, -3, -3, -1, 6, -1, -4, 22, -7, -4, -4, -4, -5, -4, -19)$
10	$(-8, -15, -15, 0, -7, -7, -6, -1, -8, -9, 7, 6, -2, -1, -1, 6, -61)$
11	$(0, 0, 2, -5, 1, 0, 1, 1, 22, -5, -5, -6, -5, -7, 0, -6, -12)$
12	$(-10, -3, -4, -10, 0, -2, -3, -10, -7, -2, -5, -8, 3, 14, -1, 1, -47)$
13	$(2, 3, 1, 4, 1, 1, 1, 16, -13, -14, -1, -16, 42, -18, -14, -12, -17)$
14	$(-1, -5, -4, -7, -8, -8, -2, -2, 4, -3, 3, 6, 4, 7, 3, 7, -32)$
15	$(-1, -3, -7, -4, -7, -2, -7, -5, 6, 6, 4, 6, 7, 6, 6, 1, -28)$
16	$(1, -1, 0, 0, 0, 4, 0, 1, -8, 3, -7, 17, -6, -3, -6, -4, -9)$
17	$(-3, 3, -2, -2, -2, -3, -2, -1, 1, 1, 2, -4, -4, 1, 2, -6, -19)$
18	$(-2, 0, -2, 3, -2, -2, 0, -5, 2, -5, 2, 5, -3, -1, -3, -3, -16)$
19	$(-4, -7, -5, -4, -1, -8, -8, -3, 2, 9, 9, 5, 6, 7, 9, 9, -30)$
20	$(-2, -3, -1, -5, -5, 0, -4, -4, 1, 2, -4, 3, 2, 2, 3, 2, -23)$
21	$(2, 8, 6, 1, 1, 1, 1, 1, -18, -11, -6, -18, -12, -18, -10, 55, -17)$
22	$(-2, -2, 1, 3, -2, -2, -1, -1, -1, -5, -3, -3, -1, 2, 0, 4, -13)$
23	$(0, -3, 2, 0, 0, 0, 0, 1, -3, 1, -3, 11, -2, -4, -4, -3, -7)$
24	$(-1, 0, -4, -3, -5, -5, -5, -1, 2, 5, 6, 7, 6, 6, 5, 4, -17)$
25	$(1, 2, 1, -2, 2, 0, -2, -2, 19, -4, -5, -5, -4, -4, -4, -4, -11)$
26	$(-3, -4, -4, -2, -1, -1, -4, -2, 3, 3, 3, 0, 3, 3, 3, 3, -17)$
27	$(-3, -1, -4, -4, -4, -2, -1, 0, 4, 3, 1, 4, 4, 5, 3, 4, -14)$
28	$(-4, -5, -9, -9, -9, -9, 6, 0, -5, -15, 4, 3, -6, -1, 5, 3, -51)$
29	$(-1, -2, -2, 0, -2, -1, -2, -1, 2, 1, 1, 1, -2, -2, 1, -4, -13)$
30	$(-1, -2, 0, -1, -1, -1, -1, -2, 1, 1, -1, 0, -1, 0, 0, 0, -9)$
31	$(0, -1, -1, 0, 1, -1, -1, -1, -2, -1, 1, 1, -1, 1, -1, 0, -6)$
32	$(0, 1, -1, 1, -1, -1, -1, -1, -1, 1, 0, 0, 0, 0, 0, 0, -5)$

B Linear Inequalities of Noekeon's S-Box

Denote $(x_0, x_1, x_2, x_3) \xrightarrow{S_{\text{Noekeon}}} (y_0, y_1, y_2, y_3)$ a division trail of S_{Noekeon}. The linear inequality system (6) is sufficient to describe the propagation.

$$\begin{cases} x_0 + x_1 + x_2 + x_3 - y_0 - y_1 - y_2 - y_3 \geqslant 0 \\ -x_0 - x_1 - 2x_2 - 3x_3 + 2y_0 - y_1 + y_2 + y_3 \geqslant -4 \\ -2x_0 - x_1 - 3y_0 + 2y_1 - y_2 + y_3 \geqslant -4 \\ x_0 + x_1 + x_2 + x_3 + y_0 - 2y_1 - 2y_2 - 2y_3 \geqslant -1 \\ -x_0 - x_2 - x_3 + y_0 + y_1 + 2y_2 + 2y_3 \geqslant 0 \\ x_2 + x_3 - 2y_0 - y_1 + y_2 - 2y_3 \geqslant -2 \\ -2x_1 - x_2 + y_0 + y_1 - y_2 - 2y_3 \geqslant -4 \\ -x_0 - x_1 - x_3 - 3y_0 + 2y_1 - y_2 + y_3 \geqslant -4 \\ x_0 + x_3 - y_0 - y_1 - y_3 \geqslant -1 \\ -x_0 - x_1 - x_2 - 2x_3 + 2y_0 - 2y_1 + y_3 \geqslant -4 \\ x_0 + x_1 - y_0 - y_2 - y_3 \geqslant -1 \\ -x_1 - x_3 + y_0 + y_1 + 2y_2 + 2y_3 \geqslant 0 \\ x_i, y_i \text{ are binaries} \end{cases} \tag{6}$$

C Linear Inequalities of SPONGENT's S-Box

C.1 Linear Inequality System of S_{SPONGENT}

Denote $(x_0, x_1, x_2, x_3) \xrightarrow{S_{\text{SPONGENT}}} (y_0, y_1, y_2, y_3)$ a division trail of S_{SPONGENT}. The linear inequality system (7) is sufficient to describe the propagation.

$$\begin{cases} x_0 + x_1 + x_2 + x_3 - y_0 - y_1 - y_2 - y_3 \geqslant 0 \\ -x_1 - x_2 - 2x_3 + 2y_0 + y_1 + y_2 + y_3 \geqslant -1 \\ 3x_3 - y_0 - y_1 - y_2 - y_3 \geqslant -1 \\ -x_0 + x_1 - 3y_0 + 3y_1 - 2y_2 - 2y_3 \geqslant -4 \\ -x_0 + x_2 - 3y_0 - 2y_1 + 3y_2 - 2y_3 \geqslant -4 \\ -2x_0 - x_1 - x_2 - 2x_3 + 5y_0 + 4y_1 + 4y_2 + 2y_3 \geqslant 0 \\ -x_0 - y_0 - y_1 - y_2 + 2y_3 \geqslant -2 \\ -x_0 + x_2 + y_0 - 2y_1 - y_2 - y_3 \geqslant -3 \\ -x_1 - x_2 + y_1 + y_2 + y_3 \geqslant -1 \\ 3x_0 + x_1 + x_2 + x_3 - 3y_0 - 2y_1 - 2y_2 - y_3 \geqslant -2 \\ x_i, y_i \text{ are binaries} \end{cases} \tag{7}$$

C.2 Linear Inequality System of $S^{-1}_{\textbf{SPONGENT}}$

Denote $(x_0, x_1, x_2, x_3) \xrightarrow{S^{-1}_{\text{SPONGENT}}} (y_0, y_1, y_2, y_3)$ a division trail of S^{-1}_{SPONGENT}. The linear inequality system (8) is sufficient to describe the propagation.

$$\begin{cases} x_0 + x_1 + x_2 + x_3 - y_0 - y_1 - y_2 - y_3 \geqslant 0 \\ -5x_0 - 3x_1 - 3x_2 - 4x_3 - y_0 + 2y_1 + 2y_2 + 4y_3 \geqslant -8 \\ 3x_0 - y_0 - y_1 - y_2 - y_3 \geqslant -1 \\ -2x_0 - x_1 - x_3 + y_0 - 3y_1 + 2y_2 - y_3 \geqslant -5 \\ -2x_0 - x_2 - x_3 + y_0 + 2y_1 - 3y_2 - y_3 \geqslant -5 \\ -x_1 - x_2 + 2y_0 + 2y_1 + 2y_2 + y_3 \geqslant 0 \\ 3x_2 + x_3 - y_0 - y_1 - 2y_2 - 2y_3 \geqslant -2 \\ 3x_1 + x_3 - y_0 - 2y_1 - y_2 - 2y_3 \geqslant -2 \\ -2x_0 - x_1 - x_2 - x_3 + 2y_0 + 3y_1 + 3y_2 + 4y_3 \geqslant 0 \\ x_2 - y_0 - y_3 \geqslant -1 \\ -2x_0 - 2x_1 - x_2 + y_0 + y_2 + y_3 \geqslant -3 \\ x_i, y_i \text{ are binaries} \end{cases} \tag{8}$$

D Linear Inequalities of SM4's S-Box

Denote $(x_0, x_1, \ldots, x_7) \xrightarrow{S_{\text{SM4}}} (y_0, y_1, \ldots, y_7)$ a division trail of S_{SM4}. The 27 linear inequalities used to describe the propagation is given in **Table 12**. For simplicity, we only list the coefficients of the linear inequalities. For example, the first line of **Table 12** refers to as

$$x_0 + x_1 + x_2 + x_3 + x_4 + x_5 + 36x_6 + x_7 - 6y_0 - 6y_1 - 6y_2$$
$$-6y_3 - 6y_4 - 6y_5 - 6y_6 - 6y_7 \geqslant -5.$$

Table 12: Linear Inequalities Used to Describe S_{SM4}.

Number	**Linear Inequalities Used to Describe S_{SM4}**
1	$(1, 1, 1, 1, 1, 1, 36, 1, -6, -6, -6, -6, -6, -6, -6, -6, -5)$
2	$(0, 0, 0, 0, 0, 0, 0, 7, -1, -1, -1, -1, -1, -1, -1, -1, -1)$
3	$(40, 1, 1, 1, 1, 1, 1, 1, -6, -6, -7, -6, -7, -7, -7, -7, -6)$
4	$(1, 1, 40, 1, 1, 1, 1, 1, -7, -6, -7, -6, -7, -7, -6, -7, -6)$
5	$(0, 0, 0, 11, 0, 0, 0, 0, -2, -1, -2, -1, -2, -2, -2, -1, -2)$
6	$(1, 1, 1, 1, 1, 37, 1, 1, 1, -8, -8, -8, -8, -7, -6, -7, -7)$
7	$(-4, 5, -4, -4, 0, -4, -4, -4, -5, -1, -1, -1, -1, -1, -1, 5, -25)$
8	$(-4, 0, -4, -5, 0, -5, -4, -4, 4, -1, 0, 4, -1, -1, -1, -5, -27)$
9	$(-2, -4, -1, -2, -1, -5, -4, -5, 23, 23, 23, 19, 23, 22, 22, 23, 0)$
10	$(0, 2, -5, 0, 7, 1, 1, 0, -5, -12, -11, -11, -11, 42, -10, -5, -17)$
	Continued on next page

Table 12 – continued from previous page

Number	Linear Inequalities Used to Describe $S_{\mathbf{SM4}}$
11	$(-14,-13,-16,-16,-15,-12,-14,-16,-2,3,3,2,2,1,4,1,-102)$
12	$(0,-1,0,0,5,0,0,0,-1,-1,-1,0,-1,0,-1,-1,-2)$
13	$(0,3,1,0,-4,1,1,0,1,-8,27,-6,-9,-7,-6,-7,-13)$
14	$(0,-3,-5,0,-8,-4,-1,0,-1,-1,-3,-3,17,-5,-5,-4,-26)$
15	$(-3,-4,-1,-1,-4,-3,-2,-4,2,-4,-1,3,-1,-1,3,3,-22)$
16	$(-3,-1,-4,-6,-9,-10,-6,-10,4,7,5,-2,7,7,7,5,-41)$
17	$(-3,-3,-1,-3,0,-2,-2,-3,1,2,2,2,2,2,-1,1,-15)$
18	$(-2,-2,-2,-2,-1,0,-2,-2,4,-2,-2,0,-1,-3,1,2,-14)$
19	$(0,-4,-5,-5,-4,-1,-5,-5,-3,5,2,1,5,2,4,2,-27)$
20	$(-3,-5,-5,-5,-5,-5,-2,-4,2,2,3,2,1,4,3,4,-28)$
21	$(-2,-1,-2,-1,-2,0,-2,-2,0,2,2,1,1,2,2,-1,-11)$
22	$(0,0,0,0,0,1,0,0,-2,1,1,0,-1,-1,0,-1,-2)$
23	$(0,-2,-2,-1,-2,-1,-2,-2,2,-3,-1,2,-2,-1,1,1,-13)$
24	$(0,-1,-2,0,0,-1,-2,-1,1,1,-1,-2,1,2,-1,1,-8)$
25	$(1,-6,4,0,1,-6,0,0,-6,-5,-6,-3,22,-5,-3,-6,-18)$
26	$(-2,-1,-2,-1,0,-3,-2,-2,1,1,0,2,0,-1,-1,-2,-13)$
27	$(-2,-2,0,-1,-2,-1,-2,-2,0,-4,-2,2,-2,1,2,1,-14)$