Towards a Further Understanding of Bit-Based Division Property

Ling Sun[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; mqwang@sdu.edu.cn

Abstract. At EUROCRYPT 2015, Todo proposed the division property. Since then, many researches about the division property had occurred in succession. Inspired by the bit-based division property on SIMON introduced by Todo and Morri at FSE 2016, we give a further understanding of bit-based division property and come up with a new method to reconsider the **Substitution** rule given by Todo. By integrating the method of division property with the concrete boolean function expressions of S-box, this new idea can help us trace the propagation of division property at the bit level and escape the tedious and direct application of the original propagation rules. Benefit from this fact, this method can be applied to find integral distinguishers for some bit-oriented block ciphers other than SIMON. Since this method replaces the **Substitution** rules with a subtle propagation table, we call it table-aided bit-based division property. In order to verify our new method, we apply it to find integral distinguishers for CipherFour. The experimental results indicate that the table-aided bit-based division property is indeed a valid and efficient tool to search for integral distinguishers for some bit-oriented block ciphers. To handle the huge memory complexity of utilizing this new method, we apply early reduce technique, which was proposed by Zhang and Wu at INDOCRYPT 2015. With the help early reduce technique, a 8-round higher-order integral distinguisher for RECTANGLE can be constructed, which attains one more round than the previous one proposed by the designers. For PRESENT, we can find new 5-round and 6-round integral distinguishers. As to SPONGENT-88, a new 14-round zero-sum distinguisher with data complexity 2^{80} can be found by combining our new method with previous techniques. The table-aided bit-based division property can also be applied to find integral distinguishers for some word-oriented block ciphers, like TWINE and LBlock. Although we do not find any new integral distinguishers for these two ciphers, we believe that considering the S-box at the bit level is of great importance even for a word-oriented block cipher.

Keywords: Integral cryptanalysis, Table-aided bit-based division property, Early reduce, RECTANGLE, PRESENT, SPONGENT, TWINE, LBlock

* Corresponding Author

1 Introduction

The integral cryptanalysis was first introduced as a dedicate attack for the block cipher SQUARE [7]. After that, this method was also utilized to analyze the security of Rijndael [9], which was the predecessor of AES [10]. Knudsen and Wagner [14] formalized integral cryptanalysis and introdced an improtant extension called higher-order integral attack. For a fairly long time, integral cryptanalysis was well-known to operate on word-oriented block ciphers and was particularly applicable to block ciphers which use bijective components only. At FSE 2008, Z'aba [24] *et al.* proposed bit-pattern based integral attack which was a generic method of finding integral distinguisher for bit-oriented block ciphers. They successfully utilized this method to find 3.5-round integral distinguishers for Noekeon [8], Serpent [1] and PRESENT [4].

Note that the integral property does not clearly exploit the algebraic degree of block ciphers, the integral distinguishers consctructed under this property seems not to be useful. At EUROCRYPT 2015, Todo [18] proposed the division property, which is a generalization of integral property. It can effectively construct the integral distinguisher even if the block cipher has non-bijective functions, bit-oriented structures and low-degree functions. From the viewpoint of the attackable number of rounds or the number of chosen plaintexts, the division property sometimes can give better integral distinguishers than previous methods. To illustrate division property's utility on bit-oriented structures and non-bijective functions, Todo theoretically proved that SIMON32, 48, 64, 96, and 128 have 9-, 11-, 11-, 13-, and 13-round integral distinguisher, respectively.

Since then, many researches about the division property had occurred in succession. For example, Sun *et al.* [16] paid attention to the property of multiset $\mathbb{X}$ with division property $\mathcal{D}_k^n$ and proved that the number of elements in $\mathbb{X}$ is at least 2^k. Although the division property had introduced many beautiful results since it was developed and it seems that it exploits the algebraic degree of the components inside the cipher, the roughly invocation of the algebraic degree of the non-linear components[3] indicates that the division property can obtain better results if we can deal with these components more carefully.

At CRYPTO 2015, Todo [19] showed that the division property could be more useful if the boolean function expressions of an S-box were known. He applied his technique to the cryptanalysis on MISTY1 and found that S_7 had a vulnerable property. Based on this fact, he constructed a new 6-round integral distinguisher and theoretically recovered the secret key of the full MISTY1.

At INDOCRYPT 2015, Zhang and Wu [25] studied the propagation characteristic of the division property for the GFS [15]. They found that it was difficult to directly trace the propagation when the branch number m was not less than 14 due to the rapid expansion of the vectors in the division property. They came up with early reduce technique, which worked by detecting and discarding "use-

[3] Just as Todo's statements in [19], actually, he focused only on the secret S-box restricted by an algebraic degree. But many realistic block ciphers use a public S-box.

less" vectors. With the aid of early reduce technique, they proposed a generic algorithm of constructing integral distinguishers and extended the length of integral distinguishers for Type-1 GFS and Type-2 GFS. Besides, they constructed several 16-round integral distinguishers for LBlock [23] and TWINE [17] for the first time.

The bit-based division property were first proposed by Todo and Morri [20] at FSE 2016. They considered the division property for SIMON [2] family and were motivated by the 5 rounds gap between the 15-round experimental integral distinguisher for SIMON32 found by Wang *et al.* [21] and the 10-round integral distinguisher proposed by Todo [18]. They decomposed the round function of SIMON into every bit and directly used the propagation rules of the division property. As a result, the conventional bit-based division property proved that SIMON32 had a 14-round integral distinguisher.

Inspired by the method of Todo and Morri, we wonder whether the bit-based division property can be used to analyze other bit-oriented block ciphers with S-boxes such as PRESENT [4]. Recently, this question is also considered by Boura and Canteaut in [6] and they found a 4-round integral distinguisher for PRESENT. Note that the only obstacle to efficiently apply conventional bit-based division property lines in the lack of a sophisticated propagation rule for the S-box. By integrating the method of division property with the concrete boolean function expressions of S-box, we come up with a new method to reconsider the propagation rule for **Substitution** operation. Unlike the direct application of the propagation rules for the analysis of SIMON introduced by Todo and Morri, this new method starts from the boolean function expressions of the nonlinear component (*e.g.* S-box) inside an encryption algorithm. We replace the original **Substitution** rule with a more subtle propagation table during tracing the property of a chosen multi-set, and this new method can trace the propagation of division property at the bit level. Since this method escape from the direct application of the propagation rules by using a propagation table, we name our new method *table-aided bit-based division property*, which is often denoted by 'TABBDP' for short in this paper.

We apply TABBDP to find integral distinguishers for bit-oriented block ciphers PRESENT [4] and RECTANGLE [26]. A new higher-order integral distinguisher for lightweight hash function SPONGENT-88 [5] is also proposed with the aid of TABBDP. This method not only can be used to build integral distinguishers for more bit-oriented block ciphers but also can play a part in finding integral distinguishers for some word-oriented block ciphers. To illustrate the utility of TABBDP for word-oriented block ciphers, we successfully use it to find integral distinguishers for LBlock [23] and TWINE [17].

Since TABBDP breaks the vectors into bits, the memory complexity of utilizing it is apparently larger than that of conventional division property operating on words. The exponential growth in the number of vectors belonging to $\mathbb{K}$ also produces the rapid inflation in memory complexity. Additionally, the time used to deal with these vectors also increases. Due to the limitations produced by the huge time and memory complexities, the distinguishers we found are not so well

as we expected. But it is worth to note that TABBDP is a common method for finding integral distinguishers and can be applied to much more bit-oriented block ciphers.

Notice that the early reduce technique proposed by Zhang and Wu [25] can handle a similar problem existing in conventional division property, we adopt this technique to improve TABBDP. By integrating early reduce technique with TABBDP, we construct an 8-round higher-order integral distinguisher for RECTANGLE, which attains one more round than the previous one proposed by the designers of RECTANGLE. Although early reduce technique settles the limitation for memory requirement to some degree, an additional early reduce procedure may increase the time complexity. So there is a trade-off between time and memory. Whether the early reduce procedure should be taken or not is determined by the attackers according to their resource at hand.

1.1 Our Contributions

Table-Aided Bit-Based Division Property By integrating the method of bit-based division property with the concrete boolean function expressions of S-box, we come up with a new method to reconsider the **Substitution** rule, and this new method can trace the propagation of division property at the bit level. In order to verify our methods, we apply it to analyze a toy algorithm named CipherFour. The experimental results show that the integral distinguishers found by TABBDP are indeed valid integral distinguishers. The results also indicate that TABBDP allows us to build longer integral distinguishers than those found by utilizing conventional division property under the same number of chosen plaintexts.

Improved Higher-Order Integral Distinguishers for RECTANGLE The designers of RECTANGLE gave a 7-round higher-order integral distinguisher in [26]. 2^{48} plaintexts were required in order to observe zero-sum property in the output bits after seven rounds of encryption. We firstly propose a new 7-round higher-order integral distinguisher of RECTANGLE by utilizing TABBDP in this paper, whose data complexity is 2^{36}, which is 2^{12} less than the original result. After applying early reduce technique, an 8-round higher-order integral distinguisher for RECTANGLE can be constructed, which attains one more round than the previous one proposed by the designers.

New Integral Distinguishers for PRESENT and SPONGENT-88 In 2008, Z'aba *et al.* [24] found a 3.5-round integral distinguisher by using bit-pattern integral cryptanalysis. After that, Zhang *et al.* [28] presented a unified algorithm of searching for the best possible higher-order integral distinguishers for block ciphers and found a 5-round higher-order integral distinguisher for PRESENT. In 2013, by using specific algebraic properties of S-box and special structure of linear layer, Wu and Wang [22] proposed a new 5-round integral distinguisher and a 7-round integral distinguisher for PRESENT. In this paper,

by utilizing TABBDP, we find a new 5-round integral distinguisher which covers the 5-round integral distinguisher found by Wu and Wang. Although our newly found distinguisher has some overlap with the 5-round distinguisher proposed in [22], we can not deny that TABBDP is a general and automated searching method which escapes from exploiting the tanglesome property in the encryption algorithm. Besides, we propose a new 6-round integral distinguisher.

Dong *et al.* [11] combined the higher-order differential attack and integral attack to construct a zero-sum distinguisher for 14-round SPONGENT-88 in data complexity 2^{84}. In 2015, Fan and Duan [12] followed the same ideas as those in [11] to construct an improved zero-sum distinguisher for 14-round SPONGENT-88 in data complexity 2^{80}, which is 2^4 less than the one in [11]. In this paper, we combine TABBDP with the techniques in [11, 12] and find a new zero-sum distinguisher for 14-round SPONGENT-88 with data complexity 2^{80}.

Bit-Based Division Property Goes to General: Potential Usage Scenarios for Word-Oriented Block Ciphers The table-aided bit-based division property can also be applied to find integral distinguishers for some word-oriented block ciphers. In this paper, we successfully use it to find integral distinguishers for LBlock and TWINE. Although we do not find any new integral distinguishers, we believe that considering the S-box at the bit level is of great importance even for a word-oriented block cipher. A more adequate invocation of the details inside an encryption algorithm usually guarantees a better analysis result. And we expect that TABBDP will find better integral distinguisher for some encryption algorithms with relatively weak S-boxes. The analysis at the bit level for a word-oriented block cipher may become a tendency in the future.

Organization of the Paper. The remainder of this paper is organized as follows. In Section 2, we briefly review some definitions including integral cryptanalysis, higher-order integral distinguisher, conventional division property and bit-based division property for SIMON family. Section 3 presents table-aided bit-based division property first and then illustrates how to improve table-aided bit-based division property by using early reduce technique. Section 4 gives the resulting integral distinguishers for various objects including block ciphers and hash functions by applying table-aided bit-based division property. We point out the future works and conclude the paper in Section 5. Some auxiliary materials are given in Appendixes.

2 Preliminaries

2.1 Notations

In this subsection, we present the notations used throughout this paper. In order to simplify the representation, we will express a long vector in hexadecimal format and make it `verbatim` font. For example, for a 32-bit vector

$$\boldsymbol{a} = [0111\ 1001\ 0011\ 1010\ 1001\ 0111\ 0000\ 1111],$$

we will present it with $\boldsymbol{a} = \texttt{[793a970f]}$ in this paper. For a n-bit vector, the bit positions are label as 0 to $n-1$ from left to right.

The following notations was also used in [18]. Please refer to [18] for more details.

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 $wt(a) = \sum_{i=0}^{n-1} a[i]$. Let $\mathbf{1} \in \mathbb{F}_2^n$ be a vector whose all elements are all 1, and let $\mathbf{0} \in \mathbb{F}_2^n$ be a vector whose all elements are all 0.

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, $\boldsymbol{k} \not\succeq \boldsymbol{k'}$. $\mathbb{K} \leftarrow \boldsymbol{k}$ means that $\mathbb{K}$ turns into $\mathbb{K} \bigcup \{\boldsymbol{k}\}$.

Let $\mathbb{K}$ be a set whose elements take $\mathbb{Z}^m$. Let $\mathbb{S}_{\mathbb{K}}^{\ell_0,\ell_1,\ldots,\ell_{m-1}}$ be a subset of $\mathbb{F}_2^{\ell_0} \times \mathbb{F}_2^{\ell_1} \times \cdots \mathbb{F}_2^{\ell_{m-1}}$ defined as

$$\mathbb{S}_{\mathbb{K}}^{\ell_0,\ell_1,\ldots,\ell_{m-1}} = \left\{ \boldsymbol{a} \in \mathbb{F}_2^{\ell_0} \times \mathbb{F}_2^{\ell_1} \times \cdots \mathbb{F}_2^{\ell_{m-1}} \Big| \bigcup_{\boldsymbol{k} \in \mathbb{K}} \Big(Wt(\boldsymbol{a}) \succeq \boldsymbol{k} \Big) \right\}.$$

Definition 1 (Bit Product Function [18]). *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 Integral Cryptanalysis and Higher-Order Integral Distinguisher

Following the definition in [14], let $\mathbb{X}$ be a multi-set of vectors. An integral over $\mathbb{X}$ is defined as the sum of all vectors in $\mathbb{X}$. In other words, the integral is $\int \mathbb{X} = \sum_{\boldsymbol{x} \in \mathbb{X}} \boldsymbol{x}$. In integral cryptanalysis, one tries to predict the values of the integrals after a certain number of rounds of encryption. Specifically, if $\int \mathbb{X} = \sum_{\boldsymbol{x} \in \mathbb{X}} \boldsymbol{x} = 0$, we say that $\mathbb{X}$ has a zero-sum property.

Knudsen and Wagner [14] also introduced an important extension to previous work named the higher-order integral attack which is analogous to higher-order differentials. Consider a set $\widetilde{\mathbb{X}} = \mathbb{X}_0 \bigcup \mathbb{X}_1 \bigcup \cdots \bigcup \mathbb{X}_{s-1}$ made up of s sets of vectors, where each $\mathbb{X}_i$ forms an integral. Then clearly, if one can determine the sum of elements of $\mathbb{X}_i$ for each i, then one also can determine the sum of all vectors in $\widetilde{\mathbb{X}}$.

2.3 Division Property

The division property is a new method to find integral distinguishers, which was proposed in [18]. In this subsection, we briefly show the definitions and the propagation rules.

The division property is defined for a multi-set, and it is calculated by using the bit product function.

Definition 2 (Division Property [18]). *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 $Wt(\boldsymbol{u}) \succeq \boldsymbol{k}^{(i)}$.

An Intuitive Example To understand the above definitions and symbols, we will give an intuitive example here. We consider $\mathcal{D}^{6,6}_{\mathbb{K}}$, where $\mathbb{K} = \{[1,4],[3,3]\}$. Note that the region covered by red or blue in Fig. 1 is the 'unknown' region, which will be denoted by $\mathcal{U}$ throughout this paper. The region $\mathcal{U}$ is defined by two vectors in $\mathbb{K}$, and the vectors in $\mathbb{K}$ are called *boundary points*.

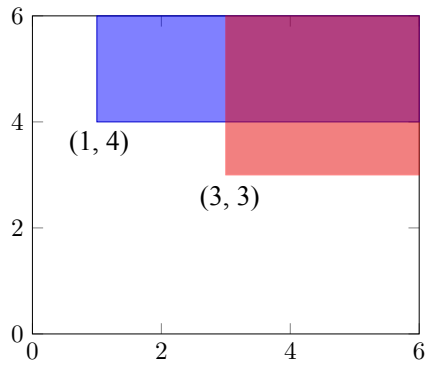


Fig. 1: Illustration of $\mathcal{U}$.

Propagation Rules of Division Property [18] proves some propagation rules and these rules are summarized in [19] as follows.

Rule 1 (Substitution) *Let F be a function that consists of m S-boxes, where the bit length and the algebraic degree of the i-th S-box is ℓ_i bits and d_i, respectively. The input and the output take a value of $\mathbb{F}_2^{\ell_0} \times \mathbb{F}_2^{\ell_1} \times \cdots \times \mathbb{F}_2^{\ell_{m-1}}$, and $\mathbb{X}$ and $\mathbb{Y}$ denote the input multi-set and the output multi-set, respectively. Assuming that the multi-set $\mathbb{X}$ has division property $\mathcal{D}_{\mathbb{K}}^{\ell_0,\ell_1,\ldots,\ell_{m-1}}$, where $\mathbb{K}$ denotes a set of m-dimensional vectors whose i-th element takes a value between 0 and ℓ_i, the division property of the multi-set $\mathbb{Y}$ is $\mathcal{D}_{\mathbb{K}'}^{\ell_0,\ell_1,\ldots,\ell_{m-1}}$ as*

$$\mathbb{K}' \leftarrow \left(\left\lceil \frac{k_0}{d_0} \right\rceil, \left\lceil \frac{k_1}{d_1} \right\rceil, \cdots, \left\lceil \frac{k_{m-1}}{d_{m-1}} \right\rceil \right), \forall \boldsymbol{k} \in \mathbb{K}.$$

Rule 2 (Copy) *Let F be a copy function, where the input x takes a value of $\mathbb{F}_2^n$ 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 $\mathcal{D}_k^n$, the division property of the multi-set $\mathbb{Y}$ is $\mathcal{D}_{\mathbb{K}'}^{n,n}$ as*

$$\mathbb{K}' \leftarrow (k-i, i), \text{ for } 0 \leqslant i \leqslant k.$$

Rule 3 (Compression by `XOR`) *Let F be a function compressed by an `XOR`, where the input (x_1, x_2) takes a value of $\mathbb{F}_2^n \times \mathbb{F}_2^n$ 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 $\mathcal{D}_{\mathbb{K}}^{n,n}$, the division property of the multi-set $\mathbb{Y}$ is $\mathcal{D}_{k'}^n$ as*

$$k' = \min_{(k_0,k_1)\in\mathbb{K}} \{k_1 + k_2\}.$$

Here, if the minimum value of k' is larger than n, the propagation characteristic of the division property is aborted. Namely, a value of $\bigoplus_{y\in\mathbb{Y}} \pi_v(y)$ is 0 for all $v \in \mathbb{F}_2^n$.

Rule 4 (Split) *Let F be a split function, where the input x takes a value of $\mathbb{F}_2^n$ and the output is calculated as $x = y_0 \| y_1$, where (y_0, y_1) takes a value of $\mathbb{F}_2^{n_0} \times \mathbb{F}_2^{n-n_0}$. 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 $\mathcal{D}_k^n$, the division property of the multi-set $\mathbb{Y}$ is $\mathcal{D}_{\mathbb{K}'}^{n_0,n-n_0}$ as*

$$\mathbb{K}' \leftarrow (k-i, i), \text{ for } 0 \leqslant i \leqslant k.$$

Here, $(k-i)$ is less than or equal to n_0, and i is less than or equal to $n - n_0$.

Rule 5 (Concatenation) *Let F be a concatenation function, where the input (x_0, x_1) take a value of $\mathbb{F}_2^{n_0} \times \mathbb{F}_2^{n_1}$ and the output is calculated as $y = x_1 \| x_2$. Let $\mathbb{X}$ and $\mathbb{Y}$ be the input multi-set and output multi-set, respectively. Assuming that the multi-set $\mathbb{X}$ has the division property $\mathcal{D}_{\mathbb{K}}^{n_0,n_1}$, the division property of the multi-set $\mathbb{Y}$ is $\mathcal{D}_{k'}^{n_0+n_1}$ as*

$$k' = \min_{(k_0,k_1)\in\mathbb{K}} \{k_1 + k_2\}.$$

2.4 Bit-Based Division Property for SIMON Family [20]

At FSE 2016, Todo and Morii [20] introduce the bit-based division property for SIMON [2] family and use the conventional bit-based division property to find a 14-round integral distinguisher for SIMON32. We briefly review the main idea here and refer the readers to [20] for more details.

When attackers use division property to find integral distinguisher for an n-bit block cipher, they will consider $\mathcal{D}_{\mathbb{K}}^{\ell_0,\ell_1,\ldots,\ell_{m-1}}$, where ℓ_i and m are chosen by attackers in the range of $n = \sum_{i=0}^{m-1} \ell_i$. But for a bit-oriented block cipher like SIMON, they will decompose the round function into every bit and directly apply the conventional division property. In other word, the conventional bit-based division property consider $\mathcal{D}_{\mathbb{K}}^{1^n}$. In [20], the authors decompose the round function of SIMON2n into every bit and directly apply the propagation rules. They focus on only one bit of the right half in SIMON2n. The core operation of the round function is represented as Fig. 2. They first analyze the division property $\mathcal{D}_{\mathbb{K}}^{1^4}$, since the input and output bit length of the core operation are both 4. By applying the five rules introduced in Section 2.3, we can get the propagation of division property, Table 1 summarizes the propagation characteristics from $\mathcal{D}_{\boldsymbol{k}}^{1^4}$ to $\mathcal{D}_{\mathbb{K}'}^{1^4}$.

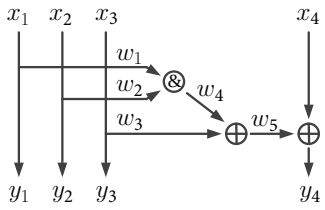


Fig. 2: Core Operation of SIMON Family [20].

Let $\mathcal{D}_{\mathbb{K}}^{1^4}$ and $\mathcal{D}_{\mathbb{K}'}^{1^4}$ be the division property of the input multi-set and the output multi-set, respectively. To get $\mathcal{D}_{\mathbb{K}'}^{1^4}$ from $\mathcal{D}_{\mathbb{K}}^{1^4}$, they first independently calculate vectors belonging to $\mathbb{K}'$ by evaluating the propagation from every vector in $\mathbb{K}$. Then $\mathbb{K}'$ is represented as the union of all calculated vectors. Finally, if there exist $\boldsymbol{k} \in \mathbb{K}'$ and $\boldsymbol{k}' \in \mathbb{K}'$ such that $\boldsymbol{k} \succeq \boldsymbol{k}'$, $\boldsymbol{k}$ is removed from $\mathbb{K}'$ because the vector is redundant.

The round function of SIMON2n repeats the core operation for all n-bit value in the right half, *i.e.*, they consider $\mathcal{D}_{\mathbb{K}}^{1^{2n}}$. In every core operation, they only focus on four bits and evaluate the propagation independent of other $2n-4$ bits. In [20], the authors show a toy example to help readers to understand the propagation procedure and the readers can check [20] for more details.

Table 1: Propagation of Division Property for Core Operation in SIMON [20]

Input $\mathcal{D}_{\boldsymbol{k}}^{1^4}$	Output $\mathcal{D}_{\mathbb{K}'}^{1^4}$
$\boldsymbol{k}=[0,0,0,0]$	$\mathbb{K}'=\{[0,0,0,0]\}$
$\boldsymbol{k}=[1,0,0,0]$	$\mathbb{K}'=\{[1,0,0,0],[0,0,0,1]\}$
$\boldsymbol{k}=[0,1,0,0]$	$\mathbb{K}'=\{[0,1,0,0],[0,0,0,1]\}$
$\boldsymbol{k}=[1,1,0,0]$	$\mathbb{K}'=\{[1,1,0,0],[0,0,0,1]\}$
$\boldsymbol{k}=[0,0,1,0]$	$\mathbb{K}'=\{[0,0,1,0],[0,0,0,1]\}$
$\boldsymbol{k}=[1,0,1,0]$	$\mathbb{K}'=\{[1,0,1,0],[0,0,1,1],[1,0,0,1]\}$
$\boldsymbol{k}=[0,1,1,0]$	$\mathbb{K}'=\{[0,1,1,0],[0,0,1,1],[0,1,0,1]\}$
$\boldsymbol{k}=[1,1,1,0]$	$\mathbb{K}'=\{[1,1,1,0],[0,0,1,1],[1,1,0,1]\}$
$\boldsymbol{k}=[k_1,k_2,k_3,1]$	$\mathbb{K}'=\{[k_1,k_2,k_3,1]\}$

3 Table-Aided Bit-Based Division Property

In this section, we will introduce table-aided bit-based division property first and then illustrate how to improve table-aided bit-based division property by using early reduce technique.

3.1 Table-Aided Bit-Based Division Property

After running into the bit-based division property on SIMON, we wonder whether this bit-based division property can be used to analyze other bit-oriented block cipher with S-box such as PRESENT [4]. An intuitive idea is to expand the S-box with the logical operators `AND` and `XOR`. Theoretically, a picture like Fig. 2 can be drawn for PRESENT's S-box according to the corresponding boolean functions. Note that the boolean functions for PRESENT's S-box are far more complicated than those for the core operation of SIMON, it is not easy to draw this picture. (We illustrate the boolean function expressions for the core operation of SIMON and PRESENT's S-box in the left side and in the right side of Fig. 3, respectively.) Besides, it is not a simple task to build a propagation table in this case just like Table 1 for SIMON since we have to consider much more **Split** and **Copy** operations.

Since the intuitive idea does not make much sense, we turn to try to explain Table 1 by integrating the division property with the boolean functions. We will take $\boldsymbol{k}=[1,0,0,0]$ as an example. By the definition of division property, we know that $\bigoplus_{x\in\mathbb{X}} x_1$ is unknown and this boundary point defines the unknown region $\mathcal{U}_{\mathbb{X}}$ of the input multi-set $\mathbb{X}$. Besides, we have the following observation.

Observation 1 *Suppose that the multi-set $\mathbb{X}$ follows $\mathcal{D}_{\{[1,0,0,0]\}}^{1^4}$. For any monomial including x_1, its sum over all $x\in\mathbb{X}$ is unknown.*

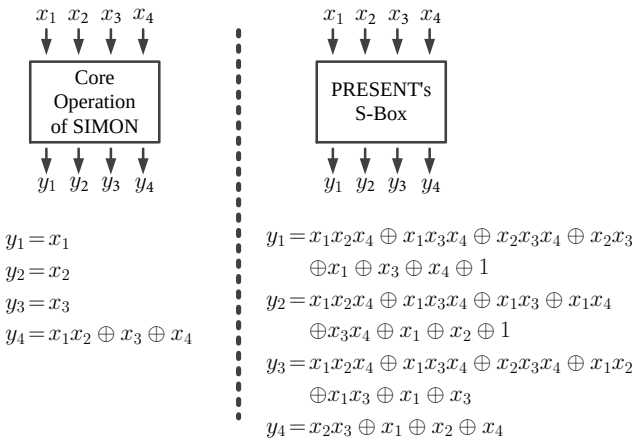


Fig. 3: Boolean Function Expansions for Core operation of SIMON and PRESENT's S-box

We aim at deducing the unknown region $\mathcal{U}_{\mathbb{Y}}$ of the output multi-set $\mathbb{Y}$. For this purpose, we have to use the informations of the input multi-set $\mathbb{X}$. In Table 2, we make a correspondence between the vector $\boldsymbol{k}$ and the parity defined by this vector for the output multi-set $\mathbb{Y}$. The monomials which are dyed in red are those monomials including x_1 and these red monomials make the corresponding parities for $\mathbb{Y}$ to be unknown by **Observation 1**. Those vectors matching these unknown parities constitute the unknown region $\mathcal{U}_{\mathbb{Y}}$ of $\mathbb{Y}$.

Alternatively, if the division property of the multi-set $\mathbb{X}$ is $\mathcal{D}^{1^4}_{[1,0,0,0]}$, the division property of the multi-set $\mathbb{Y}$ is $\mathcal{D}^{1^4}_{\mathbb{K}'}$, as

$$\begin{aligned}\mathbb{K}' = \{&[0,0,0,1],[0,0,1,1],[0,1,0,1],[0,1,1,1],\\ &[1,0,0,0],[1,0,0,1],[1,0,1,0],[1,0,1,1],\\ &[1,1,0,0],[1,1,0,1],[1,1,1,0],[1,1,1,1]\}.\end{aligned}$$

Note that some entries in $\mathbb{K}'$ are redundant. After reducing, $\mathbb{K}'$ becomes

$$\mathbb{K}' = \{[1,0,0,0],[0,0,0,1]\}.$$

This result agrees with the propagation rule deduced by conventional bit-based division property, which can be checked in the third line in Table 1. We use the former idea to analyze every row of Table 1 and get exactly the same results as those in Table 1. Since we only use boolean functions in our analysis process, we can apply this idea to analyze other bit-oriented block cipher beside SIMON.

For some bit-oriented block ciphers like PRESENT [4] and RECTANGLE [26], the only obstacle to apply conventional bit-based division property lies in the lack of propagation rule for S-box. By integrating the method of division property with the concrete boolean function expression of S-box, we can

Table 2: Correspondence between the Vectors and the Parities for the Core Operation of SIMON

Vector $\boldsymbol{k}$	Parity for the Output Multi-Set $\mathbb{Y}$ Defined by $\boldsymbol{k}$
$\boldsymbol{k} = [0,0,0,0]$	$\bigoplus_{y\in\mathbb{Y}} 1 = \bigoplus_{x\in\mathbb{X}} 1$
$\boldsymbol{k} = [0,0,0,1]$	$\bigoplus_{y\in\mathbb{Y}} y_4 = \bigoplus_{x\in\mathbb{X}} (x_1x_2 \oplus x_3 \oplus x_4)$ $= (\bigoplus_{x\in\mathbb{X}} x_1x_2) \oplus (\bigoplus_{x\in\mathbb{X}} x_3) \oplus (\bigoplus_{x\in\mathbb{X}} x_4)$
$\boldsymbol{k} = [0,0,1,0]$	$\bigoplus_{y\in\mathbb{Y}} y_3 = \bigoplus_{x\in\mathbb{X}} x_3$
$\boldsymbol{k} = [0,0,1,1]$	$\bigoplus_{y\in\mathbb{Y}} y_3y_4 = \bigoplus_{x\in\mathbb{X}} (x_3(x_1x_2 \oplus x_3 \oplus x_4))$ $= (\bigoplus_{x\in\mathbb{X}} x_1x_2x_3) \oplus (\bigoplus_{x\in\mathbb{X}} x_3x_4) \oplus (\bigoplus_{x\in\mathbb{X}} x_3)$
$\boldsymbol{k} = [0,1,0,0]$	$\bigoplus_{y\in\mathbb{Y}} y_2 = \bigoplus_{x\in\mathbb{X}} x_2$
$\boldsymbol{k} = [0,1,0,1]$	$\bigoplus_{y\in\mathbb{Y}} y_2y_4 = \bigoplus_{x\in\mathbb{X}} (x_2(x_1x_2 \oplus x_3 \oplus x_4))$ $= (\bigoplus_{x\in\mathbb{X}} x_1x_2) \oplus (\bigoplus_{x\in\mathbb{X}} x_2x_3) \oplus (\bigoplus_{x\in\mathbb{X}} x_2x_4)$
$\boldsymbol{k} = [0,1,1,0]$	$\bigoplus_{y\in\mathbb{Y}} y_2y_3 = \bigoplus_{x\in\mathbb{X}} x_2x_3$
$\boldsymbol{k} = [0,1,1,1]$	$\bigoplus_{y\in\mathbb{Y}} y_2y_3y_4 = \bigoplus_{x\in\mathbb{X}} (x_2x_3(x_1x_2 \oplus x_3 \oplus x_4))$ $= (\bigoplus_{x\in\mathbb{X}} x_1x_2x_3) \oplus (\bigoplus_{x\in\mathbb{X}} x_2x_3x_4) \oplus (\bigoplus_{x\in\mathbb{X}} x_2x_3)$
$\boldsymbol{k} = [1,0,0,0]$	$\bigoplus_{y\in\mathbb{Y}} y_1 = \bigoplus_{x\in\mathbb{X}} x_1$
$\boldsymbol{k} = [1,0,0,1]$	$\bigoplus_{y\in\mathbb{Y}} y_1y_4 = \bigoplus_{x\in\mathbb{X}} (x_1(x_1x_2 \oplus x_3 \oplus x_4))$ $= (\bigoplus_{x\in\mathbb{X}} x_1x_2) \oplus (\bigoplus_{x\in\mathbb{X}} x_1x_3) \oplus (\bigoplus_{x\in\mathbb{X}} x_1x_4)$
$\boldsymbol{k} = [1,0,1,0]$	$\bigoplus_{y\in\mathbb{Y}} y_1y_3 = \bigoplus_{x\in\mathbb{X}} x_1x_3$
$\boldsymbol{k} = [1,0,1,1]$	$\bigoplus_{y\in\mathbb{Y}} y_1y_3y_4 = \bigoplus_{x\in\mathbb{X}} (x_1x_3(x_1x_2 \oplus x_3 \oplus x_4))$ $= (\bigoplus_{x\in\mathbb{X}} x_1x_2x_3) \oplus (\bigoplus_{x\in\mathbb{X}} x_1x_3x_4) \oplus (\bigoplus_{x\in\mathbb{X}} x_1x_3)$
$\boldsymbol{k} = [1,1,0,0]$	$\bigoplus_{y\in\mathbb{Y}} y_1y_2 = \bigoplus_{x\in\mathbb{X}} x_1x_2$
$\boldsymbol{k} = [1,1,0,1]$	$\bigoplus_{y\in\mathbb{Y}} y_1y_2y_4 = \bigoplus_{x\in\mathbb{X}} (x_1x_2(x_1x_2 \oplus x_3 \oplus x_4))$ $= (\bigoplus_{x\in\mathbb{X}} x_1x_2x_3) \oplus (\bigoplus_{x\in\mathbb{X}} x_1x_2x_4) \oplus (\bigoplus_{x\in\mathbb{X}} x_1x_2)$
$\boldsymbol{k} = [1,1,1,0]$	$\bigoplus_{y\in\mathbb{Y}} y_1y_2y_3 = \bigoplus_{x\in\mathbb{X}} x_1x_2x_3$
$\boldsymbol{k} = [1,1,1,1]$	$\bigoplus_{y\in\mathbb{Y}} y_1y_2y_3y_4 = \bigoplus_{x\in\mathbb{X}} (x_1x_2x_3(x_1x_2 \oplus x_3 \oplus x_4))$ $= \bigoplus_{x\in\mathbb{X}} x_1x_2x_3x_4$

successfully overcome this obstacle. We replace the original **Substitution** rule with a more subtle propagation table, which can reflect more details of S-box. For this reason, we name our new method *table-aided bit-based division property* (TABBDP). Now, we use PRESENT's S-box as an example. Again, based on the Boolean function expressions of PRESENT's S-box, a correspondence table similar to Table 2 can be built for PRESENT. Please see Table 3.

Table 3: Correspondence between the Vectors and the Parities for PRESENT's S-box

Vector $\boldsymbol{k}$	Parity for the Output Multi-Set $\mathbb{Y}$ Defined by $\boldsymbol{k}$
$\boldsymbol{k}=[0,0,0,0]$	$\bigoplus_{y\in\mathbb{Y}} 1 = \bigoplus_{x\in\mathbb{X}} 1$
$\boldsymbol{k}=[0,0,0,1]$	$\bigoplus_{y\in\mathbb{Y}} y_4 = \bigoplus_{x\in\mathbb{X}}(x_2x_3 \oplus x_1 \oplus x_2 \oplus x_4)$
$\boldsymbol{k}=[0,0,1,0]$	$\bigoplus_{y\in\mathbb{Y}} y_3 = \bigoplus_{x\in\mathbb{X}}(x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_1x_2 \oplus x_1x_3 \oplus x_1 \oplus x_3)$
$\boldsymbol{k}=[0,0,1,1]$	$\bigoplus_{y\in\mathbb{Y}} y_3y_4 = \bigoplus_{x\in\mathbb{X}}(x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_1x_2 \oplus x_1x_4 \oplus x_3x_4 \oplus x_1)$
$\boldsymbol{k}=[0,1,0,0]$	$\bigoplus_{y\in\mathbb{Y}} y_2 = \bigoplus_{x\in\mathbb{X}}(x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_1x_3 \oplus x_1x_4 \oplus x_3x_4 \oplus x_1 \oplus x_2 \oplus 1)$
$\boldsymbol{k}=[0,1,0,1]$	$\bigoplus_{y\in\mathbb{Y}} y_2y_4 = \bigoplus_{x\in\mathbb{X}}(x_1x_2x_3 \oplus x_1x_3 \oplus x_1x_4 \oplus x_2x_4 \oplus x_3x_4 \oplus x_4)$
$\boldsymbol{k}=[0,1,1,0]$	$\bigoplus_{y\in\mathbb{Y}} y_2y_3 = \bigoplus_{x\in\mathbb{X}}(x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_1x_4 \oplus x_2x_3 \oplus x_3x_4 \oplus x_3)$
$\boldsymbol{k}=[0,1,1,1]$	$\bigoplus_{y\in\mathbb{Y}} y_2y_3y_4 = \bigoplus_{x\in\mathbb{X}}(x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_1x_3)$
$\boldsymbol{k}=[1,0,0,0]$	$\bigoplus_{y\in\mathbb{Y}} y_1 = \bigoplus_{x\in\mathbb{X}}(x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_2x_3 \oplus x_1 \oplus x_3 \oplus x_4 \oplus 1)$
$\boldsymbol{k}=[1,0,0,1]$	$\bigoplus_{y\in\mathbb{Y}} y_1y_4 =$ $\bigoplus_{x\in\mathbb{X}}(x_1x_2x_4 \oplus x_2x_3x_4 \oplus x_1x_2 \oplus x_1x_3 \oplus x_2x_3 \oplus x_2x_4 \oplus x_3x_4 \oplus x_2)$
$\boldsymbol{k}=[1,0,1,0]$	$\bigoplus_{y\in\mathbb{Y}} y_1y_3 = \bigoplus_{x\in\mathbb{X}}(x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_1x_3 \oplus x_1x_4 \oplus x_2x_3 \oplus x_3x_4)$
$\boldsymbol{k}=[1,0,1,1]$	$\bigoplus_{y\in\mathbb{Y}} y_1y_3y_4 = \bigoplus_{x\in\mathbb{X}}(x_1x_2x_3 \oplus x_2x_3x_4 \oplus x_1x_3 \oplus x_3x_4)$
$\boldsymbol{k}=[1,1,0,0]$	$\bigoplus_{y\in\mathbb{Y}} y_1y_2 =$ $\bigoplus_{x\in\mathbb{X}}(x_1x_3x_4 \oplus x_1x_2 \oplus x_2x_3 \oplus x_2x_4 \oplus x_3x_4 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus 1)$
$\boldsymbol{k}=[1,1,0,1]$	$\bigoplus_{y\in\mathbb{Y}} y_1y_2y_4 = \bigoplus_{x\in\mathbb{X}}(x_1x_2x_3 \oplus x_1x_3x_4 \oplus x_2x_3x_4 \oplus x_1x_3)$
$\boldsymbol{k}=[1,1,1,0]$	$\bigoplus_{y\in\mathbb{Y}} y_1y_2y_3 = \bigoplus_{x\in\mathbb{X}}(x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3 \oplus x_1x_4)$
$\boldsymbol{k}=[1,1,1,1]$	$\bigoplus_{y\in\mathbb{Y}} y_1y_2y_3y_4 = \bigoplus_{x\in\mathbb{X}}(x_1x_2x_3x_4 \oplus x_1x_2x_3 \oplus x_1x_3x_4 \oplus x_1x_3)$

An analysis similar to the former one for SIMON's core operation gives the propagation table for PRESENT's S-box[4]. Please refer to Table 4 for the propagation rule for PRESENT's S-box. Then the integral distinguishers for PRESENT based on TABBDP can be built accordingly.

Similar to the realization of bit-based division property on SIMON. Let $\mathcal{D}_{\mathbb{K}}^{1^4}$ and $\mathcal{D}_{\mathbb{K}'}^{1^4}$ be the division property of the input multi-set and the output multi-set

[4] Note that this procedure can be realized with the help of a program.

Table 4: Propagation of Division Property for PRESENT's S-box

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

for PRESENT's S-box, respectively. To get $\mathcal{D}_{\mathbb{K}'}^{1^4}$ from $\mathcal{D}_{\mathbb{K}}^{1^4}$, we first independently calculate vectors belonging to $\mathbb{K}'$ by evaluating the propagation from every vector in $\mathbb{K}$ according to propagation rule listed in Table 4. Then $\mathbb{K}'$ is represented as the union of all calculated vectors. Finally, if there exist $\boldsymbol{k} \in \mathbb{K}'$ and $\boldsymbol{k}' \in \mathbb{K}'$ such that $\boldsymbol{k} \succeq \boldsymbol{k}'$, $\boldsymbol{k}$ is removed form $\mathbb{K}'$ because the vector is redundant. We can define the unknown region only if the boundary points are known. Also, we only need to evaluate the propagation of those boundary points since the boundary points move faster than those inner points located in the unknown region. In other words, the unknown region defined by the boundary points always contains the region defined by any one of the inner point. The nonlinear layer of PRESENT repeats the S-box's propagation rule for all 16 S-boxes. We only focus on one S-box at a time and evaluate the propagation independent of other 60 bits. For the propagation rule of PRESENT's linear layer, all we need to do is just a bit permutation for every vector $\boldsymbol{k}$.

A generic algorithm used to construct integral distinguishers for bit-oriented block ciphers is given in Algorithm 1. The functions used in Algorithm 1 are explained as follows.

- $TerminateCondition(\cdot)$ is used to determine when the procedure should be terminated. In other words, the propagation procedure should be stopped when all $\boldsymbol{k}$s' hamming weights are equal to 1.
- $PropagateTable(\cdot)$ refers to the process of accessing the propagation table.
- $BitPermutation(\cdot)$ deals with the bit permutation.
- $AddDivision(\cdot)$ adds the resulting division property of every vector to the final division property.
- $Reduce(\cdot)$ eliminates those redundant vectors.
- $EarlyReduce(\cdot)$ refers to the process of accessing the early reduce table. This procedure will be introduced soon.

For Algorithm 1, we have the following remarks.

Remark 1. The overall structure of Algorithm 1 is similar to Algorithm 1 in [25]. Our algorithm are used to search integral distinguisher for bit-oriented block ciphers while their algorithm is used to find integral distinguisher for GFS.

Remark 2. Theoretically, our algorithm can only be used to find integral distinguisher for certain bit-oriented block cipher whose linear layer is just a permutation. But when we deal with a word-oriented block cipher whose linear layer is not a permutation, like TWINE, we will use some techniques to transform its linear layer to be a permutation. This will be explained in the following.

Remark 3. In order to know which bit position satisfy zero-sum property, we can output the division property of the last intermediate round.

Verification In order to verify our new method and to give the readers a further understanding, we illustrate our technique with a toy algorithm named Cipher-Four [13] in Appendix A. We use TABBDP to find the integral distinguishers for

Algorithm 1: Integral Distinguisher Searching Algorithm for Bit-Oriented Block Ciphers

Input: Division property of the input set $\mathbb{K} = \{\boldsymbol{k}^{(0)}, \boldsymbol{k}^{(1)}, \ldots, \boldsymbol{k}^{(m-1)}\}$
Output: The length of the integral distinguisher

```
r ← 0
q ← m
{k^(0), k^(1), ..., k^(q-1)} ← {k^(0), k^(1), ..., k^(m-1)}
while TerminateCondition({k^(0), k^(1), ..., k^(q-1)}) do
    r ← r + 1
    Ω ← ∅
    p ← 0
    for i = 0 to q − 1 do
        {k_i^0, k_i^1, ..., k_i^(q_i−1)} ← PropagateTable(k^(i))
        {k_i^0, k_i^1, ..., k_i^(q_i−1)} ← BitPermutation({k_i^0, k_i^1, ..., k_i^(q_i−1)})
        (Ω, p) ← AddDivision({k_i^0, k_i^1, ..., k_i^(q_i−1)})
    end for
    (Ω, p) ← Reduce(Ω)
    {(Ω, p) ← EarlyReduce(Ω) This procedure can be omitted accordingly.}
    q ← p
    Ω ← {k^(0), k^(1), ..., k^(q−1)}
end while
return r
```

CipherFour first and then randomly generate 400 keys, and exhaustively search for the integral distinguishers. The test results show that the integral distinguishers found by TABBDP are indeed correct integral distinguishers. And the newly found integral distinguishers are longer than the integral distinguishers found by using conventional division property. All these facts convince us that TABBDP is a valid and efficient method to find integral distinguishers for several bit-oriented block ciphers.

3.2 Improved Table-Aided Bit-Based Division Property

When we attempt to apply TABBDP, we find that the huge time and memory complexities make up the main obstacles. On the one hand, it breaks the intermediate states into bits and we need more space to store the vectors. On the other hand, the exponential growth in the number of vectors included in $\mathbb{K}$ also produces rapid inflation in memory complexity. As such, the time used to deal with these vectors also increases.

The similar problem is also encountered by others. Zhang and Wu [25] evaluated the security of the GFS against integral attack and found that it was difficult to directly trace the propagation when the branch number m of GFS was larger than 14 due to the rapid expansion of the vectors in the division property. They devised a technique named early reduce technique to simplify the procedure, which worked by detecting and discarding "useless" vectors. In this

section, we will briefly review this technique first and then apply this technique to TABBDP. With the help of early reduce technique, we construct an 8-round higher-order integral distinguisher for RECTANGLE with data complexity 2^{48}.

Early Reduce Technique This technique is based on the following observation:

Observation 2 ([25], Observation 1) *Let $\boldsymbol{k}$ and $\boldsymbol{k}'$ be two vectors which respectively propagates to $\{\boldsymbol{k}^{(0)}, \boldsymbol{k}^{(1)}, \ldots, \boldsymbol{k}^{(q-1)}\}$ and $\{\boldsymbol{k}'^{(0)}, \boldsymbol{k}'^{(1)}, \ldots, \boldsymbol{k}'^{(q'-1)}\}$ through the round function. If there exists a vector $\boldsymbol{k}'^{(j)} \in \{\boldsymbol{k}'^{(0)}, \boldsymbol{k}'^{(1)}, \ldots, \boldsymbol{k}'^{(q'-1)}\}$, such that $\boldsymbol{k}^{(i)} \succeq \boldsymbol{k}'^{(j)}$ for each $\boldsymbol{k}^{(i)} \in \{\boldsymbol{k}^{(0)}, \boldsymbol{k}^{(1)}, \ldots, \boldsymbol{k}^{(q-1)}\}$, $\Omega \cup \{\boldsymbol{k}, \boldsymbol{k}'\}$ and $\Omega \cup \{\boldsymbol{k}'\}$ propagate to the same division property for any vector set Ω.*

An intuitive example is given in [25], please refer to the corresponding part for more details.

In order to utilize this technique, an additional procedure are required. We should discards vector $\boldsymbol{k}$ if there exist $\boldsymbol{k}' \in \Omega$ satisfying Observation 2 after reducing the redundant vectors in $\mathbb{K}$. In doing so, it can amazingly reduce the division property and does not change the division property achieved through one round function.

Potential of Table-Aided Bit-Based Division Property The early reduce technique also can be applied to TABBDP, this can be realized by an addition table called *early reduce table*. This table includes two columns. The first column represents the vector $\boldsymbol{k}'$ and the corresponding values in the second column is the vectors $\boldsymbol{k}$ that can be reduced by $\boldsymbol{k}'$[5]. We use RECTANGLE's S-box as an example and the early reduce table for RECTANGLE's S-box is given in Table 5. When we want to apply early reduce technique to TABBDP, an additional early reduce procedure should be performed following the original reduce procedure. The procedure is surrounded by braces in Algorithm 1.

In order to illustrate the effect of using early reduce technique, we apply this technique to the analysis of CipherFour[6] and give a comparison for the number of vectors in $\mathbb{K}$ before and after using this technique, which is shown in Table 6. From Table 6, we observe that the number of vectors in $\mathbb{K}$ has a tremendous decline.

Although early reduce technique settles the limitation for memory requirement to some degree, an additional early reduce procedure may increase the time complexity. So there is a trade-off between time and memory. Whether the early reduce procedure should be taken or not is determined by the attackers according to their resource at hand.

[5] This table also can be built with the help of a program.
[6] We also use PRESENT's S-box here.

Table 5: Early Reduce Table for RECTANGLE's S-box

$\boldsymbol{k}'$	The Vectors Can Be Early Reduced by $\boldsymbol{k}'$
$\boldsymbol{k}' = [0,0,0,0]$	$[0,0,0,0], [0,0,0,1], [0,0,1,0], [0,0,1,1], [0,1,0,0], [0,1,0,1],$ $[0,1,1,0], [0,1,1,1], [1,0,0,0], [1,0,0,1], [1,0,1,0], [1,0,1,1],$ $[1,1,0,0], [1,1,0,1], [1,1,1,0], [1,1,1,1]$
$\boldsymbol{k}' = [0,0,0,1]$	$[0,0,0,1], [0,0,1,0], [0,0,1,1], [0,1,0,0], [0,1,0,1], [0,1,1,0],$ $[0,1,1,1], [1,0,0,0], [1,0,0,1], [1,0,1,0], [1,0,1,1], [1,1,0,0],$ $[1,1,0,1], [1,1,1,0], [1,1,1,1]$
$\boldsymbol{k}' = [0,0,1,0]$	$[0,0,0,1], [0,0,1,0], [0,0,1,1], [0,1,0,0], [0,1,0,1], [0,1,1,0],$ $[0,1,1,1], [1,0,0,0], [1,0,0,1], [1,0,1,0], [1,0,1,1], [1,1,0,0],$ $[1,1,0,1], [1,1,1,0], [1,1,1,1]$
$\boldsymbol{k}' = [0,0,1,1]$	$[0,0,1,1], [0,1,1,1], [1,0,1,1], [1,1,0,1], [1,1,1,1]$
$\boldsymbol{k}' = [0,1,0,0]$	$[0,0,0,1], [0,0,1,0], [0,0,1,1], [0,1,0,0], [0,1,0,1], [0,1,1,0],$ $[0,1,1,1], [1,0,0,0], [1,0,0,1], [1,0,1,0], [1,0,1,1], [1,1,0,0],$ $[1,1,0,1], [1,1,1,0], [1,1,1,1]$
$\boldsymbol{k}' = [0,1,0,1]$	$[0,1,0,1], [0,1,1,0], [0,1,1,1], [1,0,0,1], [1,0,1,1], [1,1,0,0]$ $[1,1,0,1], [1,1,1,0], [1,1,1,1]$
$\boldsymbol{k}' = [0,1,1,0]$	$[0,1,0,1], [0,1,1,0], [0,1,1,1], [1,0,0,1], [1,0,1,1], [1,1,0,0]$ $[1,1,0,1], [1,1,1,0], [1,1,1,1]$
$\boldsymbol{k}' = [0,1,1,1]$	$[0,1,1,1], [1,0,1,1], [1,1,1,1]$
$\boldsymbol{k}' = [1,0,0,0]$	$[0,0,0,1], [0,0,1,0], [0,0,1,1], [0,1,0,0], [0,1,0,1], [0,1,1,0],$ $[0,1,1,1], [1,0,0,0], [1,0,0,1], [1,0,1,0], [1,0,1,1], [1,1,0,0],$ $[1,1,0,1], [1,1,1,0], [1,1,1,1]$
$\boldsymbol{k}' = [1,0,0,1]$	$[1,0,0,1], [1,0,1,1], [1,1,0,1], [1,1,1,0], [1,1,1,1]$
$\boldsymbol{k}' = [1,0,1,0]$	$[1,0,0,1], [1,0,1,0], [1,0,1,1], [1,1,0,1], [1,1,1,0], [1,1,1,1]$
$\boldsymbol{k}' = [1,0,1,1]$	$[1,0,1,1], [1,1,1,1]$
$\boldsymbol{k}' = [1,1,0,0]$	$[0,1,0,1], [0,1,1,0], [0,1,1,1], [1,0,0,1], [1,0,1,1], [1,1,0,0],$ $[1,1,0,1], [1,1,1,0], [1,1,1,1]$
$\boldsymbol{k}' = [1,1,0,1]$	$[1,0,1,1], [1,1,0,1], [1,1,1,1]$
$\boldsymbol{k}' = [1,1,1,0]$	$[1,1,1,0], [1,1,1,1]$
$\boldsymbol{k}' = [1,1,1,1]$	$[1,1,1,1]$

Table 6: Comparison of Number of Vectors for CipherFour

For Input Division Property [fff7]						
Round	**1**	**2**	**3**	**4**	**5**	**6**
$\#\mid\mathbb{K}\mid^{u}$	2	54	299	251	25	16
$\#\mid\mathbb{K}\mid^{eu}$	2	12	25	23	5	4
For Input Division Property [fffe]						
Round	**1**	**2**	**3**	**4**	**5**	**6**
$\#\mid\mathbb{K}\mid^{u}$	3	15	52	240	97	15
$\#\mid\mathbb{K}\mid^{eu}$	3	6	9	28	13	4

[1] $\#\mid\mathbb{K}\mid^{u}$: The number of vectors in the TABBDP.
[2] $\#\mid\mathbb{K}\mid^{eu}$: The number of vectors after using early reduce technique.

4 Applications of Table-Aided Bit-Based Division Property

We try to use TABBDP to find integral distinguishers for various block ciphers. In this section, we will illustrate our results for various objects including block ciphers and hash functions. Some of the resulting integral distinguishers have been discovered in various former works and some of them are found for the first time. Due to the huge time and memory complexities of utilizing TABBDP, it is difficult to improve the existing integral distinguishers under many circumstances. For RECTANGLE, we improve the former 7-round higher-order integral distinguisher and construct an 8-round higher-order integral distinguisher. We propose a new 6-round integral distinguisher for PRESENT with data complexity 2^{12}. Note that this distinguisher can be extended one round and gives a 7-round higher-order integral distinguisher. For SPONGENT, we find a new 14-round zero-sum distinguisher whose data complexity is same to that of the best one available. It is worth to note that TABBDP is a common method for finding integral distinguishers and can be applied to many bit-oriented block ciphers. Besides, TABBDP can also play a part in finding integral distinguishers for some word-oriented block ciphers, like LBlock [23], TWINE [17]. Although we do not find any new integral distinguishers for these two ciphers, we believe that considering the S-box at the bit level is of great importance even for a word-oriented block cipher.

4.1 Application of TABBDP to PRESENT

PRESENT [4] PRESENT [4] is an ultra-lightweight block cipher designed by Bogdanov *et al.*. PRESENT uses SP-network and consists of 31 rounds. The block size is 64 bits and two key sizes of 80 and 128 bits are allowed. The round

function consists of an XOR operation to introduce a round key K_i for $1 \leq i \leq 32$, where K_{32} is used for post-whitening, a linear bitwise permutation and a non-linear substitution layer. The non-linear layer uses a 4-bit S-box in parallel which is given in Table 7. An illustration of the round function is shown in Fig. 4.

Table 7: PRESENT's S-Box $S_{PRESENT}$ [4]

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

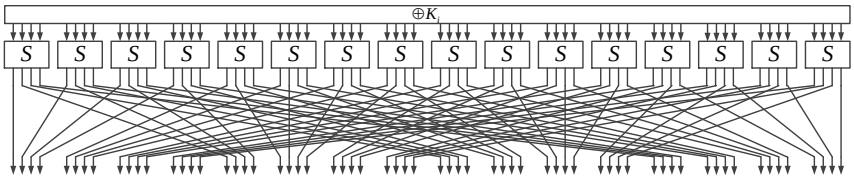


Fig. 4: Round Function of PRESENT.

Since our analysis does not involve subkeys, the key schedule is not involved here. We omit the suffix and simply write PRESENT for simplicity. We also leave behind the key schedule for the remaining block ciphers in this section for the same reason. Please refer to [4] for more details.

Former Results of PRESENT In 2008, Z'aba *et al.* [24] proposed the bit-based integral attack, and found a 3.5-round integral distinguisher. After that, Zhang *et al.* [28] presented a unified algorithm of searching for the best possible higher-order integral distinguishers for block ciphers and found a 5-round integral distinguisher for PRESENT. In 2013, Wu and Wang [22] proposed a 5-round integral distinguisher and a 7-round integral distinguisher for PRESENT by using specific algebraic properties of S-box and special structure of linear layer.

5-Round Integral Distinguisher We utilize TABBDP and find a 5-round integral distinguisher for PRESENT. Let the division property of the input multiset be $\mathcal{D}^{1^{64}}_{\{\texttt{[00000000,0000000f]}\}}$. The distribution of the boundary points for every intermediate state is given Table 8.

Table 8 tells us that if we traverse the four last bits of the plaintext, there is no bit satisfying zero-sum property after six rounds of encryption. Note that there are six elements with hamming weights equal to 2 after five rounds of encryption. These six elements are listed as follows.

Table 8: Distributions of the Boundary Points for PRESENT's Intermediate States with Input Division Property $\mathcal{D}^{1^{64}}_{\{\texttt{[00000000,0000000f]}\}}$

Round	$\#\mid\mathbb{K}\mid$	**Distribution**	
		Hamming Weight	Number
0	1	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}$	1
1	1	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}$	1
2	256	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}$	256
3	65536	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}$	65536
4	2376	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=2\}$	1800
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=3\}$	560
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}$	16
5	66	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=1\}$	60
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=2\}$	6
6	64	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=1\}$	64

`[00000000,00000003], [00000000,00000006],`
`[00000000,00000005], [00000000,0000000a],`
`[00000000,00000009], [00000000,0000000c].`

From the definition of division property, these six vectors show that the four last bits after five rounds of encryption satisfy zero-sum property if the four last bits of the plaintext take every possible value with the remaining sixty bits fixed. Alternatively, we find a 5-round integral distinguisher for PRESENT which is formally expressed by

$$(\mathcal{CCCC},\mathcal{CCCC},\mathcal{CCCC},\mathcal{CCCA}) \xrightarrow{\text{Five Rounds}} (****,****,****,***\mathcal{B}),$$

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

Note that the zero-sum bit of the 5-round integral distinguisher found by Wu and Wang [22] is included among the four zero-sum bits of our 5-round integral distinguisher. The remaining three zero-sum bits are not mentioned in [22] but can be proved by using exactly the same method in [22] where the authors combined the low algebraic degree for the last bit of PRESENT's S-box and PRESENT's specific linear layer. Although our newly found distinguisher has some overlap with the 5-round distinguisher proposed in [22], we can not deny that TABBDP is a general and automated searching method which escapes from exploiting the tanglesome property in the encryption algorithm.

6-Round Integral Distinguisher We also find a 6-round integral distinguisher by using TABBDP. Let the division property of the input multi-set be

$\mathcal{D}^{1^{64}}_{\{\texttt{[00000000,00000fff]}\}}$, *i.e.*, the twelve last bits of the plaintext are traversed. We put all the details about how to find this distinguisher in Appendix B and only show the final results here. The newly found 6-round integral distinguisher for PRESENT is formally expressed by

$$(\mathcal{CCCC}, \mathcal{CCCC}, \mathcal{CCCC}, \mathcal{CAAA}) \xrightarrow{\text{Six Rounds}} (**\,**, **\,**, **\,**, **?),$$

where '?' represents a nibble with the first three bits unknown and the last one zero-sum bit.

Since this distinguisher only traverses twelve bits, we can verify this distinguisher by randomly generating some master keys and testing the zero-sum bit just like what had been done for SIMON32 by Wang *et al.* in [21]. We randomly generate 2^{10} master keys and verify the zero-sum bit. The experimental results show that the last one bit surely satisfies zero-sum property after six rounds of encryption. Note that this 6-round integral distinguisher is proposed for the first time and can be extended to give a 7-round higher-order integral distinguisher with data complexity 2^{48}.

Due to the huge time and memory complexities, we can not find the 7-round integral distinguisher proposed in [22] with TABBDP. Since TABBDP breaks the vectors into bits, the memory complexity of it is apparently larger than that of conventional division property operating on words. The exponential growth in the number of vectors in $\mathbb{K}$ also produces rapid inflation in memory complexity. Additionally, the time used to deal with these vectors also increases. But it is worth to note that TABBDP is a common method for finding integral distinguishers and can be applied to much more bit-oriented block ciphers.

4.2 Application of TABBDP to SPONGENT

SPONGENT [5] SPONGENT [5] 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 9. An illustration of the round function is given in Fig. 5. For more details, please refer to [5].

Table 9: SPONGENT's S-Box $S_{SPONGENT}$ [5]

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

Former Results of SPONGENT In 2012, Dong *et al.* [11] 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

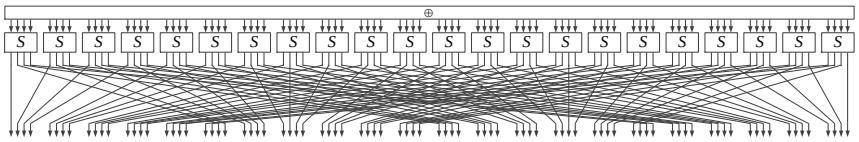


Fig. 5: Round Function of SPONGENT.

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 [12] 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 [11].

14-Round Zero-Sum Distinguisher In this paper, we combine TABBDP and the techniques introduced in [11,12] to find a new zero-sum distinguisher for 14-round SPONGENT-88 in computational complexity 2^{80}. Unlike traditional way of tracking the algebraic degree, we utilize TABBDP to find short zero-sum distinguishers. Since we need to find zero-sum distinguishers in opposite directions, the propagation tables 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 $S^{-1}_{SPONGENT}$. For space limitation, this two tables are given in Appendix C.1.

Due to the huge time and memory complexities, we can not find 6-round zero-sum distinguisher in the forward direction. So we use the zero-sum distinguisher in the forward direction introduced in [12]. In the backward direction, we can find a 4-round zero-sum distinguisher by utilizing TABBDP whose input multi-set satisfies $\mathcal{D}^{1^{88}}_{\{\texttt{[f0000000000,00000000000]}\}}$, *i.e.*, the first four bits for the input of fifth round are traversed[7]. We put all the details about how to find this distinguisher in Appendix C.1 and only give the final results here. By using TABBDP, we known that there are three bits (labeled as 8, 16 and 20, respectively) satisfying zero-sum property after four rounds in the backward direction.

The fifth to eighth round of our 14-round zero-sum distinguisher is illustrated in Fig. 6.

5-Round Integral Distinguisher We also find a 5-round integral distinguisher whose input multi-set satisfies division property $\mathcal{D}^{1^{88}}_{\{\texttt{[00000000000,0000000000f]}\}}$, *i.e.*, we can find some zero-sum bits after five rounds of encryption only if the four last bits of the input are traversed. We put all the related details in Appendix C.3.

[7] Note that we consider zero-sum distinguisher in the backward direction under this situation.

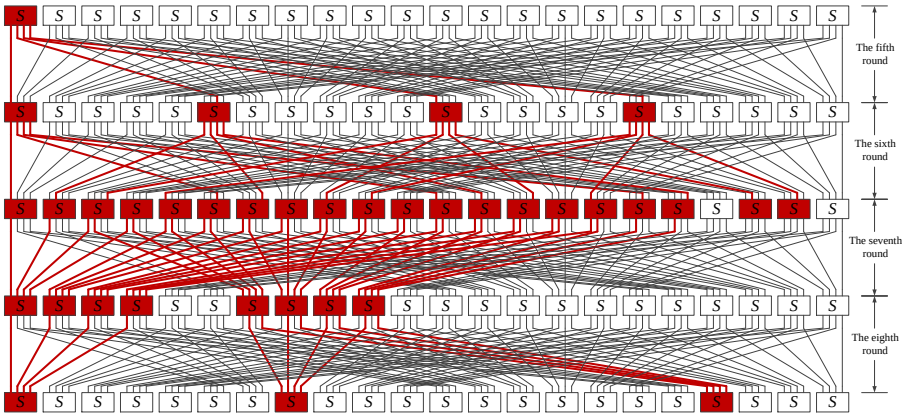


Fig. 6: The Fifth to Eighth Round of Our 14-round Zero-Sum Distinguisher.

4.3 Application of TABBDP to RECTANGLE

RECTANGLE [26] RECTANGLE [26] is a new lightweight block cipher and utilizes SP-network. The design philosophy of RECTANGLE is to allow lightweight and fast implementations using bit-slice technique. The block size is 64 bits, and the key length is 80 or 128 bits. Again, we leave behind the key schedule and do not distinguish these two variants. Each of the 25 rounds consists of three steps: AddRoundKey, SubColumn, ShiftRow. After the last round, there is a final AddRoundKey. The 4-bit S-box used in RECTANGLE is given in Table 10.

Table 10: RECTANGLE's S-Box $S_{RECTANGLE}$ [26]

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

We use a representation similar to the representation of Serpent in [3] to express the 64-bit state of RECTANGLE. The figures describe data blocks by rectangles of 4 rows and 16 columns. Let $\boldsymbol{a} = (a[0], a[1], \ldots, a[63])$ denote a cipher state, the cipher state is described in a two-dimensional way, as illustrated in the following.

$$\begin{bmatrix} a[48] & a[49] & \cdots & a[63] \\ a[32] & a[33] & \cdots & a[47] \\ a[16] & a[17] & \cdots & a[31] \\ a[0] & a[1] & \cdots & a[15] \end{bmatrix}$$

The non-linear operation SubColumn includes 16 parallel applications of S-boxes to the 4 bits in the same column. We illustrate the round function of RECTANGLE in Fig. 7. Please read [26] for more details.

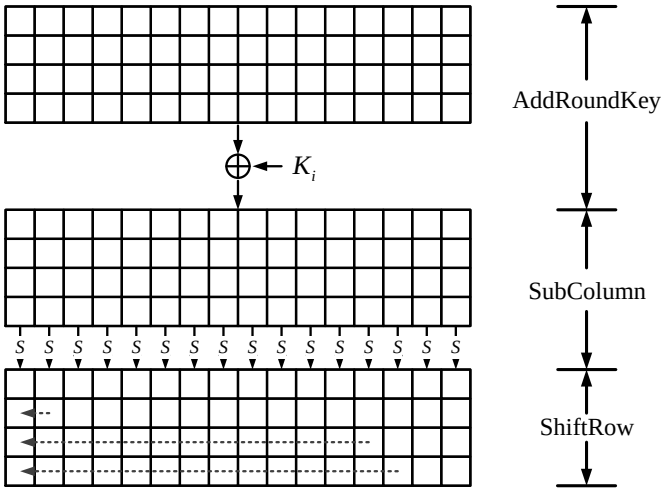


Fig. 7: Round Function of RECTANGLE.

Former Results of RECTANGLE The designers of RECTANGLE gave a 7-round higher-order integral distinguisher in [26]. 2^{48} plaintexts are required in order to observe zero-sum bits after seven rounds of encryption. We propose a new 7-round higher-order integral distinguisher of RECTANGLE here whose data complexity is 2^{36}, which is 2^{12} less than the original result.

7-Round Higher-Order Integral Distinguisher We combine TABBDP with higher-order differential attack. Firstly, we find a 5-round integral distinguisher by using TABBDP whose input multi-set satisfies $\mathcal{D}^{1^{64}}_{\{\texttt{[00010001,00010001]}\}}$. We put all the related details in Appendix D and only give the final results here.

After analysing, we know that there still are 22 bits satisfy zero-sum property after five rounds of encryption, which are illustrated in Fig. 8. The squares dyed in red in Fig. 8 correspond to the bits with zero-sum property. We randomly generate 2^{10} master keys and verify the zero-sum bits. The experimental results show that the bits corresponding to those red squares in Fig. 8 indeed satisfy zero-sum property after five rounds of encryption.

In order to get the 7-round higher-order integral distinguisher, we extend two rounds before the 5-round integral distinguisher. This procedure is illustrated in

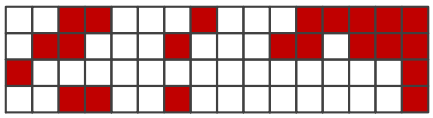


Fig. 8: Zero-Sum Bits after Five Rounds.

Fig. 9. Note that the resulting higher-order integral distinguisher requires 2^{36} plaintexts to observe zero-sum bits after seven rounds of encryption, which is 2^{12} less than the original one.

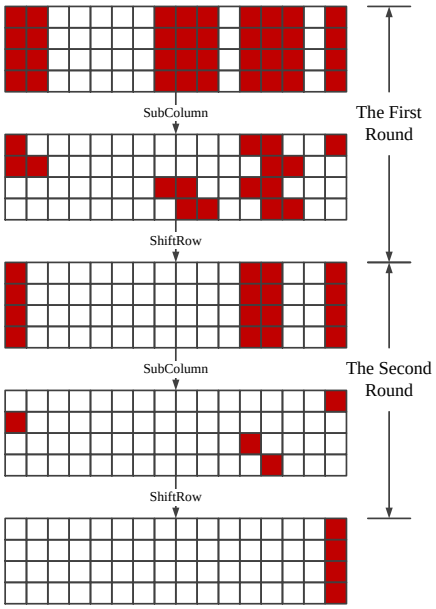


Fig. 9: Extending the 5-Round Integral Distinguisher to a 7-Round Higher-Order Integral Distinguisher.

8-Round Higher-Order Integral Distinguisher Firstly, we find a 6-round integral distinguisher by applying early reduce technique to TABBDP whose input multi-set satisfies $\mathcal{D}_{\{\texttt{[00030003,00030003]}\}}$, *i.e.*, we traverse eight bits located at the given positions. If we do not use early reduce technique, we can find rapid inflation in memory complexity which prevents us from getting this integral distinguisher. We give a comparison for the number of vectors in $\mathbb{K}$ before and after using this technique in Table 11.

Table 11: Comparison of Number of Vectors for RECTANGLE with Input Division Property $\mathcal{D}_{\{\texttt{[00030003,00030003]}\}}$

Round	**1**	**2**	**3**	**4**	**5**	**6**
$\#\mid\mathbb{K}\mid^{u}$	1	2304	260281	-†	-†	-†
$\#\mid\mathbb{K}\mid^{eu}$	1	1668	4207	455	34	63‡

[1] $\#\mid\mathbb{K}\mid^{u}$: The number of vectors in the TABBDP.
[2] $\#\mid\mathbb{K}\mid^{eu}$: The number of vectors after using early reduce technique.
† For the rapid inflation of memory complexity, we can not trace the propagation after 4 rounds of encryption.
‡ Even through early reduce technique does not change the division property achieved through one round function, it may prematurely reduce the division property to a terminate condition just as what has been pointed out in [25]. So we do not use early reduce technique in the last round.

Among the 63 vectors after six rounds of encryption, the hamming weights of 62 vectors are equal to one and the hamming weight of the remaining one is equal to 2. The unique vector with hamming weight equal to 2 is `[00000000,00000003]`, which implies that the 62-nd and 63-rd bit of the state after six rounds of encryption satisfy zero-sum property. We randomly generate 2^{10} master keys and verify the zero-sum bits. The experimental results show that these two bits indeed satisfy zero-sum property after six rounds of encryption.

With this 6-round integral distinguisher, we can construct an 8-round higher-order integral distinguisher. Similar to the procedure of constructing the 7-round higher-order integral distinguisher, we extend two rounds before the 6-round integral distinguisher found above, which is illustrated in Fig. 10. Note that this new higher-order integral distinguisher attains one more round than the existing one.

4.4 Application of TABBDP to TWINE

TWINE [17] TWINE [17] is a 64-bit lightweight block cipher supporting 80 and 128-bit keys. The global structure of TWINE is a variant of Type-2 GFS [27] with 16 4-bit words. The round function of TWINE consists of a nonlinear layer using eight 4-bit S-boxes and a diffusion layer, which permutes the 16 words. The S-box used in TWINE is given in Table 12 and one round of TWINE is illustrated in Fig. 11. Please refer to [17] for more details.

Integral Distinguisher for TWINE TABBDP not only can be used to build integral distinguishers for more bit-oriented block ciphers but also can play a part in finding integral distinguishers for some word-oriented block ciphers. In

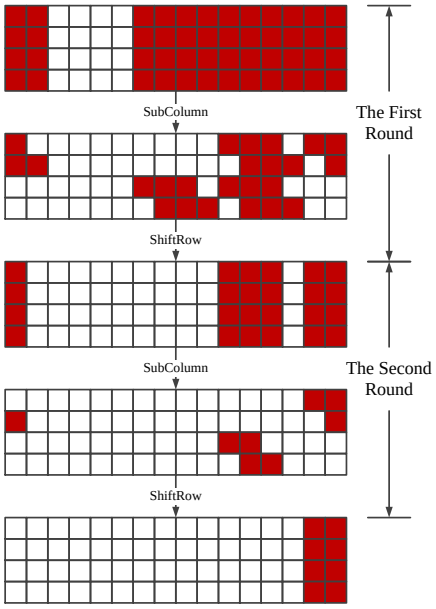


Fig. 10: Extending the 6-Round Integral Distinguisher to an 8-Round Higher-Order Integral Distinguisher.

Table 12: TWINE's S-Box S_{TWINE} [17]

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

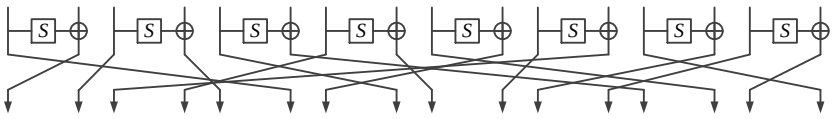


Fig. 11: Round Function of TWINE.

the remaining part of this section, we will illustrate how TABBDP can be applied to find integral distinguishers for TWINE and LBlock.

Note that the global structure of TWINE is a variant of Type-2 GFS and the output nibble of every S-box is XORed with some another branch. An intuitive idea is to transform the 4-bit S-box into an 8-bit S-box (This procedure is illustrated in Fig. 12.) and build propagation table for this 8-bit S-box. In this situation, the propagation table for the enlarged S-box contain 256 rows and can be constructed with the help of a program. The propagation table for TWINE's enlarged S-box is given in Appendix E.1.

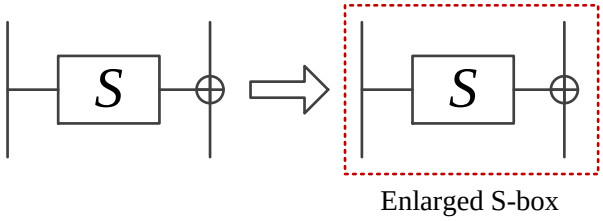


Fig. 12: The Enlarged S-box.

Let the division property of the input multi-set be $\mathcal{D}^{1^{64}}_{\{\texttt{[00000f00,00000000]}\}}$. Then there are 16 bits with zero-sum property after nine rounds of encryption, which is coincidence with the result found by using conventional integral cryptanalysis. The distribution of the boundary points for every intermediate state is given in Table 23 in Appendix E.2. The final 9-round integral distinguisher is of the form

$$(\mathcal{CCCC}, \mathcal{CACC}, \mathcal{CCCC}, \mathcal{CCCC}) \xrightarrow{\text{Nine Rounds}} (*\mathcal{B}*\mathcal{B}, ****, ****, *\mathcal{B}*\mathcal{B}).$$

4.5 Application of TABBDP to LBlock

LBlock [23] LBlock [23] is a lightweight block cipher proposed by Wu and Zhang. The block size is 64 bits and the key size is 80 bits. It employs a variant Feistel structure and consists of 32 rounds. The round function includes an XOR operation to introduce a round key K_i for $1 \leq i \leq 32$, a non-linear substitution layer and a linear wordwise permutation. The non-linear layer uses eight 4-bit S-boxes and we do not list these S-boxes due to the space limitation. One round of LBlock is give in Fig. 13. For a more detailed description, please refer to [23].

Integral Distinguisher for LBlock Note that LBlock employs a variant Feistel structure and the output nibble of every S-box is XORed with some another nibble of the right branch. Similar to the analysis of TWINE, we enlarge every 4-bit S-box of LBlock. After generating the propagation table for every enlarged

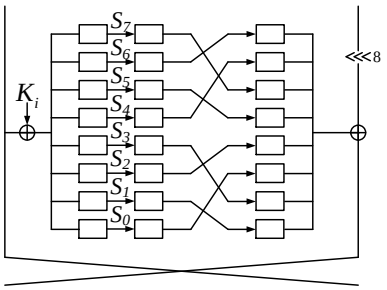


Fig. 13: Round Function of LBlock.

S-box, we can use TABBDP to find integral distinguishers for LBlock[8]. Suppose that the input multi-set satisfies $\mathcal{D}_{\{\texttt{[00000000,00000f00]}\}}$. Then there are 16 bits with zero-sum property after nine rounds of encryptions, which is coincidence with the result found by using conventional integral cryptanalysis. The distribution of the boundary points for every intermediate round is given in Table 24 in Appendix F.

The final 9-round integral distinguisher is of the form

$$(\mathcal{CCCC}, \mathcal{CCCC}, \mathcal{CCCC}, \mathcal{CACC}) \xrightarrow{\text{Nine Rounds}} (**\,**, **\,**, *\mathcal{B}*\mathcal{B}, *\mathcal{B}*\mathcal{B}).$$

Although we do not find any new integral distinguishers for TWINE and LBlock, we believe that considering the S-box at the bit level is of great importance even for a word-oriented block cipher. A more adequate invocation of the details inside an encryption algorithm usually guarantees a better analysis result. And we expect that TABBDP will find better integral distinguisher for some encryption algorithms with relatively weak S-boxes. The analysis at the bit level for a word-oriented block cipher may become a tendency in the future.

5 Future Works and Conclusion

5.1 Future Works

Although TABBDP can be used to find integral distinguishers not only for some bit-oriented block ciphers but also for some word-oriented block ciphers, there still are some further works which can be considered in the following works.

To begin with, note that we do not involve key schedule in our analysis. But the key schedule indeed will have effect on the algebraic degree of the corresponding cipher. And thus, some better results may be attained by integrating the key schedule in the analysis just as what had been done by Todo and Morri for SIMON in [20].

[8] We do not given the propagation tables for the space limitation.

Secondly, the complexity of the linear layer directly influences the utility of TABBDP. We also have attempted to apply TABBDP to some block ciphers with more complex linear layer like AES [10]. Absolutely, we can treat the complex linear layer as the enlarged S-box used in Section 4. But the cost of dealing with this linear layer is intolerably high if the linear layer involves too many bits. If we can handle this problem, we can construct integral distinguishers for more block ciphers and make the TABBDP to be more general.

Last but not least, in the analysis of various block ciphers, we find that the propagation tables for different S-boxes may have apparently distinct looks. So we doubt that the propagation table can reflect the strength of the S-box. Maybe, a new design rationale for S-box against division property following this way can be deduced in the future.

5.2 Conclusion

By integrating the method of bit-based division property with the concrete boolean function expressions of S-box, we come up with a new method to deal with the propagation rule for **Substitution** operation, which can tracing the propagation of division property at the bit level. In order to verify our new method, we apply it to find integral distinguishers for CipherFour. The experimental results show that TABBDP is indeed a valid and efficient tool to search for integral distinguishers for some bit-oriented block ciphers. By combining early reduce technique with our new method, we construct an 8-round higher-order integral distinguisher of RECTANGLE, which attains one more round than the previous one proposed by the designers. For PRESENT and SPONGENT-88, we can find some new distinguishers.

In this paper, we also illustrate that TABBDP can also play a part in finding integral distinguishers for some word-oriented block ciphers, like LBlock and TWINE. Although we do not find any new integral distinguishers for these two ciphers, we believe that considering the S-box at the bit level is of great importance even for a word-oriented block cipher. A more adequate invocation of the details inside an encryption algorithm usually guarantees a better analysis result. And we expect that TABBDP will find better integral distinguisher for some encryption algorithms with relatively weak S-boxes. The analysis at the bit level for a word-oriented block cipher may become a tendency in the future.

References

1. Anderson, R., Biham, E., & Knudsen, L.: Serpent: A Proposal for the Advanced Encryption Standard. NIST AES Proposal, 174, 1-23. (1998)
2. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., & Wingers, L.: The SIMON and SPECK Lightweight Block Ciphers. In Proceedings of the 52nd Annual Design Automation Conference (p. 175). ACM. (2015)
3. Biham, E., Dunkelman, O., & Keller, N.: The Rectangle Attack—Rectangling the Serpent. In Advances in Cryptology—EUROCRYPT 2001 (pp. 340-357). Springer Berlin Heidelberg. (2001)

4. Bogdanov, A., Knudsen, L. R., Leander, G., Paar, C., Poschmann, A., Robshaw, M. J., ... & Vikkelsoe, C.: PRESENT: An Ultra-Lightweight Block Cipher (pp. 450-466). Springer Berlin Heidelberg. (2007)
5. Bogdanov, A., Knezevic, M., Leander, G., Toz, D., Varici, K., & Verbauwhede, I.: SPONGENT: The Design Space of Lightweight Cryptographic Hashing. Computers, IEEE Transactions on, 62(10), 2041-2053. (2013)
6. Boura, C., & Cauteaut, A.: Another View of the Division Property, Dagstuhl Seminar on Symmetric Cryptography 2016.
7. Daemen, J., Knudsen, L., & Rijmen, V.: The Block Cipher Square. In Fast software encryption (pp. 149-165). Springer Berlin Heidelberg. (1997)
8. Daemen, J., Peeters, M., Van Assche, G., & Rijmen, V.: Nessie Proposal: NOEKEON. In First Open NESSIE Workshop (pp. 213-230). (2000)
9. Daemen, J., & Rijmen, V.: AES Proposal: Rijndael. (1999)
10. Daemen, J., & Rijmen, V.: Advanced Encryption Standard (AES)(FIPS 197). Katholijke Universiteit Leuven/ESAT, Tech. Rep. (2001)
11. Dong, L., Wu, W. L., Wu, S., & Zou, J.: Another Look at the Integral Attack by the Higher-Order Differential Attack. Jisuanji Xuebao(Chinese Journal of Computers), 35(9), 1906-1917. (2012)
12. Fan, S., & Duan, M.: Improved Zero-Sum Distinguisher for SPONGENT-88. In International Conference on Electromechanical Control Technology and Transportation-ICECTT 2015. (2015)
13. Knudsen, L. R., & Robshaw, M. J.: Differential Cryptanalysis: The Idea. In The Block Cipher Companion (pp. 109-126). Springer Berlin Heidelberg. (2011)
14. Knudsen, L., & Wagner, D.: Integral Cryptanalysis. In Fast Software Encryption (pp. 112-127). Springer Berlin Heidelberg. (2002)
15. Nyberg, K.: Generalized Feistel Networks. In Advances in Cryptology—ASIACRYPT'96 (pp. 91-104). Springer Berlin Heidelberg. (1996)
16. Sun, B., Hai, X., Zhang, W., Cheng, L., & Yang, Z.: New Observation on Division Property. IACR Cryptology ePrint Archive, 2015, 459. (2015)
17. Suzaki, T., Minematsu, K., Morioka, S., & Kobayashi, E.: TWINE: A Lightweight Block Cipher for Multiple Platforms. In Selected Areas in Cryptography (pp. 339-354). Springer Berlin Heidelberg. (2012)
18. Todo, Y.: Structural Evaluation by Generalized Integral Property. In Advances in Cryptology–EUROCRYPT 2015 (pp. 287-314). Springer Berlin Heidelberg. (2015)
19. Todo, Y.: Integral Cryptanalysis on Full MISTY1. In Advances in Cryptology–CRYPTO 2015 (pp. 413-432). Springer Berlin Heidelberg. (2015)
20. Todo, Y & Morii, M.: Bit-Based Division Property and Application to Simon Family. In Fast Software Encryption (pp.). Springer Berlin Heidelberg. (2016)
21. Wang, Q., Liu, Z., Varıcı, K., Sasaki, Y., Rijmen, V., & Todo, Y.: Cryptanalysis of Reduced-Round SIMON32 and SIMON48. In Progress in Cryptology–INDOCRYPT 2014 (pp. 143-160). Springer International Publishing. (2014)
22. Wu, S., & Wang, M.: Integral Attacks on Reduced-Round PRESENT. In Information and Communications Security (pp. 331-345). Springer International Publishing. (2013)
23. Wu, W., & Zhang, L.: LBlock: A Lightweight Block Cipher. In Applied Cryptography and Network Security (pp. 327-344). Springer Berlin Heidelberg. (2011)
24. Z'aba, M. R., Raddum, H., Henricksen, M., & Dawson, E.: Bit-Pattern Based Integral Attack. In Fast Software Encryption (pp. 363-381). Springer Berlin Heidelberg. (2008)

25. Zhang, H., & Wu, W.: Structural Evaluation for Generalized Feistel Structures and Applications to LBlock and TWINE. In Progress in Cryptology–INDOCRYPT 2015 (pp. 218-237). Springer International Publishing. (2015)
26. Zhang, W., Bao, Z., Lin, D., Rijmen, V., Yang, B., & Verbauwhede, I.: RECTANGLE: A Bit-Slice Lightweight Block Cipher Suitable for Multiple Platforms. Science China Information Sciences, 58(12), 1-15. (2015)
27. Zheng, Y., Matsumoto, T., & Imai, H.: On the Construction of Block Ciphers Provably Secure and Not Relying on Any Unproved Hypotheses. In Advances in Cryptology—CRYPTO'89 Proceedings (pp. 461-480). Springer New York. (1989)
28. Zhang, W., Su, B., Wu, W., Feng, D., & Wu, C.: Extending Higher-Order Integral: An Efficient Unified Algorithm of Constructing Integral Distinguishers for Block Ciphers. In Applied Cryptography and Network Security (pp. 117-134). Springer Berlin Heidelberg. (2012)

A Illustration of Table-Aided Bit-Based Division Property on CipherFour

A.1 CipherFour

CipherFour [13] is a iterated block cipher with r rounds that operates on 16-bit blocks and has been served as experimental subjects in various works such as [13]. Its $r+1$ subkeys are generated randomly. We do not use the original S-box used in CipherFour and use PRESENT's S-box instead. Encryption with CipherFour is illustrated in Fig. 14. Since our main aim is to find the integral distinguishers, the absence of linear layer in the last round has no effect on our analysis.

A.2 Illustration of Table-Aided Bit-Based Division Property on CipherFour

Suppose that the input multi-set follows division property $\mathcal{D}^{1^{16}}_{\{\texttt{[fffe]}\}}$. Using the propagation rule in Table 4, the resulting division property for every intermediate round is presented in Table 13.

Table 13: Division Property for Intermediate State

Round	# | $\mathbb{K}$ |	$\boldsymbol{k} \in \mathbb{K}$	
0	1	$wt(\boldsymbol{k}) = 15$	[fffe]
1	3	$wt(\boldsymbol{k}) = 14$	[efef]
		$wt(\boldsymbol{k}) = 15$	[feff],[fffe]
2	15	$wt(\boldsymbol{k}) = 12$	[5f5f]
		$wt(\boldsymbol{k}) = 13$	[7d7f],[7f7d],[d7df],[dfd7]
		$wt(\boldsymbol{k}) = 14$	[f5ff],[f7fd],[fdf7],[fff5],[bfbf],[efef]
		$wt(\boldsymbol{k}) = 15$	[fbff],[fffb],[feff],[fffe]

Continued on next page

Table 13 – continued from previous page

Round	# | $\mathbb{K}$ |	$\boldsymbol{k} \in \mathbb{K}$	
3	52	$wt(\boldsymbol{k}) = 7$	[15b1],[3591],[9531],[b511],[45e4],[65c4], [c564],[e544],[11f1],[13d1],[31d1],[51b1], [5391],[7191],[1971],[1b51],[3951],[5931], [5b11],[7911],[9171],[9351],[b151],[d131], [d311],[f111],[44f4],[54e4],[46d4],[56c4], [64d4],[74c4],[4c74],[5c64],[4e54],[5e44], [6c54],[7c44],[c474],[d464],[c654],[d644], [e454],[f444]
		$wt(\boldsymbol{k}) = 10$	[abea],[ebaa],[aafa],[baea],[aeba],[beaa] [eaba],[faaa]
4	240	$wt(\boldsymbol{k}) = 4$	[0e01],[0e10],[0f00],[1e00],[002d],[003c], [012c],[102c],[020d],[021c],[030c],[120c], [200d],[201c],[210c],[300c],[0069],[0078], [0168],[1068],[0249],[0258],[0348],[1248], [2049],[2058],[2148],[3048],[0429],[0438], [0528],[1428],[0609],[0618],[0708],[1608], [2409],[2418],[2508],[3408],[4029],[4038], [4128],[5028],[4209],[4218],[4308],[5208], [6009],[6018],[6108],[7008],[0483],[0492], [0582],[1482],[04a1],[04b0],[05a0],[14a0], [0681],[0690],[0780],[1680],[2481],[2490], [2580],[3480],[8403],[8412],[8502],[9402], [8421],[8430],[8520],[9420],[8601],[8610], [8700],[9600],[a401],[a410],[a500],[b400], [0c03],[0c12],[0d02],[1c02],[0c21],[0c30], [0d20],[1c20],[2c01],[2c10],[2d00],[3c00], [0087],[0096],[0186],[1086],[00a5],[00b4], [01a4],[10a4],[0285],[0294],[0384],[1284], [2085],[2094],[2184],[3084],[00c3],[00d2], [01c2],[10c2],[00e1],[00f0],[01e0],[10e0], [02c1],[02d0],[03c0],[12c0],[20c1],[20d0], [21c0],[30c0],[4083],[4092],[4182],[5082], [40a1],[40b0],[41a0],[50a0],[4281],[4290], [4380],[5280],[6081],[6090],[6180],[7080], [0807],[0816],[0906],[1806],[0825],[0834], [0924],[1824],[0a05],[0a14],[0b04],[1a04], [2805],[2814],[2904],[3804],[0843],[0852], [0942],[1842],[0861],[0870],[0960],[1860], [0a41],[0a50],[0b40],[1a40],[2841],[2850], [2940],[3840],[4803],[4812],[4902],[5802], [4821],[4830],[4920],[5820],[4a01],[4a10], [4b00],[5a00],[6801],[6810],[6900],[7800],

Continued on next page

Table 13 – continued from previous page

Round	$\#\mid\mathbb{K}\mid$	$\boldsymbol{k}\in\mathbb{K}$	
			[8007], [8016], [8106], [9006], [8025], [8034], [8124], [9024], [8205], [8214], [8304], [9204], [a005], [a014], [a104], [b004], [8043], [8052], [8142], [9042], [8061], [8070], [8160], [9060], [8241], [8250], [8340], [9240], [a041], [a050], [a140], [b040], [c003], [c012], [c102], [d002], [c021], [c030], [c120], [d020], [c201], [c210], [c300], [d200], [e001], [e010], [e100], [f000]
5	97	$wt(\boldsymbol{k})=2$	[0102], [0104], [0108], [0204], [0012], [1002], [0021], [0030], [0120], [1020], [0201], [0210], [0300], [1200], [2001], [2010], [2100], [3000], [0208], [0014], [1004], [0041], [0050], [0140], [1040], [0401], [0410], [0500], [1400], [4001], [4010], [4100], [5000], [0024], [2004], [0042], [0060], [0240], [2040], [0402], [0420], [0600], [2400], [4002], [4020], [4200], [6000], [0048], [0408], [4008], [0018], [1008], [0028], [2008], [0081], [0090], [0180], [1080], [0801], [0810], [0900], [1800], [8001], [8010], [8100], [9000], [0082], [00a0], [0280], [2080], [0802], [0820], [0a00], [2800], [8002], [8020], [8200], [a000], [0084], [00c0], [0480], [4080], [0804], [0840], [0c00], [4800], [8004], [8040], [8400], [c000]
		$wt(\boldsymbol{k})=3$	[0007], [000d], [000b], [000e]
		$wt(\boldsymbol{k})=4$	[4444], [2222], [8888]
6	15	$wt(\boldsymbol{k})=1$	[0002], [0004], [0008], [0010], [0100], [1000], [0020], [0200], [2000], [0040], [0400], [4000], [0080], [0800], [8000]

From the results in Table 13, we know that if we traverse the first fifteen bits of the plaintext, then all the sixteen bits after five rounds of encryption satisfy zero-sum property. We randomly generate 400 keys and exhaustively search for the integral distinguisher. The test results tell that all the sixteen bits after five rounds of encryption indeed satisfy zero-sum property which is in accordance with our analysis. Besides, the results of exhaustively searching also show that the last one bit after six rounds of encryption also satisfy zero-sum property. Ecstatically, Our analysis tells exactly the same thing. To explain this, we need the following analysis. First, let us consider the following question.

Question 1 *How to use division property to express that there is only one single bit satisfy zero-sum property?*

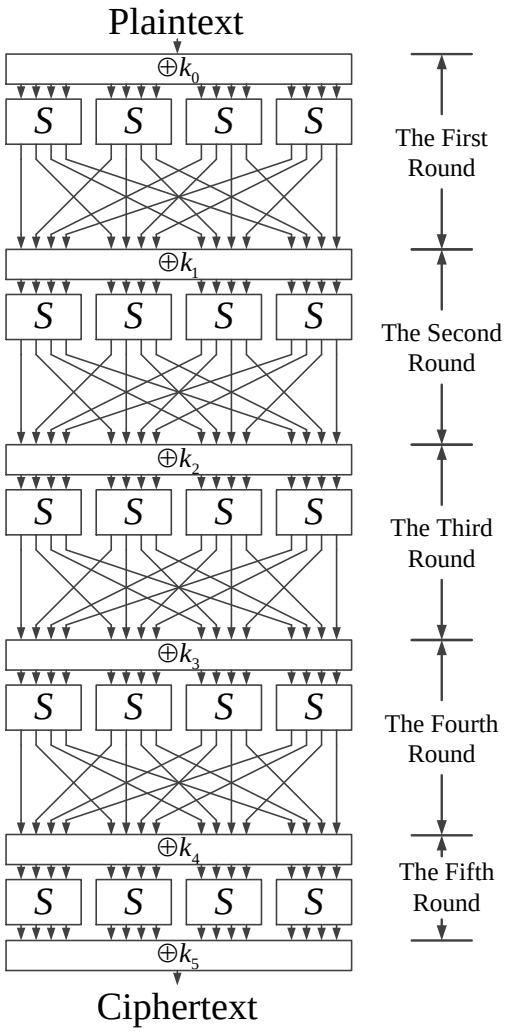


Fig. 14: Encryption with Five Rounds of CipherFour [13].

Any vector with hamming weight larger than 1 can not be qualified for this work. For example, $\boldsymbol{k} = [1, 0, 1, 0]$ indicates that $\bigoplus_{x \in \mathbb{X}} x_1$ and $\bigoplus_{x \in \mathbb{X}} x_3$ are equal to zero when we consider a multi-set $\mathbb{X}$ with division property $\mathcal{D}^{1^4}_{\boldsymbol{k}}$. We find the answer in the last line of Table 13. Note that there are only fifteen elements whose hamming weight are all equal to one in $\mathbb{K}$ after six rounds of encryption, and the absence of `[0001]` shows that the last one bit after six rounds of encryption satisfies zero-sum property.

We also use Todo's conventional division property in [18] to analysis Cipher-Four. For the input multi-set $\mathbb{X}$ with division property $\mathcal{D}^{4^4}_{\{[4,4,4,3]\}}$, we can only find 3-round integral distinguisher. In contract, TABBDP are surely efficient method for finding integral distinguisher.

We change the input division property and the S-box used in the algorithm and repeat the former analysis. All the integral distinguishers found by TABBDP are same to those found by exhaustively search. All these facts convince us that TABBDP is a valid and efficient method to find integral distinguishers for several bit-oriented block ciphers.

B Details of Finding 6-Round Integral Distinguisher for PRESENT

Let the division property of the input multi-set be $\mathcal{D}^{1^{64}}_{\{\texttt{[00000000,00000fff]}\}}$, *i.e.*, the twelve last bits of the plaintext are traversed. The distribution of the boundary points for every intermediate state is given in Table 14.

Table 14: Distributions of the Boundary Points for PRESENT's Intermediate States with Input Division Property $\mathcal{D}^{1^{64}}_{\{\texttt{[00000000,00000fff]}\}}$

Round	$\#\mid\mathbb{K}\mid$	**Distribution**	
		Hamming Weight	Number
0	1	$\#\{\boldsymbol{k} \in \mathbb{K} \mid wt(\boldsymbol{k}) = 12\}$	1
1	1	$\#\{\boldsymbol{k} \in \mathbb{K} \mid wt(\boldsymbol{k}) = 12\}$	1
2	16	$\#\{\boldsymbol{k} \in \mathbb{K} \mid wt(\boldsymbol{k}) = 4\}$	16
3	4096	$\#\{\boldsymbol{k} \in \mathbb{K} \mid wt(\boldsymbol{k}) = 4\}$	4096
4	484	$\#\{\boldsymbol{k} \in \mathbb{K} \mid wt(\boldsymbol{k}) = 2\}$	420
		$\#\{\boldsymbol{k} \in \mathbb{K} \mid wt(\boldsymbol{k}) = 3\}$	56
		$\#\{\boldsymbol{k} \in \mathbb{K} \mid wt(\boldsymbol{k}) = 4\}$	8
5	388	$\#\{\boldsymbol{k} \in \mathbb{K} \mid wt(\boldsymbol{k}) = 1\}$	36
		$\#\{\boldsymbol{k} \in \mathbb{K} \mid wt(\boldsymbol{k}) = 2\}$	348
6	63	$\#\{\boldsymbol{k} \in \mathbb{K} \mid wt(\boldsymbol{k}) = 1\}$	63
7	64	$\#\{\boldsymbol{k} \in \mathbb{K} \mid wt(\boldsymbol{k}) = 1\}$	64

Table 14 tells us that there is no bit satisfying zero-sum property after seven rounds of encryption if we traverse the twelve last bits of the plaintext. Also note that there are only 63 vectors with hamming weight equal to 1 after six rounds of encryption. The absence[9] of `[00000000,00000001]` indicates that the last one bit after six rounds of encryption satisfies zero-sum property if the twelve last bits of the plaintext take all possible values.

C Details of Finding Zero-Sum Distinguishers for SPONGENT

C.1 Propagation Tables for SPONGENT

The propagation tables for $S_{SPONGENT}$ and $S^{-1}_{SPONGENT}$ are are given in Table 15 and Table 16, respectively.

Table 15: Propagation of Division Property for SPONGENT's S-box

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

[9] A similar phenomenon occur when we try to utilize TABBDP to find integral distinguishers for CipherFour. We give a reasonable explanation in Appendix A.

Table 16: Propagation of Division Property for $S^{-1}_{SPONGENT}$

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

C.2 Details of Finding 14-Round Zero-Sum Distinguisher for SPONGENT

In the backward direction, we can find a 4-round zero-sum distinguisher by utilizing TABBDP whose input multi-set satisfies $\mathcal{D}^{1^{88}}_{\{\texttt{[f0000000000,00000000000]}\}}$, *i.e.*, the first four bits for the input of fifth round are traversed. The distribution of the boundary points for every intermediate state is given in Table 17.

Table 17: Distributions of the Boundary Points for SPONGENT's Intermediate States with Input Division Property $\mathcal{D}^{1^{88}}_{\{\texttt{[f0000000000,00000000000]}\}}$ in the Backward Direction

Round	$\#\mid\mathbb{K}\mid$	**Distribution**	
		Hamming Weight	Number
4	1	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}$	1
3	256	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}$	256
2	65536	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}$	65536
1	11114	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=2\}$	2040
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=3\}$	9024
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}$	50
0	88	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=1\}$	85
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=2\}$	3
-1	88	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=1\}$	88

Note that there still are three elements with hamming weight equal to two in the second last line. These three vectors are listed as follows.

```
[00008800000,00000000000],
[00800800000,00000000000],
[00808000000,00000000000].
```

These three vectors show that there are three bits (labeled as 8, 16 and 20, respectively) satisfying zero-sum property after four rounds in the backward direction.

C.3 Details of Finding 5-Round Integral Distinguisher for SPONGENT

Let the input multi-set satisfies division property $\mathcal{D}^{1^{88}}_{\{\texttt{[00000000000,0000000000f]}\}}$, *i.e.*, the four last bits of the input are traversed. The distribution of the boundary points for every intermediate state is given in Table 18.

Table 18 tells us that there exist some bits satisfying zero-sum property after five rounds of encryption. The 72 vectors with hamming weight larger than 1

Table 18: Distributions of the Boundary Points for SPONGENT's Intermediate States with Input Division Property $\mathcal{D}^{1^{88}}_{\{\texttt{[00000000000,0000000000f]}\}}$ in the Forward Direction

Round	$\#$ | $\mathbb{K}$ |	**Distribution**	
		Hamming Weight	Number
0	1	$\lvert\#\{\boldsymbol{k} \in \mathbb{K} \mid wt(\boldsymbol{k}) = 4\}\rvert$	1
1	1	$\lvert\#\{\boldsymbol{k} \in \mathbb{K} \mid wt(\boldsymbol{k}) = 4\}\rvert$	1
2	256	$\lvert\#\{\boldsymbol{k} \in \mathbb{K} \mid wt(\boldsymbol{k}) = 4\}\rvert$	256
3	65536	$\lvert\#\{\boldsymbol{k} \in \mathbb{K} \mid wt(\boldsymbol{k}) = 4\}\rvert$	65536
4	20631	$\lvert\#\{\boldsymbol{k} \in \mathbb{K} \mid wt(\boldsymbol{k}) = 2\}\rvert$	1728
		$\lvert\#\{\boldsymbol{k} \in \mathbb{K} \mid wt(\boldsymbol{k}) = 3\}\rvert$	17608
		$\lvert\#\{\boldsymbol{k} \in \mathbb{K} \mid wt(\boldsymbol{k}) = 4\}\rvert$	1295
5	147	$\lvert\#\{\boldsymbol{k} \in \mathbb{K} \mid wt(\boldsymbol{k}) = 1\}\rvert$	75
		$\lvert\#\{\boldsymbol{k} \in \mathbb{K} \mid wt(\boldsymbol{k}) = 2\}\rvert$	71
		$\lvert\#\{\boldsymbol{k} \in \mathbb{K} \mid wt(\boldsymbol{k}) = 4\}\rvert$	1
6	88	$\lvert\#\{\boldsymbol{k} \in \mathbb{K} \mid wt(\boldsymbol{k}) = 1\}\rvert$	88

are listed in Table 19. With the help of these 72 vectors, we can determine the zero-sum bits after five rounds of encryption.

Table 19: 72 Vectors with Hamming Weight Larger than 1 after Five Rounds of Encryption for Input Division Property $\mathcal{D}^{1^{88}}_{\{\texttt{[00000000000,0000000000f]}\}}$

$\boldsymbol{k} \in \mathbb{K}$ **with Hamming Weight Larger than** 1
[00000000000,00000000011],[00000000008,00000000008],
[00000000000,00000000009],[00000000000,00000000018],
[00000000000,00000000801],[00000000000,00000000808],
[00000000000,00002000800],[00000000008,00000000800],
[00000000000,00002000008],[00000000000,00002000001],
[00000000008,00000000001],[00000000000,00002000010],
[00000000008,00000000010],[00000000000,00000000003],
[00000000000,00000000102],[00000000000,00000800100],
[00000000002,00000000100],[00000800000,00000000100],
[00000000000,00000000012],[00000000000,00000000802],
[00000000000,00000800800],[00000000002,00000000800],
[00000800000,00000000800],[00000000000,00000000810],
[00000000000,0000000000a],[00000000000,00000800010],
[00000000002,00000000010],[00000000000,00000800001],
[00000000002,00000000001],[00000000000,00000800008],
[00000000002,00000000008],[00000000000,00000000804],
Continued on next page

Table 19 – continued from previous page

$k \in \mathbb{K}$ with Hamming Weight Larger than 1
[00000000000,00000000101],[00000000000,00000000900],
[00000000000,00000000108],[00000000000,00002000100],
[00000000008,00000000100],[00000000000,00000000820],
[00000000000,00000000120],[00000800000,00000000001],
[00000000000,00000000021],[00000800000,00000000008],
[00000800000,00000000010],[00000000000,00002000002],
[00000000000,00002800000],[00000000002,00002000000],
[00000800000,00002000000],[00000000008,00000000002],
[00000000008,00000800000],[0000000000a,00000000000],
[00000800008,00000000000],[00000000000,00000000104],
[00000000000,0000000000c],[00000000000,00000000014],
[00000000000,00000000110],[00000000000,00000000024],
[00000000000,00000000030],[00000000000,00000000022],
[00000000000,00000800020],[00000000002,00000000020],
[00000800000,00000000020],[00000000000,00000000028],
[00000000000,00002000020],[00000000008,00000000020],
[00000000000,00000000005],[00000000000,00002000004],
[00000000008,00000000004],[00000000000,00000000006],
[00000000000,00000800004],[00000000002,00000000004],
[00000800000,00000000004],[00000800002,00000800002].

D Details of Finding 5-Round Integral Distinguishers for RECTANGLE

The propagation table for RECTANGLE's S-box is given in Table 20. The distribution of the boundary points for every intermediate state is given in Table 21.

Table 21 tells us that there exist some zero-sum bits after five rounds of encryption if we traverse four bits located at the given positions. After analyzing 212 vectors after five rounds with hamming weight larger than one, there are 22 bits satisfy zero-sum property after five rounds of encryption, which are illustrated in Fig. 8. The squares dyed in red in Fig. 8 correspond to the bits with zero-sum property.

E Details of Finding Integral Distinguisher for TWINE

E.1 Propagation Rule for TWINE's Enlarged S-box

The propagation rule for TWINE's enlarged S-box is given Table 22.

Table 20: Propagation of Division Property for RECTANGLE's S-box

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

Table 21: Distributions of the Boundary Points for RECTANGLE's Intermediate States with Input Division Property $\mathcal{D}^{1^{64}}_{\{\texttt{[00010001,00010001]}\}}$

Round	$\#\mid\mathbb{K}\mid$	**Distribution**	
		Hamming Weight	Number
0	1	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}$	1
1	1	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}$	1
2	256	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}$	256
3	5076	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=2\}$	40
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=3\}$	1118
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}$	4548
4	4162	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=1\}$	8
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=2\}$	373
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=3\}$	2773
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}$	1008
5	254	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=1\}$	42
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=2\}$	207
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=3\}$	5
6	64	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=1\}$	64

Table 22: Propagation of Division Property for TWINE' Enlarged S-box

Input $\mathcal{D}^{1^8}_{\boldsymbol{k}}$	**Output** $\mathcal{D}^{1^8}_{\mathbb{K}'}$
00	00
01	01
02	02
03	03
04	04
05	05
06	06
07	07
08	08
09	09
0a	0a
0b	0b
0c	0c
0d	0d
0e	0e
0f	0f
10	01,02,04,08,10
11	03,05,09,11

Continued on next page

Table 22 – continued from previous page

Input $\mathcal{D}_{\boldsymbol{k}}^{1^8}$	Output $\mathcal{D}_{\mathbb{K}'}^{1^8}$
12	03,06,0a,12
13	07,0b,13
14	05,06,0c,14
15	07,0d,15
16	07,0e,16
17	0f,17
18	09,0a,0c,18
19	0b,0d,19
1a	0b,0e,1a
1b	0f,1b
1c	0d,0e,1c
1d	0f,1d
1e	0f,1e
1f	1f
20	01,02,04,08,20
21	03,05,09,21
22	03,06,0a,22
23	07,0b,23
24	05,06,0c,24
25	07,0d,25
26	07,0e,26
27	0f,27
28	09,0a,0c,28
29	0b,0d,29
2a	0b,0e,2a
2b	0f,2b
2c	0d,0e,2c
2d	0f,2d
2e	0f,2e
2f	2f
30	01,04,08,12,22,30
31	05,09,13,23,31
32	03,06,0a,32
33	07,0b,33
34	05,0c,16,26,34
35	0d,17,27,35
36	07,0e,36
37	0f,37
38	09,0c,1a,2a,38
39	0d,1b,2b,39
3a	0b,0e,3a

Continued on next page

Table 22 – continued from previous page

Input $\mathcal{D}_{\boldsymbol{k}}^{1^8}$	Output $\mathcal{D}_{\mathbb{K}'}^{1^8}$
3b	0f,3b
3c	0d,1e,2e,3c
3d	1f,2f,3d
3e	0f,3e
3f	3f
40	01,02,04,08,40
41	03,05,09,41
42	03,06,0a,42
43	07,0b,43
44	05,06,0c,44
45	07,0d,45
46	07,0e,46
47	0f,47
48	09,0a,0c,48
49	0b,0d,49
4a	0b,0e,4a
4b	0f,4b
4c	0d,0e,4c
4d	0f,4d
4e	0f,4e
4f	4f
50	01,04,08,12,42,50
51	05,09,13,43,51
52	03,06,0a,52
53	07,0b,53
54	05,0c,16,46,54
55	0d,17,47,55
56	07,0e,56
57	0f,57
58	09,0c,1a,4a,58
59	0d,1b,4b,59
5a	0b,0e,5a
5b	0f,5b
5c	0d,1e,4e,5c
5d	1f,4f,5d
5e	0f,5e
5f	5f
60	02,04,08,21,41,60
61	03,05,09,61
62	06,0a,23,43,62
63	07,0b,63

Continued on next page

Table 22 – continued from previous page

Input $\mathcal{D}_{\boldsymbol{k}}^{1^8}$	Output $\mathcal{D}_{\mathbb{K}'}^{1^8}$
64	06,0c,25,45,64
65	07,0d,65
66	0e,27,47,66
67	0f,67
68	0a,0c,29,49,68
69	0b,0d,69
6a	0e,2b,4b,6a
6b	0f,6b
6c	0e,2d,4d,6c
6d	0f,6d
6e	2f,4f,6e
6f	6f
70	04,08,12,21,41,62,70
71	05,09,13,63,71
72	06,0a,23,43,72
73	07,0b,73
74	0c,16,25,45,66,74
75	0d,17,67,75
76	0e,27,47,76
77	0f,77
78	0c,1a,29,49,6a,78
79	0d,1b,6b,79
7a	0e,2b,4b,7a
7b	0f,7b
7c	1e,2d,4d,6e,7c
7d	1f,6f,7d
7e	2f,4f,7e
7f	7f
80	01,02,04,08,80
81	03,05,09,81
82	03,06,0a,82
83	07,0b,83
84	05,06,0c,84
85	07,0d,85
86	07,0e,86
87	0f,87
88	09,0a,0c,88
89	0b,0d,89
8a	0b,0e,8a
8b	0f,8b
8c	0d,0e,8c

Continued on next page

Table 22 – continued from previous page

Input $\mathcal{D}_{\boldsymbol{k}}^{1^8}$	Output $\mathcal{D}_{\mathbb{K}'}^{1^8}$
8d	0f,8d
8e	0f,8e
8f	8f
90	01,02,04,08,90
91	03,05,09,91
92	03,06,0a,92
93	07,0b,93
94	05,06,0c,94
95	07,0d,95
96	07,0e,96
97	0f,97
98	09,0a,0c,98
99	0b,0d,99
9a	0b,0e,9a
9b	0f,9b
9c	0d,0e,9c
9d	0f,9d
9e	0f,9e
9f	9f
a0	02,04,08,21,81,a0
a1	03,05,09,a1
a2	06,0a,23,83,a2
a3	07,0b,a3
a4	06,0c,25,85,a4
a5	07,0d,a5
a6	0e,27,87,a6
a7	0f,a7
a8	0a,0c,29,89,a8
a9	0b,0d,a9
aa	0e,2b,8b,aa
ab	0f,ab
ac	0e,2d,8d,ac
ad	0f,ad
ae	2f,8f,ae
af	af
b0	03,05,08,12,14,21,22,24,81,84,b0
b1	09,13,15,23,25,85,b1
b2	07,0a,16,23,26,83,86,b2
b3	0b,17,27,87,b3
b4	07,0c,16,25,26,85,b4
b5	0d,17,27,b5

Continued on next page

Table 22 – continued from previous page

Input $\mathcal{D}_{\boldsymbol{k}}^{1^8}$	Output $\mathcal{D}_{\mathbb{K}'}^{1^8}$
b6	0e,27,87,b6
b7	0f,b7
b8	0b,0d,1a,1c,29,2a,2c,89,8c,b8
b9	1b,1d,2b,2d,8d,b9
ba	0f,1e,2b,2e,8b,8e,ba
bb	1f,2f,8f,bb
bc	0f,1e,2d,2e,8d,bc
bd	1f,2f,bd
be	2f,8f,be
bf	bf
c0	01,02,04,08,c0
c1	03,05,09,c1
c2	03,06,0a,c2
c3	07,0b,c3
c4	05,06,0c,c4
c5	07,0d,c5
c6	07,0e,c6
c7	0f,c7
c8	09,0a,0c,c8
c9	0b,0d,c9
ca	0b,0e,ca
cb	0f,cb
cc	0d,0e,cc
cd	0f,cd
ce	0f,ce
cf	cf
d0	01,06,12,14,18,42,44,48,84,88,d0
d1	07,13,15,19,43,45,49,85,89,d1
d2	03,16,1a,46,4a,86,8a,d2
d3	17,1b,47,4b,87,8b,d3
d4	05,16,1c,46,4c,8c,d4
d5	17,1d,47,4d,8d,d5
d6	07,1e,4e,8e,d6
d7	1f,4f,8f,d7
d8	09,0e,1a,1c,4a,4c,8c,d8
d9	0f,1b,1d,4b,4d,8d,d9
da	0b,1e,4e,8e,da
db	1f,4f,8f,db
dc	0d,1e,4e,dc
dd	1f,4f,dd
de	0f,de

Continued on next page

Table 22 – continued from previous page

Input $\mathcal{D}_{\boldsymbol{k}}^{1^8}$	Output $\mathcal{D}_{\mathbb{K}'}^{1^8}$
df	df
e0	02,08,21,24,44,84,c1,e0
e1	03,09,25,45,85,e1
e2	0a,23,26,46,86,c3,e2
e3	0b,27,47,87,e3
e4	06,0c,25,c5,e4
e5	07,0d,e5
e6	0e,27,c7,e6
e7	0f,e7
e8	0a,29,2c,4c,8c,c9,e8
e9	0b,2d,4d,8d,e9
ea	2b,2e,4e,8e,cb,ea
eb	2f,4f,8f,eb
ec	0e,2d,cd,ec
ed	0f,ed
ee	2f,cf,ee
ef	ef
f0	0f,12,18,21,26,34,43,45,48,54,62,64,84,88,c1,f0
f1	13,19,27,35,49,55,63,65,85,89,f1
f2	1a,23,36,47,4a,56,66,86,8a,c3,f2
f3	1b,37,4b,57,67,87,8b,f3
f4	16,1c,25,47,4c,66,8c,c5,f4
f5	17,1d,4d,67,8d,f5
f6	1e,27,4e,8e,c7,f6
f7	1f,4f,8f,f7
f8	1a,29,2e,3c,4b,4d,5c,6a,6c,8c,c9,f8
f9	1b,2f,3d,5d,6b,6d,8d,f9
fa	2b,3e,4f,5e,6e,8e,cb,fa
fb	3f,5f,6f,8f,fb
fc	1e,2d,4f,6e,cd,fc
fd	1f,6f,fd
fe	2f,cf,fe
ff	ff

E.2 Details of Finding Integral Distinguisher for TWINE

Let the division property of the input multi-set be $\mathcal{D}^{1^{64}}_{\{\texttt{[00000f00,00000000]}\}}$. The distribution of the boundary points for every intermediate state is given in Table 23.

Table 23 tells us that there is no bit satisfying zero-sum property after ten rounds of encryption if we only traverse four bits of the input. The 120 vectors

Table 23: Distributions of the Boundary Points for TWINE's Intermediate States with Input Division Property $\mathcal{D}^{1^{64}}_{\{\texttt{[00000f00,00000000]}\}}$

Round	$\#\mid\mathbb{K}\mid$	**Distribution**	
		Hamming Weight	Number
0	1	$\lvert\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}\rvert$	1
1	1	$\lvert\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}\rvert$	1
2	16	$\lvert\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=2\}\rvert$	6
		$\lvert\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=3\}\rvert$	8
		$\lvert\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}\rvert$	2
3	47	$\lvert\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=2\}\rvert$	28
		$\lvert\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=3\}\rvert$	16
		$\lvert\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}\rvert$	3
4	157	$\lvert\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=2\}\rvert$	120
		$\lvert\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=3\}\rvert$	32
		$\lvert\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}\rvert$	5
5	442	$\lvert\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=2\}\rvert$	378
		$\lvert\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=3\}\rvert$	56
		$\lvert\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}\rvert$	8
6	1052	$\lvert\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=2\}\rvert$	972
		$\lvert\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=3\}\rvert$	69
		$\lvert\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}\rvert$	11
7	1486	$\lvert\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=1\}\rvert$	4
		$\lvert\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=2\}\rvert$	1440
		$\lvert\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=3\}\rvert$	32
		$\lvert\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}\rvert$	10
8	946	$\lvert\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=1\}\rvert$	20
		$\lvert\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=2\}\rvert$	922
		$\lvert\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}\rvert$	4
9	168	$\lvert\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=1\}\rvert$	48
		$\lvert\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=2\}\rvert$	120
10	64	$\lvert\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=1\}\rvert$	64

of the ninth round with hamming weight equal to 2 give the following 9-round integral distinguisher.

$$(\mathcal{CCCC}, \mathcal{CACC}, \mathcal{CCCC}, \mathcal{CCCC}) \xrightarrow{\text{Nine Rounds}} (*\mathcal{B}*\mathcal{B}, ****, ****, *\mathcal{B}*\mathcal{B}).$$

F Details of Finding Integral Distinguishers for LBlock

Suppose that the input multi-set satisfies $\mathcal{D}_{\{\texttt{[00000000,00000f00]}\}}$. The distribution of the boundary points for every intermediate round is given in Table 24.

Table 24 shows that there is no zero-sum bit after ten rounds of encryption if we only traverse four bits of the input. The 120 vectors of the ninth round with hamming weight equal to 2 give the following 9-round integral distinguisher.

$$(\mathcal{CCCC}, \mathcal{CCCC}, \mathcal{CCCC}, \mathcal{CACC}) \xrightarrow{\text{Nine Rounds}} (****, ****, *\mathcal{B}*\mathcal{B}, *\mathcal{B}*\mathcal{B}).$$

Table 24: Distributions of the Boundary Points for LBlock's Intermediate States with Input Division Property $\mathcal{D}_{\{\texttt{[00000000,00000f00]}\}}$

Round	$\#\mid\mathbb{K}\mid$	**Distribution**	
		Hamming Weight	Number
0	1	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}$	1
1	1	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}$	1
2	21	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=2\}$	3
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=3\}$	9
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}$	9
3	69	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=2\}$	18
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=3\}$	32
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}$	19
4	208	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=2\}$	97
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=3\}$	74
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}$	37
5	552	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=2\}$	333
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=3\}$	152
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}$	66
6	1142	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=2\}$	919
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=3\}$	150
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}$	73
7	1514	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=1\}$	4
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=2\}$	1413
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=3\}$	57
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}$	40
8	946	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=1\}$	20
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=2\}$	918
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=3\}$	4
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=4\}$	4
9	168	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=1\}$	48
		$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=2\}$	120
10	64	$\#\{\boldsymbol{k}\in\mathbb{K}\mid wt(\boldsymbol{k})=1\}$	64