Improved Distinguisher Search Techniques Based on Parity Sets

Xiaofeng Xie[1] and Tian Tian[1]

National Digital Switching System Engineering & Technological Research Center, P.O.Box 407, 62 Kexue Road, Zhengzhou, 450001, China. tiantian_d@126.com

Abstract. Division property is a distinguishing property against block ciphers proposed by Todo at EUROCRYPT 2015. To give a new approach to division property, Christina et al. proposed a new notion called the parity set at CRYPTO 2016. Using parity sets, they successfully took further properties of S-boxes and linear layers into account and found improved distinguishers against PRESENT. However, the time and memory complexities to compute parity sets are expensive. In this paper, we introduce the idea of meet-in-the-middle to the integral distinguisher search along with a variety of techniques to reduce computation complexity. As a result, we obtain a new distinguisher against 9-round PRESENT which has 22 balanced bits.

Keywords: Division property · Parity set · Integral attacks · Meet-in-the-middle · PRESENT.

1 Introduction

Division property was a technique proposed by Todo at EUROCRYPT 2015 to search integral distinguishers against block ciphers [Tod15,KW02]. Todo applied this technique to structural evaluation against both the Feistel and the SPN constructions and attack the full MISTY1 [Tod17]. After that, many improved techniques based on division property were proposed [SWW17,TIHM17,Tod16]. At FSE 2016, Todo and Morii introduced bit-based division property and made it effective to find distinguishers against non-S-box-based ciphers [TM16]. Although this technique found more accurate integral distinguishers, it could not be applied to ciphers whose block length is more than 32 because of its high time and memory complexities. Based on Todo's work, Xiang *et al.* converted the distinguisher search algorithm based on bit-based division property into an MILP problem at ASIACRYPT 2016 [XZBL16]. They used this method to analyze serval lightweight block ciphers and obtained a series of improved results including a 9-round PRESENT distinguisher with only one balanced bit. This distinguisher is the best known distinguishers with respect to the round number.

At CRYPTO 2016, Boura and Christina introduced the parity set to study division property in another view [BC16]. They utilized the parity set to exploit further properties of the PRESENT S-box and PRESENT linear layer, leading to serval improved distinguishers against reduced-round PRESENT with all

the output bits balanced. Since more properties of S-boxes and the linear layer are utilized, parity sets can find more accurate integral characteristics. But the problem is that, though the authors did not point out, it requires higher time and memory complexities than division property does.

Table 1. Integral characteristics against PRESENT

Methods	#Rounds	Number of Balanced bits	Data	Reference
Degree evaluation	7	1	$2^{20.3}$	[WW13]
Parity set	6	64	2^{32}	[BC16]
Parity set	7	64	2^{52}	[BC16]
Parity set	8	64	2^{63}	[BC16]
MILP model	9	1	2^{60}	[XZBL16]
Our method	9	22	2^{63}	Sect. 5

Our work aims at reducing time and memory complexities when using parity sets to search integral distinguishers. To achieve this goal, we revisit the definition of division property and parity set. As a result, we find it is possible to introduce the idea of meet-in-the-middle into the distinguisher search, which permits us to reduce time and memory complexities. Before the proposal of our new search framework, we introduce a new concept which is called the term set and investigate the propagation rules of the term set through different block cipher operations in this paper. The term set of an output bit describes all the terms contained in the ANF(Algebraic normal form) of this output bit. Recall that the vectors contained in the parity set correspond to all the terms whose parity over the input set is odd. Then if none of the term in the ANF of an output bit is contained in the parity set, which means the intersection of the parity set and the term set is empty, then we can make a conclusion that this output bit is balanced. Thus, our idea is dividing the n-round propagation of the parity set into n_1-round propagation of the parity set and $(n - n_1)$-round propagation of the term set and transforming the distinguisher search problem into the comparison of these two sets. Moreover, to improve the efficiency of the involved set comparisons, we propose two useful techniques: size reduction of parity/term sets and multiple comparisons. As illustrations, we perform extensive experiments on PRESENT, and find a 9-round distinguisher with 22 balanced output bits. Table 1 shows the comparison of our distinguisher and previous ones.

The rest of the paper is organized as follows. In Section 2, we review PRESENT, the division property, and the parity set briefly. In Section 3, we introduce the term set, and investigate the propagation rules of it. Section 4 introduces our distinguisher search techniques in detail. Section 5 applies our new techniques to PRESENT. Finally, conclusions are drawn in Section 6.

2 Preliminaries

2.1 Notation

Notation 1 (Hamming Weight) *For* $\mathbf{x} \in \mathbb{F}_2^n$*, denote by* $wt(\mathbf{x})$ *the hamming weight of* $\mathbf{x}$*. The hamming weight of* $\mathbf{x}$ *is the number of 1's appear in the* $\mathbf{x}$*. For* $\mathbf{x} \in \mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \ldots \times \mathbb{F}_2^{n_m}$*, where* $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m), \mathbf{x}_i \in \mathbb{F}_2^{n_i}$ $(0 < i \leq m)$*, denote* $W(\mathbf{x})$

$$W(\mathbf{x}) = (wt(\mathbf{x}_1), wt(\mathbf{x}_2), \ldots, wt(\mathbf{x}_m)) \in \mathbb{Z}^m.$$

Notation 2 (Bit Product Function) *Let* $\mathbf{u}, \mathbf{x} \in \mathbb{F}_2^n$*. Denote*

$$\mathbf{x}^{\mathbf{u}} = \prod_{i=1}^{n} x[i]^{u[i]},$$

and for $\mathbf{u}, \mathbf{x} \in \mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \ldots \times \mathbb{F}_2^{n_m}$*, where* $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$*,* $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m)$*, define bit product function as*

$$\mathbf{x}^{\mathbf{u}} = \prod_{i=1}^{n} x_i^{u_i}.$$

Notation 3 (Comparison between Vectors) *For* $\mathbf{a}, \mathbf{b} \in \mathbb{Z}^m$*, denote* $\mathbf{a} \geq \mathbf{b}$ *if* $a_i \geq b_i$ *for all* $0 < i \leq m$*, and* $\mathbf{a} > \mathbf{b}$ *if* $\mathbf{a} \geq \mathbf{b}$ *but* $\mathbf{a} \neq \mathbf{b}$*. This notion is also suitable for* $\mathbf{a}, \mathbf{b} \in \mathbb{F}_2^n$*. It is obvious that* $\mathbf{a} > \mathbf{b}$ *if and only if* $\mathbf{x}^{\mathbf{a}}$ *is divisible by* $\mathbf{x}^{\mathbf{b}}$*.*

For $\boldsymbol{u} \in \mathbb{F}_2^n$, let us denote

$$Prec(\boldsymbol{u}) = \{\boldsymbol{v} \in \mathbb{F}_2^n : \boldsymbol{v} \leq \boldsymbol{u}\},\ Succ(\boldsymbol{u}) = \{\boldsymbol{v} \in \mathbb{F}_2^n : \boldsymbol{u} \leq \boldsymbol{v}\}.$$

It is worth noticing that the set $Prec(\boldsymbol{u})$ just takes on all the elements satisfying $\boldsymbol{x}^{\boldsymbol{v}} \mid \boldsymbol{x}^{\boldsymbol{u}}$ and $Succ(\boldsymbol{u})$ is the set of elements satisfying $\boldsymbol{x}^{\boldsymbol{u}} \mid \boldsymbol{x}^{\boldsymbol{v}}$.

Notation 4 (Comparison between Sets) *For* A *and* B *be two sets whose elements are in* $\mathbb{F}_2^n$*, let us denote* $A \geq B$ *if there exist* $\mathbf{a} \in A$ *and* $\mathbf{b} \in B$ *with* $\mathbf{a} \geq \mathbf{b}$*, and* $A \ngeq B$ *if none of such couple exists.*

Proposition 1. *Let* A *and* B *be two sets whose elements are in* $\mathbb{F}_2^n$ *with* $A \geq B$*. If there are* $\mathbf{a}_1, \mathbf{a}_2 \in A, \mathbf{b}_1, \mathbf{b}_2 \in B$ *such that* $\mathbf{a}_2 \geq \mathbf{a}_1$ *and* $\mathbf{b}_1 \geq \mathbf{b}_2$*, then* $A \setminus \{\mathbf{a}_1\} \geq B \setminus \{\mathbf{b}_1\}$*.*

Proof. Since $A \geq B$, there exist $\boldsymbol{a} \in A$ and $\boldsymbol{b} \in B$ such that $\boldsymbol{a} \geq \boldsymbol{b}$. If we remove $\boldsymbol{a}_1$ from A and $\boldsymbol{b}_1$ from B, then we still have

$$\begin{cases} \boldsymbol{a}_2 \geq \boldsymbol{b} & \text{if } \ \boldsymbol{a}_1 = \boldsymbol{a}; \\ \boldsymbol{a} \ \geq \boldsymbol{b}_2 & \text{if } \ \boldsymbol{b}_1 = \boldsymbol{b}; \\ \boldsymbol{a}_2 \geq \boldsymbol{b}_2 & \text{if } \ \boldsymbol{a}_1 = \boldsymbol{a} \text{ and } \boldsymbol{b}_1 = \boldsymbol{b}; \\ \boldsymbol{a} \ \geq \boldsymbol{b} & \text{if } \ \boldsymbol{a}_1 \neq \boldsymbol{a} \text{ and } \boldsymbol{b}_1 \neq \boldsymbol{b}. \end{cases}$$

Thus, it can be seen that $A \setminus \{\boldsymbol{a}_1\} \geq B \setminus \{\boldsymbol{b}_1\}$ holds.

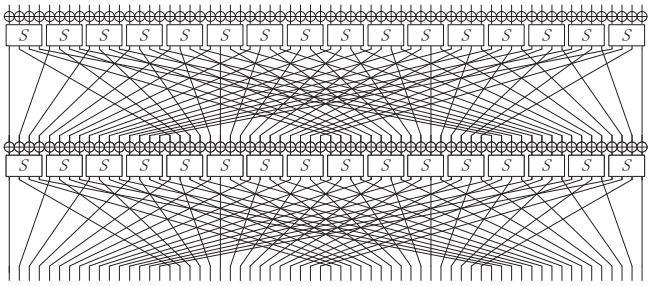

Fig. 1. Round function of the block cipher PRESENT

Notation 5 (Round Function) *Let F be a permutation of $\mathbb{F}_2^n$ defined by*

$$F : \mathbf{x} = (x_1, x_2, \ldots, x_n) \mapsto \mathbf{y} = (y_1, y_2, \ldots, y_n).$$

Then every y_i can be seen as a Boolean function on $x_1, x_2, \ldots, x_n$, denoted by $y_i = F_i(\mathbf{x})$. For a positive integer r, we denote F^r as a composition of r permutation F, and F^{-r} as its inverse permutation.

Remark 1. We denote E^r as a r-round cipher where $r \in \mathbb{Z}$, and E_i^r as the ANF of the i-th output bit.

2.2 PRESENT

PRESENT is an example of SPN construction proposed in 2007 and consists of 31 rounds. The block length is 64 bits and two key lengths of 80 and 128 bits are supported [BKL+07]. Its round function is depicted in Figure 1.

The S-box used in PRESENT is a 4-bit S-box which is a permutation of $\mathbb{F}_2^4$. The ANF is giving by

$$\begin{aligned}
S_1(x_1, x_2, x_3, x_4) &= x_1 \oplus x_3 \oplus x_4 \oplus x_2x_3,\\
S_2(x_1, x_2, x_3, x_4) &= x_2 \oplus x_4 \oplus x_2x_4 \oplus x_3x_4 \oplus x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4,\\
S_3(x_1, x_2, x_3, x_4) &= 1 \oplus x_3 \oplus x_4 \oplus x_1x_2 \oplus x_1x_4 \oplus x_2x_4 \oplus x_1x_2x_4 \oplus x_1x_3x_4,\\
S_4(x_1, x_2, x_3, x_4) &= 1 \oplus x_1 \oplus x_2 \oplus x_4 \oplus x_2x_3 \oplus x_1x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4.
\end{aligned}$$

Observation 1 *The cubic terms in the ANFs of the second and fourth coordinates (say S_2 and S_4) are the same.*

As a result, the xor of these two coordinates

$$S_2 \oplus S_4 = 1 \oplus x_1 \oplus x_2x_3 \oplus x_2x_4 \oplus x_3x_4$$

only has degree 2. Moreover, it can be observed that every term in $S_2 \oplus S_4$ has a multiple in S_2 and S_4 respectively. Based on this observation, we could improve the efficiency of PRESENT distinguisher search in Section 4.

2.3 Division Property and Parity Sets

In the following we will introduce the division property, parity set, and the propagation rules of the parity set.

Definition 1 (Division Property [Tod15]). *Let X be a multiset whose elements belong to $\mathbb{F}_2^n$. Then X is said to have the division property D_k^n when it fulfills the following conditions: For $\mathbf{u} \in \mathbb{F}_2^n$, the parity of $\mathbf{x}^\mathbf{u}$ over all elements in X is always even when $wt(\mathbf{u}) < k$. For further study of division property, please refer to [Tod15] and [SHZ+15] in detail.*

Remark 2. In this definition, X is a multiset, but we focus on sets in this paper.

Definition 2 (Parity Set [BC16]). *Let X be a set whose elements take value of $\mathbb{F}_2^n$. The parity set of X is denoted by $\mathcal{U}(X)$ and defined as follow:*

$$\mathcal{U}(X) = \{\mathbf{u} \in \mathbb{F}_2^n : \bigoplus_{\mathbf{x} \in X} \mathbf{x}^\mathbf{u} = 1\}.$$

Remark 3. If the parity set $\mathcal{U}(X)$ of X is known, then the division property of X is given by D_k^n, where

$$k = \min_{\boldsymbol{u} \in \mathcal{U}(X)} wt(\boldsymbol{u}).$$

By definitions, it can be seen that parity sets always contains more information than division property does. Besides, we shall show that the parity set could exploit further properties of S-layers and linear layers. As a result, the distinguisher search algorithm based on the parity set can find more accurate integral distinguishers generally.

In the following, we choose affine subspaces of $\mathbb{F}_2^n$ as input sets, and review the propagation rules of the parity set through operations of the SPN construction.

Propagation rule 1 (Key Addition) *Let X be a set whose parity set is $\mathcal{U}(X)$. Then*

$$\mathcal{U}(\mathbf{k} + X) \subseteq \bigcup_{\mathbf{u} \in \mathcal{U}(X)} Succ(\mathbf{u})$$

for any $\mathbf{k} \in \mathbb{F}_2^n$.

Propagation rule 2 (S-box) *Let S be a permutation of $\mathbb{F}_2^n$ and X be a set with parity set $\mathcal{U}(X)$. Then the parity set of $S(X)$ satisfies*

$$\mathcal{U}(S(X)) = \{\mathbf{v} \in \mathbb{F}_2^n : \ \mathbf{x}^\mathbf{u} \ \text{ appears in the ANF of } S^\mathbf{v}(\mathbf{x}), \mathbf{u} \in \mathcal{U}(X)\}.$$

In [BC16], the authors defined the set

$$Vs(\boldsymbol{u}) = \{\boldsymbol{v} \in \mathbb{F}_2^n : \ \boldsymbol{x}^{\boldsymbol{u}} \ \text{ appears in the ANF of } S^{\boldsymbol{v}}(\boldsymbol{x})\},$$

and presented the look-up table of $Vs(\boldsymbol{u})$ for the PRESENT S-box, see [BC16, Table 1]. In [BC16, Table 1], all 4-bit words are represented in hexadecimal number, and the rightmost bit of the word corresponds to the least bit.

As for the S-layer, the authors regarded it as the concatenation of several independent S-boxes. The propagation rule can be described as following.

Propagation rule 3 (S-boxes) *Let S be a permutation of $\mathbb{F}_2^{mt}$ which consists of t parallel independent S-boxes over $\mathbb{F}_2^m$: $S(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t) = (S(\mathbf{x}_1), S(\mathbf{x}_2), \ldots, S(\mathbf{x}_t))$. For an input set X contained in $\mathbb{F}_2^{mt}$ whose parity set is $\mathcal{U}(X)$, we have*

$$\mathcal{U}(S(X)) \subseteq \bigcup_{(\mathbf{u}_1,\mathbf{u}_2,\ldots,\mathbf{u}_t)\in\mathcal{U}(X)} Vs_1(\mathbf{u}_1) \times Vs_2(\mathbf{u}_2) \times \ldots \times Vs_t(\mathbf{u}_t).$$

Propagation rule 4 (S-box with Key Addition) *Now consider the propagation where key addition is inserted before the S-boxes. Let $\mathcal{U}(X)$ be the parity set of the input set. Then the parity set after a key addition and a S-box satisfies*

$$\mathcal{U}(S(X+\mathbf{k})) \subseteq \bigcup_{\mathbf{u}\in\mathcal{U}(X)} \bigcup_{\mathbf{v}\in Succ(\mathbf{u})} Vs(\mathbf{v}).$$

In [BC16], the authors defined the set

$$\mathcal{V}_S(\boldsymbol{u}) = \bigcup_{\boldsymbol{v}\in Succ(\boldsymbol{u})} Vs(\boldsymbol{v}).$$

The authors also presented $\mathcal{V}s(\boldsymbol{u})$ for all $\boldsymbol{u} \in \mathbb{F}_2^4$ for the PRESENT S-box, see [BC16, Table 2].

For the proofs of these propagation rules, please refer to [BC16]. The distinguisher search algorithm based on the division property and the search algorithm based on the parity set are similar [TM16,BC16]. In these two algorithms, we should give the initial parity set $\mathcal{U}(X)$(or initial division property) of an input set X first, and then compute the parity set (or division property) after r-round propagation, say $\mathcal{U}(E^r(X))$. If the parity set of outputs does not include all the unit vectors (or the output set does not satisfy all the division property of order 1), a distinguisher is found. But the sets can be very large when the round number is high, which decides time and memory complexities when searching distinguishers. Thus, our work mainly focus on how to improve the search algorithm in respect of time and memory complexities.

3 Term Set and Their Propagation Rules

In this section, we propose a new concept that we call term set and show some propagation rules of term set on SPN.

3.1 Term Set

Definition 3 (Term Set). *Let $f(\mathbf{x})$ be an n-variable Boolean function. The term set of $f(\mathbf{x})$ denoted by $T(f)$, is the subset of $\mathbb{F}_2^n$ defined by*

$$T(f) = \{\mathbf{u} \in \mathbb{F}_2^n : \mathbf{x}^{\mathbf{u}} \text{ appears in the ANF of } f(\mathbf{x})\}.$$

In order to find a distinguisher, we need to compare $T(E_i^r)$ with $\mathcal{U}(E^r(X))$ and verify whether $T(E_i^r) \cap \mathcal{U}(E^r(X))$ is empty. Now the problem is how to calculate $T(E_i^r)$. To solve this problem, we focus on its propagation rules for round functions of iterated ciphers. With propagation rules we can obtain a set A satisfying $T(E_i^r) \subseteq A$, thus $A \cap \mathcal{U}(E^r(X)) = \emptyset$ implies $T(E_i^r) \cap \mathcal{U}(E^r(X)) = \emptyset$.

Actually, the propagation process of the term set corresponds to the calculation process of ANF. To calculate $E_i^r(\boldsymbol{x})$, a simple and direct method is to compute $E_i^{r-1}(\boldsymbol{x})$ firstly. This means calculating $E^j(\boldsymbol{x}) = E^{j-1}(E(\boldsymbol{x}))$ for $0 < j \leq r-1$ recursively. Then calculate $E_i^r(\boldsymbol{x})$ by $E_i^r(\boldsymbol{x}) = E_i(E^{r-1}(\boldsymbol{x}))$. This means we need to store $E^j(X)$ for $i = 1, 2, \ldots, r-1$. However, as j increases the scale of $E^j(X)$ increases dramatically, and we run out of memory soon. Hence, to reduce memory, we propose to calculate $E_i^j(\boldsymbol{x})$ upside-down(refer to Figure 2), which means to calculate $E_i^r(\boldsymbol{x})$ by recursively computing:

$$E_i^j(\boldsymbol{x}) = E_i^{j-1}(E(\boldsymbol{x})),\ (0 < j \leq r).$$

Take 2-round PRESENT for example. To calculate $E_1^2(\boldsymbol{x})$, we first compute

$$E_1^1(\boldsymbol{x}) = x_1 \oplus x_3 \oplus x_4 \oplus x_2 x_3,$$

and then we calculate $E_1^2(\boldsymbol{x})$ by

$$\begin{aligned} E_1^2(\boldsymbol{x}) &= E_1^1(E(\boldsymbol{x})) = E_1(\boldsymbol{x}) \oplus E_3(\boldsymbol{x}) \oplus E_4(\boldsymbol{x}) \oplus E_2(\boldsymbol{x})E_3(\boldsymbol{x}) \\ &= (x_1 \oplus x_3 \oplus x_4 \oplus x_2x_3) \oplus (x_9 \oplus x_{11} \oplus x_{12} \oplus x_{10}x_{11}) \oplus (x_{13} \oplus x_{15} \oplus x_{16} \oplus x_{14}x_{15}) \\ &\quad \oplus (x_5 \oplus x_7 \oplus x_8 \oplus x_6x_7) \cdot (x_9 \oplus x_{11} \oplus x_{12} \oplus x_{10}x_{11}). \end{aligned}$$

This method utilizes $E_i^j(\boldsymbol{x})$ in every round during propagations instead of recording the whole $E^j(\boldsymbol{x})$. Thus, we investigate the propagation rules in this way in this paper.

Since the S-boxes in SPN constructions work independently, the propagation of term sets can also be divided into several parallel S-boxes. Note that we know the ANF of the PRESENT S-box and the PRESENT linear layer, and so we can easily show the propagation of term sets through PRESENT when the general propagation rule is clear. In the subsequent discussions, we take the m-bit S-box S as an example regardless of the position changes of variables during the linear layer.

3.2 Propagation through S-box

Proposition 2. *Let S be an S-box over $\mathbb{F}_2^m$. Denote*

$$Ts(\mathbf{u}) = \{\mathbf{v} \in \mathbb{F}_2^m : \mathbf{x}^{\mathbf{v}} \text{ appears in the ANF of } S^{\mathbf{u}}(\mathbf{x})\}.$$

Then for an m-variable Boolean function f with the term set $T(f)$, we have

$$T(f(S(\mathbf{x}))) \subseteq \bigcup_{\mathbf{u} \in T(f)} Ts(\mathbf{u}).$$

Table 2. Sets $Ts(u)$ for all possible inputs for the S-box of PRESENT.

	$Ts(\boldsymbol{u})$																
	0	1	2	4	8	3	5	9	6	a	c	7	b	d	e	f	
0	x																
1		x		x	x				x								
2			x		x					x	x	x	x	x			
4	x			x	x	x		x		x			x	x			
8	x	x	x		x				x			x	x	x			
3					x	x		x			x	x	x				
5		x				x	x	x		x					x		
9				x		x	x		x	x	x	x		x			
6			x			x		x	x			x	x				
a						x		x	x	x			x	x			
c	x	x	x	x	x	x	x		x		x		x				
7										x			x	x	x		
b						x				x		x			x		
d										x		x	x		x		
e								x		x				x	x		
f										x			x		x	x	

Proof. Since

$$f(S(\boldsymbol{x})) = \bigoplus_{\boldsymbol{u} \in T(f)} S^{\boldsymbol{u}}(\boldsymbol{x}),$$

we have

$$T(f(S(\boldsymbol{x}))) \subseteq \bigcup_{\boldsymbol{u} \in T(f)} \{\boldsymbol{v} \in \mathbb{F}_2^m : \boldsymbol{x}^{\boldsymbol{v}} \text{ appears in the ANF of } S^{\boldsymbol{u}}(\boldsymbol{x})\} = \bigcup_{\boldsymbol{u} \in T(f)} Ts(\boldsymbol{u}).$$

Table 2 presents the $Ts(\boldsymbol{u})$ of the PRESENT S-box where the rows describe all $\boldsymbol{v}$'s included in $Ts(\boldsymbol{u})$ for every input $\boldsymbol{u}$. All 4-bit words are represented in hexadecimal number, and the rightmost bit of the word corresponds to the least bit. Actually, we can make a conclusion from the definitions of $Ts(\boldsymbol{u})$ and $Vs(\boldsymbol{u})$ that Table 2 is the transposition of [BC16, Table 1].

3.3 Propagation through S-boxes

Proposition 3. *Let $\mathcal{S}$ be a permutation of $\mathbb{F}_2^{mt}$ which consists of t parallel independent S-boxes over $\mathbb{F}_2^m$, namely, $\mathcal{S}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t) = (S(\mathbf{x}_1), S(\mathbf{x}_2), \ldots, S(\mathbf{x}_t))$. For an mt-variable Boolean function f with the term set $T(f)$, we have*

$$T(f(\mathcal{S}(\mathbf{x}))) \subseteq \bigcup_{(\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_t) \in T(f(\mathbf{x}))} Ts_1(\mathbf{u}_1) \times Ts_2(\mathbf{u}_2) \times \ldots \times Ts_t(\mathbf{u}_t).$$

Proof. From Subsection 3.1, we know

$$T(f(\mathcal{S}(\boldsymbol{x}))) \subseteq \bigcup_{\boldsymbol{u}=(\boldsymbol{u}_1,\boldsymbol{u}_2,\ldots,\boldsymbol{u}_t)\in T(f)} Ts(\boldsymbol{u}).$$

Now the problem is how to find all $\boldsymbol{v} = (\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_t)$ such that $\boldsymbol{x}^{\boldsymbol{v}}$ contained in $\mathcal{S}^{\boldsymbol{u}}(\boldsymbol{x})$. Since only $S^{\boldsymbol{u}_i}(\boldsymbol{x}_i)$ may contain $\boldsymbol{x}_i^{\boldsymbol{v}_i}$, it follows that $\boldsymbol{v} \in Ts(\boldsymbol{u})$ if and only if $\boldsymbol{v}_i \in Ts_i(\boldsymbol{u})$ for $1 \leq i \leq n$. Therefore, it can be seen that $Ts(\boldsymbol{u}) \subseteq Ts_1(\boldsymbol{u}_1) \times Ts_2(\boldsymbol{u}_2) \times \ldots \times Ts_t(\boldsymbol{u}_t)$.

3.4 Propagation through Key Addition

Proposition 4. *Let f be an n-variable Boolean function with the term set $T(f)$. For any $\mathbf{k} \in \mathbb{F}_2^n$, the term set of $f(\mathbf{k} \oplus \mathbf{x}) = (x_1 \oplus k_1, x_2 \oplus k_2, \ldots, x_n \oplus k_n)$ satisfies*

$$T(f(\mathbf{k} \oplus \mathbf{x})) \subseteq \bigcup_{\mathbf{u}\in T(f)} Prec(\mathbf{u}).$$

Proof. For any $\boldsymbol{k} \in \mathbb{F}_2^n$, let $\boldsymbol{y} = \boldsymbol{k} \oplus \boldsymbol{x}$. We have

$$\boldsymbol{y}^{\boldsymbol{v}} = (\boldsymbol{x} \oplus \boldsymbol{k})^{\boldsymbol{v}} = \bigoplus_{\boldsymbol{u} \leq \boldsymbol{v}} \boldsymbol{x}^{\boldsymbol{u}} \boldsymbol{k}^{\boldsymbol{v} \oplus \boldsymbol{u}}.$$

It follows that

$$T(f(\boldsymbol{k} \oplus \boldsymbol{x})) = \bigcup_{\boldsymbol{u}\in T(f)} \{\boldsymbol{x}^{\boldsymbol{v}} \boldsymbol{k}^{\boldsymbol{v}\oplus\boldsymbol{u}}, \boldsymbol{v} \leq \boldsymbol{u}\} \subseteq \bigcup_{\boldsymbol{u}\in T(f)} Prec(\boldsymbol{u}).$$

This completes the proof.

3.5 Propagation through One Round

Now we consider the round function where the round key is added before the S-boxes in the SPN constructions. Because the propagation of the term set is upside-down, the S-box is always before key addition when calculating term sets(refer to Figure 2). Let f be an n-variable Boolean function. Then the term set after one round encryption can be deduced by Subsections 3.3 and 3.4, i.e.

$$T(f(S(\boldsymbol{x} \oplus \boldsymbol{k}))) \subseteq \bigcup_{\boldsymbol{u}\in T(f)} \bigcup_{\boldsymbol{v}\in Ts(\boldsymbol{u})} Prec(\boldsymbol{v}), \text{ for every } \boldsymbol{k} \in \mathbb{F}_2^n.$$

In the discussions of parity sets and term sets, we consider the round function where the round key is inserted before S-Layer. Thus, in the subsequent discussions, the round function E always implies that there is a round key addition before S-Layer. We can also search distinguishers by term sets only. In detail, if there exists a $\boldsymbol{u} \in \mathbb{F}_2^n$ such that no any elements $\boldsymbol{v}$ contained in $T(E_i^r)$ satisfy $\boldsymbol{v} \geq \boldsymbol{u}$, then a r-round distinguisher whose input set is $Prec(\boldsymbol{u})$ is found. However, the time and memory complexities of this method will be huge as well. Thus, we take advantage of the meet-in-the-middle technique so that the term set and the parity set could be combined to reduce time and memory complexities.

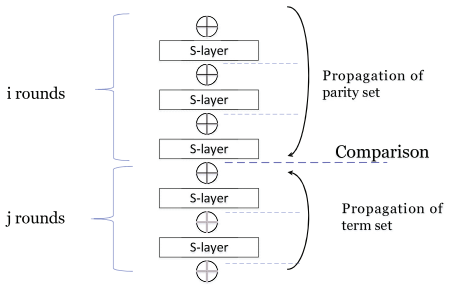

Fig. 2. Outline of our framework against SPN constructions

4 New Searching Techniques for Integral Distinguishers Based on the Parity Set

In this section, we introduce our improved distinguisher search techniques in detail.

4.1 A meet-in-the-middle framework

For an input set X and a round function E, denote the parity set after r-round encryption as $\mathcal{U}(E^r(X))$, i.e., $\mathcal{U}(E^r(X))$ is the set that contains all the $\boldsymbol{u}$'s such that

$$\bigoplus_{\boldsymbol{x}\in E^r(X)} \boldsymbol{x}^{\boldsymbol{u}} = 1.$$

Now we regard the n-bit state in the r-th round as variables $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, and consider the ANF of each output bit. If the ANF of an output bit y_i(about $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$) does not contain any term in $\{\boldsymbol{x}^{\boldsymbol{u}} : \boldsymbol{u} \in \mathcal{U}(E^r(X))\}$, then this output bit is balanced. Based on this observation, we improve the integral distinguisher search by utilizing the meet-in-the-middle technique which divides the n-round propagation of parity sets into n_1-round propagation of parity sets and $(n-n_1)$-round propagation of ANF. An outline of this framework is given in Figure 2. This search framework is similar to the one used in [YHSS16], which is applied to search impossible differential distinguishers. Here we utilize term sets to describe the ANF of output bits and transform the distinguisher search problem into comparison of term sets and parity sets.

Note that it was mentioned in [BC16] that one of the difficulties for finding a distinguisher is that the distinguisher property must hold for every secret key $\boldsymbol{k} \in \mathbb{F}_2^n$. Hence, we use

$$\bigcup_{\boldsymbol{k}\in\mathbb{F}_2^n} T(f(\boldsymbol{k}\oplus\boldsymbol{x})) = \bigcup_{\boldsymbol{u}\in T(f)} Prec(\boldsymbol{u})$$

to propagate term sets from $f(\boldsymbol{x})$ to $f(\boldsymbol{k} \oplus \boldsymbol{x})$. To guarantee the correctness of the algorithm, we utilize

$$\bigcup_{(\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_t) \in T(f)} Ts_1(\boldsymbol{u}_1) \times Ts_2(\boldsymbol{u}_2) \times \ldots \times Ts_t(\boldsymbol{u}_t)$$

to propagate term sets from $f(\boldsymbol{x})$ to $f(S(\boldsymbol{x}))$ when searching distinguishers. Similar propagation model is used on parity sets as well, specifically, we evaluate parity sets after key addition by

$$\mathcal{U}((\boldsymbol{k} \oplus X)) \subseteq \bigcup_{\boldsymbol{k} \in \mathbb{F}_2^n} \mathcal{U}(\boldsymbol{k} \oplus X) = \bigcup_{\boldsymbol{u} \in \mathcal{U}(X)} Succ(\boldsymbol{u})$$

and

$$\mathcal{U}((S(X))) = \bigcup_{(\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_t) \in \mathcal{U}(X)} Vs_1(\boldsymbol{u}_1) \times Vs_2(\boldsymbol{u}_2) \times \ldots \times Vs_t(\boldsymbol{u}_t).$$

Our distinguisher search algorithm consists of four steps which can be described as following.

Step 1 *Round division:* Choose the propagation round numbers of the parity set and the term set respectively. Let us denote r_1 as the propagation round number of parity set and r_2 as the propagation round number of term set, where $r_1 + r_2 = r$.

Step 2 *Choose an input set:* Choose an input set X.

Step 3 *Parity sets calculation:* Calculate the parity set $\mathcal{U}(E^{r_1}(X))$.

Step 4 *Term sets calculation:* Calculate the term sets $T(E_i^{r_2})$ for $1 \leq i \leq n$.

Step 5 *Sets comparison:* Compare $\mathcal{U}(E^{r_1}(X))$ with $T(E_i^{r_2})$ for $1 \leq i \leq n$. If $\mathcal{U}(E^{r_1}(X)) \cap T(E_i^{r_2}) = \emptyset$, then the i-th output bit in r-round encryption is balanced. If none of such intersections are empty, then choose another input set X and go to Step 2.

In order to compare parity sets with term sets efficiently in Step 5, we propose some novel techniques in the following subsections. Firstly, we propose a technique to reduce the size of sets during propagation. Secondly, we introduce the multiple comparison technique which could achieve quick comparisons.

4.2 Reduce the Sizes of Sets

Note that the last operation in the propagation of term sets described in Figure 2 is a key addition. Let $T(f)$ be the term set before the last operation. Then the final term set participating in the comparison step satisfies

$$\bigcup_{\boldsymbol{k} \in \mathbb{F}_2^n} T(f(\boldsymbol{k} \oplus \boldsymbol{x})) = \bigcup_{\boldsymbol{u} \in T(f)} Prec(\boldsymbol{u}).$$

This leads to the following observation.

Observation 2 *Let $\mathcal{U}(E^{r_1}(X))$ be the parity set participating in comparison. Let $T(f)$ be previously defined. Since $T(f(\mathbf{x} \oplus \mathbf{k}))$ is a union of sets of the form $Prec(\mathbf{u})$, if we find $\mathbf{v} \in T(f(\mathbf{x}))$ and $\mathbf{u} \in \mathcal{U}(E^{r_1}(X))$ with $\mathbf{u} \leq \mathbf{v}$, then there must be an element $\mathbf{v}' \in Prec(\mathbf{v}) \subseteq T(f(\mathbf{x} \oplus \mathbf{k}))$ such that $\mathbf{v}' = \mathbf{u}$, which means the output bit may be unbalanced.*

Corollary 1. *Let X be a set of elements in $\mathbb{F}_2^n$ and $T(f)$ be as defined in Observation 2. The i-th output bit of the $(r_2 + r_1)$ round encryption is balanced over the input X if and only if $T(f) \not\geq \mathcal{U}(E^{r_1}(X))$.*

Based on Corollary 1 , the comparison step in our technique against SPN construction is converted into checking whether $T(f) \geq \mathcal{U}(E^{r_1}(X))$. Corollary 1 leads to the following size reduce operation.

Size Reduce Operation. We denote the size reduce operation on a term set $T(E_i^r(\boldsymbol{x}))$ by $R^t(T(E_i^r(\boldsymbol{x})))$, and the size reduce operation on a parity set $\mathcal{U}(E^{r_1}(X))$ by $R^u(\mathcal{U}(E^{r_1}(X)))$. For the term set $T(E_i^r(\boldsymbol{x}))$, the operation R^t removes all the elements $\boldsymbol{v} \in T(E_i^r(x))$ such that there is an element $\boldsymbol{v'} \in T(E_i^r(\boldsymbol{x}))$ with $\boldsymbol{v'} \geq \boldsymbol{v}$. As for a parity set, the operation R^u removes all the elements $\boldsymbol{u} \in \mathcal{U}(E^{r_1}(X))$ such that there is an element $\boldsymbol{u'} \in \mathcal{U}(E^{r_1}(X))$ with $\boldsymbol{u} \geq \boldsymbol{u'}$.

It can be deduced from Proposition 1 that the comparison result of $T(E_i^r(\boldsymbol{x}))$ and $\mathcal{U}(E^{r_1}(X))$ is the same as the comparison result of $R^t(T(E_i^r(\boldsymbol{x})))$ and $R^u(\mathcal{U}(E^{r_1}(X)))$. This guarantees the correctness of the technique using the size reduce operation. It can be seen that by reducing the size of sets, the time complexity of comparison step can be reduced to a great extent. However, during the experiment, we find the complexity of the size reduce operation is still too costly and may even increase the whole complexity of our technique. Therefore, we combine the size reduce technique with propagation rules in the following.

4.3 Propagation Combined with Size Reduction

Notation 6 *For $X \subseteq \mathbb{F}_2^n$, define*

$$Max(X) = \bigcup_{\mathbf{x} \in X} Prec(\mathbf{x}) \quad and \quad Min(X) = \bigcup_{\mathbf{x} \in X} Succ(\mathbf{x}).$$

***Remark**: It is obvious that $Max(R^t(T(f))) = Max(T(f))$ and $Min(\mathcal{U}(X)) = Min(R^u(\mathcal{U}(X))$. Thus, if a term set $T(f)$ is a union of the $Prec(\mathbf{u})$, then we can just store $X = R^t(T(f))$ instead of $T(f)$, since we can recover $T(f)$ by $T(f) = Max(X)$.*

Note that there is always a key addition before S-Layers (see Figure 2), and so the term sets propagated before S-Layers (after key addition) are unions of $Max(\boldsymbol{u})$ according to the propagation rule

$$T(f(\boldsymbol{k} \oplus \boldsymbol{x})) = \bigcup_{\boldsymbol{u} \in T(f)} Prec(\boldsymbol{u}).$$

Table 3. Reduced $T's(\boldsymbol{u})$ for all possible inputs for the PRESENT S-box.

	$R^t(T's(\boldsymbol{u}))$															
	0	1	2	4	8	3	5	9	6	a	c	7	b	d	e	f
0	x															
1		x			x				x							
2												x	x	x		
4													x	x		
8												x	x	x		
3												x	x	x		
5													x	x	x	
9												x	x	x		
6												x	x	x		
a												x	x	x		
c												x	x	x		
7												x	x	x	x	
b												x	x	x	x	
d												x	x	x	x	
e												x	x	x	x	
f																x

As a result, we can only store $R^u(T(f))$ instead of the whole $T(f)$ after key addition to reduce the memory complexity, where each $\boldsymbol{u} \in R^t(T(f))$ represents the set $Max(\boldsymbol{u})$.

Since

$$Max(\boldsymbol{u}) \xrightarrow{S(\boldsymbol{x})} \bigcup_{\boldsymbol{v} \in Prec(\boldsymbol{u})} Ts(\boldsymbol{v}),$$

we define

$$T's(\boldsymbol{u}) = \bigcup_{\boldsymbol{v} \in Prec(\boldsymbol{u})} Ts(\boldsymbol{v}).$$

Then, for any term set $T(f)$, we have

$$T(f) \xrightarrow{\boldsymbol{x}=(\boldsymbol{x} \oplus \boldsymbol{k})} Max(R^t(T(f))) \xrightarrow{\boldsymbol{x}=S(\boldsymbol{x})} \bigcup_{\boldsymbol{v} \in R^t(T(f))} T's(\boldsymbol{v}).$$

Thus, we can apply the size reduce operation to term sets before propagating through key addition. Consequently, we apply the size reduce operation on the term sets after propagating through S-Layer. As mentioned in Subsection 4.2, it is costly to do size reduce operation after S-Layer propagation, we combine the size reduce with S-Layer propagation rule directly, which means the term set through S-layer is evaluated by $R^t(T's(\boldsymbol{u}))$ instead of $T's(\boldsymbol{u})$.

We show $T's(\boldsymbol{u})$ and $R^t(T's(\boldsymbol{u}))$ for all possible inputs for PRESENT S-box in Tables 2 and 3 respectively. Note that the propagation based on $R^t(T's(\boldsymbol{u}))$ only achieves size reduce partly. The effect of reduction depends on the length

Table 4. Reduced $\mathcal{V}s(\boldsymbol{u})$ for all possible inputs for the PRESENT S-box

	$R^u(\mathcal{V}s(\boldsymbol{u}))$																
	0	1	2	4	8	3	5	9	6	a	c	7	b	d	e	f
0	x															
1		x	x	x	x											
2		x	x	x	x											
4		x	x	x	x											
8		x	x	x	x											
3			x	x	x											
5			x	x	x											
9			x	x	x											
6		x	x		x											
a			x	x	x											
c			x	x	x											
7			x		x											
b			x	x	x											
d			x	x	x											
e							x						x		x	
f																x

of blocks. However, in this subsection, we consider a 4-bit S-box as a block. To achieve a further reduction, we propose the further reducing look-up table in Subsection 4.4.

For the parity set, we can only store $R^u(\mathcal{U}(X))$ after every key addition. Let $\mathcal{U}(X)$ be an arbitrary parity set. Then we have

$$Min(\mathcal{U}(X)) \xrightarrow{S(\boldsymbol{x}\oplus\boldsymbol{k})} \bigcup_{\boldsymbol{v}\in R^u(\mathcal{U}(X))} \mathcal{V}s(\boldsymbol{v}).$$

The propagation rule of key addition and the outline in Figure 2 make it suitable to apply the size reduce operation to parity sets propagated after key addition. Similarly, we propose $R^u(\mathcal{V}(\boldsymbol{u}))$ for propagation of parity sets. Table 4 shows the $R^u(\mathcal{V}(\boldsymbol{u}))$ for all possible inputs for PRESENT S-box.

We can roughly estimate that by Tables 3 and 4, the memory complexity can be reduced nearly by a factor of 3^k for every round compared with the original look-up table, where k is the number of active S-box whose input is either 0x0 or 0xf.

4.4 Further Reducing Look-up Table

As mentioned in Subsection 4.3, when combining the size reduce with propagation, the effect of reduction depends on the length of blocks. The effect of reduction would be better as the length of blocks increases. Consequently, we regard a 16-bit super S-box as a block in this subsection to achieve a further reduction.

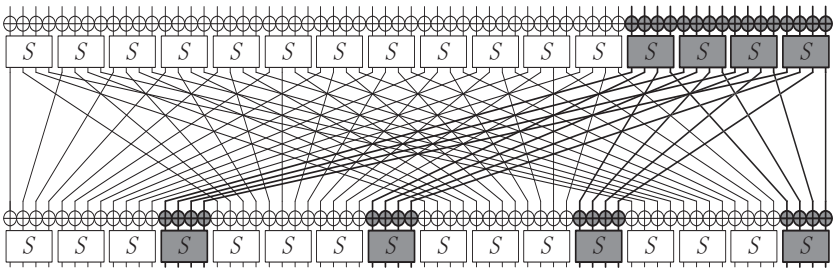

Fig. 3. work model of super S-box in PRESENT

Observation 3 *The super S-boxes in PRESENT can work independently in the* 2*-round encryption. (refer to the Fig 3)*

Based on Observation 3, we can easily construct a 2-round propagation table for the super S-box by calculating

$$\mathcal{U}(\mathcal{S}(P(\mathcal{S}(X))))$$

for all possible inputs, where $\mathcal{S}$ is a permutation of $\mathbb{F}_2^{4n}$, which consists of four PRESENT S-boxes

$$\mathcal{S}(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4) = (S(\boldsymbol{x}_1), S(\boldsymbol{x}_2), S(\boldsymbol{x}_3), S(\boldsymbol{x}_4))$$

and P changes the position of coordinations of elements in parity sets. Next, we reduce the parity set $\mathcal{U}(\mathcal{S}(P(\mathcal{S}(X))))$ for all possible inputs X. Thus, we construct a reduced table for 2-round propagation $R^u\big(\mathcal{U}(\mathcal{S}(P(\mathcal{S}(X)))))\big)$. The reduced table of the term set can be constructed in the same way. There is no doubt that the sizes of sets propagated in this way are smaller than the one propagated by $R^u(V_S'(\boldsymbol{u}))$ and $R^t(T's(\boldsymbol{u}))$. Furthermore, since this propagation table passes through two rounds, the time complexity can also be reduced. The greatest advantage is that it even reduces the time complexity in propagation. Note that this technique requires little memory, but improves the algorithm to a great extent.

4.5 Multiple Comparison

Now we consider how to reduce the time complexity of sets comparison. The simplest way is to compare every couple of $\boldsymbol{u}$,$\boldsymbol{v}$ where $\boldsymbol{u} \in \mathcal{U}(X)$ and $\boldsymbol{v} \in T(\boldsymbol{x})$, and check whether $u \leq v$. But it still requires a high time complexity even when we take the advantage of size reduce technique. Thus, we provide a multiple comparison technique to achieve quick comparisons.

Theorem 1. *Let A and B be two sets whose elements are in $\mathbb{F}_2^n$. If $A \geq B$, then*

$$\max_{\mathbf{u}\in A} wt(\mathbf{u}) \geq \min_{\mathbf{v}\in B} wt(\mathbf{v}).$$

Proof. If $A > B$, then $\boldsymbol{a} > \boldsymbol{b}$ for some $\boldsymbol{a} \in A$ and some $\boldsymbol{b} \in B$. It can be seen that

$$\max_{\boldsymbol{u} \in A} wt(\boldsymbol{u}) > wt(\boldsymbol{a}) > wt(\boldsymbol{b}) > \min_{\boldsymbol{v} \in B} wt(\boldsymbol{v}).$$

If $A = B$, then it is trivial that the result holds.

Theorem 2. *If* $\mathbf{u}, \mathbf{v} \in \mathbb{F}_2^{nt}$ *satisfy* $\mathbf{u} \geq \mathbf{v}$*, then* $W(\mathbf{u}) \geq W(\mathbf{v})$.

Proof. If $\boldsymbol{u} \geq \boldsymbol{v}$, then $u[i] \geq v[i]$ for $0 < i \leq nt$. Thus, we have

$$wt(\boldsymbol{u}_j) = \sum_{k=1}^{n} u[k + n \cdot j] \geq \sum_{k=1}^{n} v[k + n \cdot j] = wt(\boldsymbol{v}_j).$$

Since $W(\boldsymbol{x}) = (wt(\boldsymbol{x}_1), wt(\boldsymbol{x}_2), \ldots, wt(\boldsymbol{x}_m)) \in \mathbb{Z}^m$ and $wt(\boldsymbol{v}_j) \leq wt(\boldsymbol{u}_j)$ for $0 < i \leq nt$, it follows that $W(\boldsymbol{v}) \geq W(\boldsymbol{u})$.

Theorem 3. *If* $\mathbf{u}, \mathbf{v} \in \mathbb{F}_2^n$ *satisfy* $\mathbf{u} > \mathbf{v}$*, then we have* $\mathbf{u} > \mathbf{v}$ *when regard them as integers.*

Proof. Since $v[i] \leq u[i]$ for $0 < i \leq n$ and the inequality holds at least for one i, it can be seen that

$$\sum_{i=1}^{n} u[i] \cdot 2^{i-1} > \sum_{i=1}^{n} v[i] \cdot 2^{i-1}.$$

Based on the above three theorems, we propose the following multiple comparison technique.

The Multiple Comparison Technique. Let $\mathcal{U}$ and T be the parity set and the term set participate in comparison. The multiple comparison technique can be described as follow.

Step 1 Obtain the minimum weight of parity sets

$$m = \min_{\boldsymbol{u} \in \mathcal{U}} wt(\boldsymbol{u})$$

and the maximal weight of term sets

$$M = \max_{\boldsymbol{v} \in T} wt(\boldsymbol{v}).$$

If $M < m$, then we can deduce from Theorem 1 that the output bit is balanced. Otherwise, go to Step 2.

Step 2 Element filtering. Let

$$A = \{\boldsymbol{u} : \boldsymbol{u} \in \mathcal{U} \text{ and } wt(\boldsymbol{u}) > M\}$$

and

$$B = \{\boldsymbol{u} : \boldsymbol{u} \in T \text{ and } wt(\boldsymbol{u}) < m\}.$$

According to Theorem 1, we remove A from $\mathcal{U}$ and remove B from T, e.t. $\mathcal{U} = \mathcal{U}/A$ and $T = T/B$.

Step 3 Further filtering. Take an element $\boldsymbol{t}$ from T, and regard elements in $\mathcal{U}$ and T as values in $F_2^{4\times 16}$. Let us denote

$$C = \{\boldsymbol{u} : \boldsymbol{u} \in \mathcal{U} \text{ and } W(\boldsymbol{u}) < W(\boldsymbol{t})\}.$$

If $C = \varnothing$, then let

$$D = \{\boldsymbol{u} : \boldsymbol{u} \in T \text{ and } W(\boldsymbol{u}) = W(\boldsymbol{t})\}.$$

By Theorem 2, all elements in D cannot be divisible by the elements in $\mathcal{U}$. Thus, we can remove D from T, e.t., $T = T/D$, and go to Step 5.
If $C \neq \varnothing$, then regard elements in $\mathcal{U}$ and D as elements in $F_2^{16\times 4}$. Let

$$C' = \{\boldsymbol{u} : \boldsymbol{u} \in \mathcal{U} \text{ and } W(\boldsymbol{u}) < W(\boldsymbol{t})\} \text{ and } D' = \{\boldsymbol{u} \in D, W(\boldsymbol{u}) = W(\boldsymbol{t})\}.$$

If $C' = \varnothing$, then remove D' from T i.e. $T = (T/D')$. If $C' \neq \varnothing$, then go to Step 4.

Step 4 To judge whether $D' \geq C'$, the elements in C' and D' are regarded as the corresponding integers in this Step. Namely, for $\boldsymbol{u} \in C'$ or $\boldsymbol{u} \in D'$, $\boldsymbol{u}$ is seen as the integer $\sum_{i=0}^{63} u[i] \cdot 2^i$.
Firstly, we sort C' from small to large. For every $\boldsymbol{u} \in D'$, based on Theorem 3, we only need to compare $\boldsymbol{u}$ with elements $\boldsymbol{v} < \boldsymbol{u}, \boldsymbol{v} \in C'$. If $D' \ngeq C'$, then $T = T/D'$, and go to Step 5.

Step 5 If $T \neq \varnothing$, take another $\boldsymbol{t}$ and go to step 3. If $T = \varnothing$ we can stop comparison and make a conclusion that $T \ngeq \mathcal{U}$.

If we find $\boldsymbol{u} \in \mathcal{U}$ and $\boldsymbol{v} \in T$ such that $\boldsymbol{u} < \boldsymbol{v}$ in Step 4, then we can stop comparison and deduce that $T \geq \mathcal{U}$. This technique filters the terms using the divisibility condition under degree order and alphabet respectively. Its effect in time complexity is observable, and we will illustrate this improvement in detail by the application to PRESENT.

5 Application to PRESENT

To illustrate our techniques, we apply our algorithm to PRESENT distinguisher search in this section.

5.1 Search Rule Based on Observation 1

In the PRESENT S-box, for every term in S_1, there exists a multiple in S_2, S_3, and S_4 respectively. Thus, we verify the parity of S_1 first. If S_1 is unbalanced, every output of this S-box is unbalanced too. If S_1 is balanced, then we verify the property of S_3 next. The search rule of S_2 and S_4 is presented as follows. As illustrated in Observation 1, the degree of $S_2 \oplus S_4$ is 2. Hence, $S_2 \oplus S_4$ maybe balanced even if S_2 and S_4 are unbalanced. Moreover, every term of $S_2 \oplus S_4$ has a multiple in S_2 and S_4 respectively, which means the term set of $S_2 \oplus S_4$ is included in the term sets of S_2 and S_4 respectively. Accordingly, we propose the search rule of S_2 and S_4 of the PRESENT S-box in Table 5, which is based on Proposition 5.

Table 5. The search rule of S_2 and S_4 in the PRESENT S-box.

STEP1	RESULT	STEP2	RESULT	FURTHER RESULT
Judge $S_2 \oplus S_4$	balanced	Judge S_2	balanced	S_4 is balanced
			unbalanced	S_4 is unbalanced
	Not balanced			S_2 and S_4 are unbalanced

Proposition 5. *Let X be the input set of encryption E. If $E_1(X) \oplus E_2(X)$ is balanced over all the inputs. Then, the parity of these two output bits over all the inputs are the same.*

Proof. We already know $\bigoplus_{x \in X}(E_1(X) \oplus E_2(X)) = 0$ which means $\bigoplus_{x \in X}(E_1(X)) \oplus \bigoplus_{x \in X} E_2(X) = 0$, thus, the parity of $\bigoplus_{x \in X} E_1(X)$ and $\bigoplus_{x \in X} E_2(X)$ are the same.

5.2 Our Results

First, we try to find 10-round PRESENT distinguishers by our technique, but the search result of the rightmost output bit is unbalanced for all input sets with dimension 63. Since this output bit has the simplest and the lowest degree ANF among 64 output bits, our result shows that the PRESENT probably has no 10-round integral distinguishers by only using the division property. Second, we focus on the 9-round PRESENT, and find a distinguisher with 22 balanced output bits. When searching the 9-round distinguisher against PRESENT, we set the round number r_1 of parity set propagation as 5 and propagation round number r_2 of term set as 4. We take the input:

($aaaaaaaaaaaaaaaa$ $aaaaaaaaaaaaaaaa$ $aaaaaaaaaaaaaaaa$ $aaaaaaaaaaaaaaac$)

as an example. The order of the set $\mathcal{U}(E^{r_1}(X))$ is 320347578. The minimal weight of parity set is 22 and

$$\#\{\boldsymbol{u} \in \mathcal{U}(E^{r_1}(X), wt(\boldsymbol{u}) = 22\} = 59875200.$$

Table 6 shows some datum in the experiments. The effect of the size reduce technique can be observed from the order of the sets. Before applying the size reduce operation, since every row in $Vs(u)$ has nearly 10 elements, the order of parity set should be nearly $10 \times 10^4 \times ((10)^{16})^3 = 10^{53}$. And the effect of the first and second step in multiple comparison is also obvious.
PRESENT's 9-Round Distinguisher
Input:
($aaaaaaaaaaaaaaaa$ $aaaaaaaaaaaaaaaa$ $aaaaaaaaaaaaaaaa$ $aaaaaaaaaaaaaaac$),
Output:
(????????????b_3?bb_3 ????????????b_2?bb_2 ????????????b_1?bb_1 $bbbbbbbbbbbbbbb$),
where 'c' means a constant bit, 'a' means an active bit, '?' means an unknown bit, and 'b' means a balanced bit. Besides, the bits with the same notation b_i means their addition is balanced.

Table 6. Experimental data during distinguisher search against 9-round PRESENT

Term sets	$T(E_1^{r_2})$	$T(E_2^{r_2})$	$T(E_3^{r_2})$	$T(E_4^{r_2})$
#set order	9932	3208722	2680786	3208722
Maximal weight	18	22	22	22
$\#\{v : wt(v) \geq 22\}$	0	38892	29940	38892

6 Conclusions

In this paper, we propose a concept called the term set to propagate some information of ANF. With term sets, we improve the distinguisher search method based on the parity set both in terms of memory and time complexities. From the relation between the parity set and the bit-based division property, it can be seen that the term set could also be applied to improve the distinguisher search method based on bit-based division property similarly. Applying our techniques to other SPN ciphers will be one subject of our future work.

References

[BC16] Christina Boura and Anne Canteaut. Another view of the division property. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 654–682. Springer, 2016.

[BKL+07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: an ultra-lightweight block cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.

[KW02] Lars R. Knudsen and David A. Wagner. Integral cryptanalysis. In Joan Daemen and Vincent Rijmen, editors, *Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002, Revised Papers*, volume 2365 of *Lecture Notes in Computer Science*, pages 112–127. Springer, 2002.

[SHZ+15] Bing Sun, Xin Hai, Wenyu Zhang, Lei Cheng, and Zhichao Yang. New observation on division property. *IACR Cryptology ePrint Archive*, 2015:459, 2015.

[SWW17] Ling Sun, Wei Wang, and Meiqin Wang. Automatic search of bit-based division property for ARX ciphers and word-based division property. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 128–157. Springer, 2017.

[TIHM17] Yosuke Todo, Takanori Isobe, Yonglin Hao, and Willi Meier. Cube attacks on non-blackbox polynomials based on division property. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 250–279. Springer, 2017.

[TM16] Yosuke Todo and Masakatu Morii. Bit-based division property and application to simon family. In Thomas Peyrin, editor, *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, volume 9783 of *Lecture Notes in Computer Science*, pages 357–377. Springer, 2016.

[Tod15] Yosuke Todo. Structural evaluation by generalized integral property. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 287–314. Springer, 2015.

[Tod16] Yosuke Todo. Division property: Efficient method to estimate upper bound of algebraic degree. In Raphael C.-W. Phan and Moti Yung, editors, *Paradigms in Cryptology - Mycrypt 2016. Malicious and Exploratory Cryptology - Second International Conference, Mycrypt 2016, Kuala Lumpur, Malaysia, December 1-2, 2016, Revised Selected Papers*, volume 10311 of *Lecture Notes in Computer Science*, pages 553–571. Springer, 2016.

[Tod17] Yosuke Todo. Integral cryptanalysis on full MISTY1. *J. Cryptology*, 30(3):920–959, 2017.

[WW13] Shengbao Wu and Mingsheng Wang. Integral attacks on reduced-round PRESENT. In Sihan Qing, Jianying Zhou, and Dongmei Liu, editors, *Information and Communications Security - 15th International Conference, ICICS 2013, Beijing, China, November 20-22, 2013. Proceedings*, volume 8233 of *Lecture Notes in Computer Science*, pages 331–345. Springer, 2013.

[XZBL16] Zejun Xiang, Wentao Zhang, Zhenzhen Bao, and Dongdai Lin. Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 648–678, 2016.

[YHSS16] Qianqian Yang, Lei Hu, Siwei Sun, and Ling Song. Extension of meet-in-the-middle technique for truncated differential and its application to roadrunner. In Jiageng Chen, Vincenzo Piuri, Chunhua Su, and Moti Yung, editors, *Network and System Security - 10th International Conference, NSS 2016, Taipei, Taiwan, September 28-30, 2016, Proceedings*, volume 9955 of *Lecture Notes in Computer Science*, pages 398–411. Springer, 2016.