On the Division Property of Simon48 and Simon64*

Zejun Xiang[1,2], Wentao Zhang[1] and Dongdai Lin[1]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{xiangzejun, zhangwentao, ddlin}@iie.ac.cn
[2] University of Chinese Academy of Sciences, Beijing, China

Abstract. Simon is a family of lightweight block ciphers published by the U.S. National Security Agency (NSA) in 2013. Due to its novel and bit-based design, integral cryptanalysis on Simon seems a tough job. At EUROCRYPT 2015 Todo proposed division property which is a generalized integral property, and he applied this technique to searching integral distinguishers of Simon block ciphers by considering the left and right halves of Simon independently. As a result, he found 11-round integral distinguishers for both Simon48 and Simon64. Recently, at FSE 2016 Todo *et al.* proposed bit-based division property that considered each bit independently. This technique can find more accurate distinguishers, however, as pointed out by Todo *et al.* the time and memory complexity is bounded by 2^n for an n-bit block cipher. Thus, bit-based division property is only applicable to Simon32.

In this paper we propose a new technique that achieves a trade-off between considering each bit independently and considering left and right halves as a whole, which is actually a trade-off between time-memory and the accuracy of the distinguishers. We proceed by splitting the state of Simon into small pieces and study the division property propagations of circular shift and bitwise AND operations under the state partition. Moreover, we propose two different state partitions and study the influences of different partitions on the propagation of division property. We find that different partitions greatly impact the division property propagation of circular shift which will finally result in a big difference on the length of integral distinguishers. By using a tailored search algorithm for Simon, we find 12-round integral distinguishers for Simon48 and Simon64 respectively, which improve Todo's results by one round for both variants.

Key words: Simon, division property, integral cryptanalysis

* This paper has been accepted by the 11th International Workshop on Security (IWSEC 2016). This work was supported by the National Natural Science Foundation of China (Grant No. 61379138), the "Strategic Priority Research Program" of the Chinese Academy of Sciences (Grant No. XDA06010701).

1 Introduction

Due to the security requirements on resource constrained devices such as RFID tags, lightweight cryptography has been a very hot topic in cryptographic community in the past decade. This situation has motivated a lot of lightweight block cipher designs, including KATAN/KTANTAN [13], LBlock [21], PRESENT [7], PRIDE [3], PRINCE [8], RECTANGLE [23], SIMON/SPECK [5], to name but a few. SIMON is a family of lightweight block ciphers proposed by Beaulieu *et al.* from the U.S. National Security Agency (NSA) which is tuned for optimal performance in hardware. This family has 10 variants targeting on different security levels with block size varying from 32 to 128 bits and key size varying from 64 to 256 bits.

In this paper, we focus on searching integral distinguishers of SIMON48 and SIMON64. SIMON adopts a Feistel structure with a very compact round function which only uses circular shift, bitwise AND and bitwise XOR operations. However, SIMON was published only with specification, its design criteria and cryptanalysis results were not known to the public. Thus, since its publication, SIMON has received lots of attentions from cryptographic community [1][2][4][6][9][10][11][15][19][20] in terms of a variety of attack techniques. However, from all these cryptanalytic works, it seems that integral cryptanalysis is not sufficient and well-studied for the SIMON family block ciphers.

Integral attack (or square attack [12]) was initially proposed by Daemen *et al.* at FSE 1997 and formalized later by Knudsen [14] at FSE 2002. The core part of integral attack is to construct an integral distinguisher whose outputs (or some bits of the outputs) have the zero-sum property with respect to a set of carefully chosen input texts. One way to construct an integral distinguisher is to study the propagation characteristics of integral property, e.g., the $\mathcal{A}$ property, the $\mathcal{C}$ property, the $\mathcal{B}$ property and the $\mathcal{U}$ property.

At EUROCRYPT 2015, Todo [17] proposed a generalized integral property–division property and studied the propagation characteristics of division property. Furthermore, he applied division property to SPN and Feistel structures and presented some generalized integral distinguishers without knowing the details of the ciphers except for the algebraic degrees of nonlinear components. Later, Todo [16] applied division property to MISTY1 block cipher and successfully broke full-round MISTY1. In the search of integral distinguishers of Fesitel ciphers, Todo treated the cipher state as the left and right two halves, moreover the Feistel round function was viewed as a big Sbox and he only need to know its algebraic degree. In [17] integral distinguishers of SIMON for each state size are presented. Those distinguishers were searched by viewing the round function of SIMON as an Sbox of algebraic degree two.

Very recently, Todo *et al.* introduced bit-based division property in [18] that considered each bit of SIMON32 independently. As a result they found more accurate integral distinguisher for SIMON32 and confirmed the result in [20]. However, as pointed out in [18], bit-based division property was only applicable to SIMON32, since the time and memory complexity for bit-based division prop-

erty is upper bounded by 2^n for an n-bit block cipher. For Simon family with larger block size, this technique is computationally impractical.

1.1 Contributions

If the left and right halves of Simon are considered independently, the integral distinguishers found are too coarse. However, if bit-based division property is considered, it will be computational impractical for Simon family with block size larger than 32 bits. In this paper we study a trade-off between time-memory complexity and accuracy of the distinguishers. We split the state into small pieces and investigate the division property propagations of each operation used in Simon round function. We propose two state partitions and study the corresponding division property propagations. Moreover, we compare these two approaches from both theoretic and experimental aspects. Based on the propagation characteristics of each Simon operation, a search algorithm for integral distinguishers is presented, and our experimental results show that we can derive 12-round integral distinguishers for both Simon48 and Simon64 which improve the results in [17] by one round for both variants. Moreover, we note that our approach is a unified approach, the method with treating each bit independently and the method with treating the left and right halves independently are actually the two limit cases of our state partition.

There are two main contributions in this paper:

(1). We propose two different state partitions to present division property of the state of Simon. We find that different state partitions have a big impact on the division property propagation with respect to circular shift operation. According to both theoretic and experimental results we illustrate that assigning bits with some fixed interval t (that is the i-th, $(i+t)$-th, $(i+2t)$-th $\cdots$ bits, with $t \mid n$ and the addition is modular n addition) into a piece is much better than assigning consecutive bits into a piece.

(2). We split the state of Simon into as many pieces as possible to get a more detailed description of division property of the state, and we study the division property propagations of circular shift and bitwise AND operations which are used in Simon round function. Moreover, we apply these properties in a search algorithm. As a result, we find 12-round integral distinguishers for both Simon48 and Simon64, which is better than Todo's results.

The rest of the paper is organized as follows. Section 2 will present a brief review of Simon and recall the division property. Section 3 will give two kinds of state partitions and study division property propagations against circular shift and bitwise AND operations. Moreover, our search algorithm and experimental results for Simon48 and Simon64 will be presented in this section. At last Sect.4 comes the summary.

2 Preliminaries

In this section, we first give the description of Simon family block ciphers, and then we briefly recall division property.

2.1 Description of SIMON

SIMON is a family of Feistel structured lightweight block ciphers with 10 variants targeting on different block sizes and key sizes. The SIMON block cipher with an n-bit word is denoted as SIMON$2n$, which means the block size is $2n$ bits.

The round function used in the Feistel structure of SIMON block ciphers consists of circular shift, bitwise AND and bitwise XOR operations. The structure of one round SIMON encryption is depicted in Fig.1, where S^i represents a left circular shift by i bits, X_i and Y_i are n-bit words and k_i is the round key. Since the key schedule has no relevance in this paper, we omit the key schedule and refer the readers to [5] for more details.

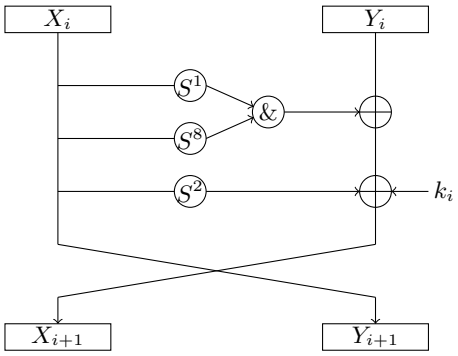

Fig. 1. Feistel Structure of SIMON Round Function

2.2 Division Property

For clarity we first introduce some notations. Denote $\mathbb{F}_2$ the finite field with only two elements, and $\mathbb{F}_2^n$ the n dimensional vector space over $\mathbb{F}_2$. For any $x \in \mathbb{F}_2^n$, the i-th element of x is denoted by $x[i]$ and the Hamming weight of x is calculated as follows:

$$w(x) = \sum_{i=0}^{n-1} x[i] \ .$$

Moreover, let $(\mathbb{F}_2^n)^m$ be the m dimensional vector space whose coordinates belong to $\mathbb{F}_2^n$. Given $\boldsymbol{x} \in (\mathbb{F}_2^n)^m$, the Hamming weight of $\boldsymbol{x} = (x_0, x_1, \cdots, x_{m-1})$ is defined as $W(\boldsymbol{x}) = (w(x_0), w(x_1), \cdots, w(x_{m-1}))$. Let $\boldsymbol{k} = (k_0, k_1, \cdots, k_{m-1})$ and $\boldsymbol{k^*} = (k_0^*, k_1^*, \cdots, k_{m-1}^*)$ denote two vectors in $\mathbb{Z}^m$ where $\mathbb{Z}^m$ denotes m dimensional vector space over all integers $\mathbb{Z}$. Define $\boldsymbol{k} \succeq \boldsymbol{k^*}$ if $k_i \geq k_i^*$ holds for all $i = 0, 1, \cdots, m-1$. Otherwise we write $\boldsymbol{k} \not\succeq \boldsymbol{k^*}$.

Bit Product Function $\pi_u(x)$ and $\boldsymbol{\pi_u}(\boldsymbol{x})$: For any $u \in \mathbb{F}_2^n$, let $\pi_u(x)$ be a function from $\mathbb{F}_2^n$ to $\mathbb{F}_2$. For any $x \in \mathbb{F}_2^n$, define $\pi_u(x)$ as follows:

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

Let $\boldsymbol{\pi_u}(\boldsymbol{x})$ be a function from $(\mathbb{F}_2^{n_0} \times \mathbb{F}_2^{n_1} \times \cdots \times \mathbb{F}_2^{n_{m-1}})$ to $\mathbb{F}_2$ for all $\boldsymbol{u} \in (\mathbb{F}_2^{n_0} \times \mathbb{F}_2^{n_1} \times, \cdots, \times \mathbb{F}_2^{n_{m-1}})$. For any $\boldsymbol{u} = (u_0, u_1, \cdots, u_{m-1}), \boldsymbol{x} = (x_0, x_1, \cdots, x_{m-1}) \in (\mathbb{F}_2^{n_0} \times \mathbb{F}_2^{n_1} \times, \cdots, \times \mathbb{F}_2^{n_{m-1}})$, define $\boldsymbol{\pi_u}(\boldsymbol{x})$ as follows:

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

Definition 1 (Division Property [16]). *Let $\mathbb{X}$ be a multiset whose elements take a value of $(\mathbb{F}_2^n)^m$, and $\boldsymbol{k}$ is an m-dimensional vector whose i-th element takes a value between 0 and n. When the multiset $\mathbb{X}$ has the division property $\mathcal{D}^{n,m}_{\boldsymbol{k^{(0)}},\boldsymbol{k^{(1)}},\cdots,\boldsymbol{k^{(q-1)}}}$, it fulfills the following conditions: The parity of $\pi_{\boldsymbol{u}}(\boldsymbol{x})$ over all $\boldsymbol{x} \in \mathbb{X}$ is always even when*

$$\boldsymbol{u} \in \left\{ (u_0, u_1, \cdots, u_{m-1}) \in (\mathbb{F}_2^n)^m | W(\boldsymbol{u}) \not\succeq \boldsymbol{k^{(0)}}, \cdots, W(\boldsymbol{u}) \not\succeq \boldsymbol{k^{(q-1)}} \right\} \ .$$

The definition of division property divides all possible values of $(\mathbb{F}_2^n)^m$ into two sets Γ_0 and $\Gamma_?$. For any $\boldsymbol{u} \in \Gamma_0$, the parity of $\pi_{\boldsymbol{u}}(\boldsymbol{x})$ over all $\boldsymbol{x} \in \mathbb{X}$ is always even, and for any $\boldsymbol{u} \in \Gamma_?$ the parity is unknown. However, we are only interested in Γ_0. If Γ_0 is nonempty and there exits a nonzero vector $\boldsymbol{u} \in \Gamma_0$, then according to the definition we have $\bigoplus_{\boldsymbol{x} \in \mathbb{X}} \boldsymbol{\pi_u}(\boldsymbol{x}) = 0$ which is a zero-sum property. Furthermore, if $\boldsymbol{u}$ has only one nonzero bit, the parity is exactly the XOR-sum of a certain fixed bit whose position is the same as the nonzero bit of $\boldsymbol{u}$. The steps of constructing an integral distinguisher by division property are to choose a set of plaintexts with some given division property, then trace its propagation through $(r+1)$-round encryption such that Γ_0 for the first time doesn't contain any nonzero vector. In this case, we get an r-round integral distinguisher. Todo presented in [16,17] propagation characteristics of division property for some round function operations, we review some propagation characteristics here which will be used later in this paper.

Proposition 1 (Copy [16]). *Denote $\mathbb{X}$ an input multiset, and let $x \in \mathbb{X}$. The copy function creates (y_1, y_2) from x where $y_1 = x, y_2 = x$. Assuming the input multiset has division property $\mathcal{D}^n_k$. Let $\mathbb{Y}$ be the corresponding output multiset set, then $\mathbb{Y}$ has division property $\mathcal{D}^{n,n}_{(0,k),(1,k-1),\cdots,(k,0)}$.*

Proposition 2 (Compression [16]). *Denote $\mathbb{X}$ an input multiset, let $(x_1, x_2) \in \mathbb{X}$ be an input to the compression function and denote the ouput value by y where $y = x_1 \oplus x_2$. Let $\mathbb{Y}$ be the corresponding output multiset. If input multiset $\mathbb{X}$ has division property $\mathcal{D}^{n,n}_{\boldsymbol{k}}$ where $\boldsymbol{k} = (k_1, k_2)$, then the division property of $\mathbb{Y}$ is $\mathcal{D}^n_{k_1+k_2}$.*

3 Improved Division Property on SIMON48 and SIMON64

In this section, we will study improved division property by cutting the state into small subblocks. In [17] Todo treated the state as two pieces, that is, the left half and the right half which is a rather coarse treatment. However, if we consider bit-based division property as in [18], the time and memory complexity for SIMON48 and SIMON64 will be upper bounded by 2^{48} and 2^{64} respectively, which is impractical for current computing and memory capacity. Thus, we consider cutting the state into small pieces such that the time and memory is feasible. At the meantime, since the state has been cut into pieces, we should get some more accurate integral distinguishers.

After the state partition we study the division property propagation of SIMON round function and construct an algorithm to search integral distinguishers of SIMON. We will propose in this section two different partitions and discuss their influences. Furthermore, we found that different partitions on state bits have different impact on the division property propagation of circular shift operation, and thus this will affect the length of integral distinguishers.

3.1 The First Partition of the State

For SIMON$2n$, both the left and the right halves of the state are n bits, we could cut the state into t-bit pieces where t can divide n, denoted as $t \mid n$. In the following, we assume the state of SIMON is split into t-bit pieces with $t \mid n$.

We present a straightforward partition in this subsection. Denote the left half of the input as X, and the right half as Y. Let $X = a_0 a_1 \cdots a_{n-1}$ where n is the word length and $a_i \in \mathbb{F}_2$ $(i = 0, 1, \cdots, n-1)$, moreover, a_0 is the leftmost bit. Similarly, denote $Y = b_0 b_1 \cdots b_{n-1}$ where $b_i \in \mathbb{F}_2$ $(i = 0, 1, \cdots, n-1)$ and b_0 is the leftmost bit. The partition adopted in this subsection is to cut the state every t consecutive bits. Thus, the state can be expressed as follows:

$$\begin{aligned}(X, Y) &\stackrel{cut}{\Longrightarrow} (a_0 \cdots a_{t-1},\ \cdots\ , a_{n-t} \cdots a_{n-1},\ b_0 \cdots b_{t-1},\ \cdots\ , b_{n-t} \cdots b_{n-1}) \\ &\stackrel{def}{=} (x_0, \cdots, x_{\frac{n}{t}-1}, y_0, \cdots, y_{\frac{n}{t}-1})\end{aligned} \tag{1}$$

We can view the state as a vector in $(\mathbb{F}_2^t)^{\frac{2n}{t}}$. In this case, the division property of SIMON is indicated by vectors in $\mathbb{Z}^{\frac{2n}{t}}$ whose coordinates are between 0 and t. We can see from Fig.1 that one round encryption of SIMON involves copy, circular shift, bitwise AND and compression by bitwise XOR operations. Since **Copy** and **Compression** functions have been studied by Todo and we presented them in **Proposition 1** and **Proposition 2**. In the following we will propose three propositions to illustrate the propagation characteristics of division property of bitwise AND and circular shift operations which are used in SIMON round functions and some of the proofs are given in Appendix A.

SIMON round function uses bitwise AND operation to provide nonlinearity. Todo [17] viewed all nonlinear components as an Sbox, thus bitwise AND can

be viewed as an Sbox with algebraic degree two. We give here a more detailed look at division property propagation with respect to bitwise AND operation.

Proposition 3 (Bitwise AND). *Let $\mathbb{X}$ be an input multiset, and let $\boldsymbol{x} = (x_0, \cdots, x_{\frac{n}{t}-1}, x_0^*, \cdots, x_{\frac{n}{t}-1}^*) \in \mathbb{X}$ where x_i and x_i^* belong to $\mathbb{F}_2^t$. The bitwise AND function creates $\boldsymbol{z} = (x_0 \& x_0^*, \cdots, x_{\frac{n}{t}-1} \& x_{\frac{n}{t}-1}^*)$ from $\boldsymbol{x}$, and let $\mathbb{Z}$ denote the output multiset. If the input multiset has division property $\mathcal{D}_{\boldsymbol{k}}^{t,\frac{2n}{t}}$ where $\boldsymbol{k} = (k_0, \cdots, k_{\frac{n}{t}-1}, k_0^*, \cdots, k_{\frac{n}{t}-1}^*)$, then the output multiset has division property $\mathcal{D}_{\hat{\boldsymbol{k}}}^{t,\frac{n}{t}}$,where $\hat{\boldsymbol{k}} = (\hat{k_0}, \cdots, \hat{k_{\frac{n}{t}-1}})$ and $\hat{k_i} = \max\{k_i, k_i^*\}$.*

Actually, in the search algorithm of integral distinguisher we are given an initial set Γ_0 whose vectors result in an even parity, this set Γ_0 will shrink along the encryption procedure and we stop the search algorithm right before it becomes a set only with a zero vector. If we view bitwise AND operation as an Sbox with algebraic degree two, the division property will propagate from $\boldsymbol{k} = (k_0, k_1, \cdots, k_{\frac{n}{t}-1}, k_0^*, k_1^*, \cdots, k_{\frac{n}{t}-1}^*)$ to $\bar{\boldsymbol{k}} = (\bar{k}_0, \bar{k}_1, \cdots, \bar{k}_{\frac{n}{t}-1})$ where $\bar{k}_i = \lceil \frac{k_i + k_i^*}{2} \rceil$.[3] Note that $\bar{k}_i \leq \hat{k}_i$ always holds, and this makes Proposition 3 much better than viewing bitwise AND as an Sbox with algebraic degree two. Since $\hat{\boldsymbol{k}} \succeq \bar{\boldsymbol{k}}$ indicates that $\bar{\boldsymbol{k}}$ will rule out more vectors from the possible values of $\boldsymbol{u}$ for making the parity even, and thus make set Γ_0 smaller.

Another operation adopted in SIMON round function is circular shift which is used to get diffusion across the state. The rotation parameter for SIMON is one, eight and two. We first study division property propagation of circular shift by one bit in terms of the given state partition.

Proposition 4 (Circular Shift by One Bit). *Let $\mathbb{X}$ be the input multiset, and let $\boldsymbol{x} = (x_0, x_1, \cdots, x_{\frac{n}{t}-1}) \in \mathbb{X}$ where x_i belongs to $\mathbb{F}_2^t$ for all $i \in \{0, 1, \cdots, \frac{n}{t} - 1\}$. Denote $x_0 = a_0 a_1 \cdots a_{t-1}, x_1 = a_t a_{t+1} \cdots a_{2t-1}, \cdots, x_{\frac{n}{t}-1} = a_{n-t} a_{n-t+1} \cdots a_{n-1}$ where $a_i \in \mathbb{F}_2$. The circular shift by one bit function $\boldsymbol{x} \lll 1$ creates $\boldsymbol{x^*} = (x_0^*, x_1^*, \cdots, x_{\frac{n}{t}-1}^*)$ where $x_0^* = a_1 a_2 \cdots a_t, x_1^* = a_{t+1} a_{t+2} \cdots a_{2t}, \cdots$, $x_{\frac{n}{t}-1}^* = a_{n-t+1} a_{n-t+2} \cdots a_{n-1} a_0$. Denote the output multiset by $\mathbb{X}^*$, if the input multiset has division property $\mathcal{D}_{\boldsymbol{k}}^{t,\frac{n}{t}}$ with $\boldsymbol{k} = (k_0, k_1, \cdots, k_{\frac{n}{t}-1})$, the division property of $\mathbb{X}^*$ is $\mathcal{D}_{\boldsymbol{k^{(0)}}, \boldsymbol{k^{(1)}}, \ldots, \boldsymbol{k^{(q-1)}}}^{t,\frac{n}{t}}$ where $\boldsymbol{k^{(0)}}, \boldsymbol{k^{(1)}}, \cdots, \boldsymbol{k^{(q-1)}}$ are all the solutions $(k_0^*, k_1^*, \cdots, k_{\frac{n}{t}-1}^*)$ of the following equations:*

$$\begin{cases} k_0^* = k_0 - c_0 + c_1 & 0 \leq k_0^* \leq t, \\ \vdots & \\ k_{\frac{n}{t}-2}^* = k_{\frac{n}{t}-2} - c_{\frac{n}{t}-2} + c_{\frac{n}{t}-1} & 0 \leq k_{\frac{n}{t}-2}^* \leq t, \\ k_{\frac{n}{t}-1}^* = k_{\frac{n}{t}-1} - c_{\frac{n}{t}-1} + c_0 & 0 \leq k_{\frac{n}{t}-1}^* \leq t, \end{cases} \tag{2}$$

where $c_i \in \{0, 1\}$.

[3] This result is according to Rule 1 and Rule 5 in [16].

Since circular shift by one bit will shift the leftmost bit of a piece into another piece, the division property of any two adjacent pieces will interfere with each other. We illustrate this procedure in Fig.2 where the piece size is 2 bits. For the sake of simplicity we give an illustration on SIMON32, however we note that for SIMON48 and SIMON64 this partition is similar.

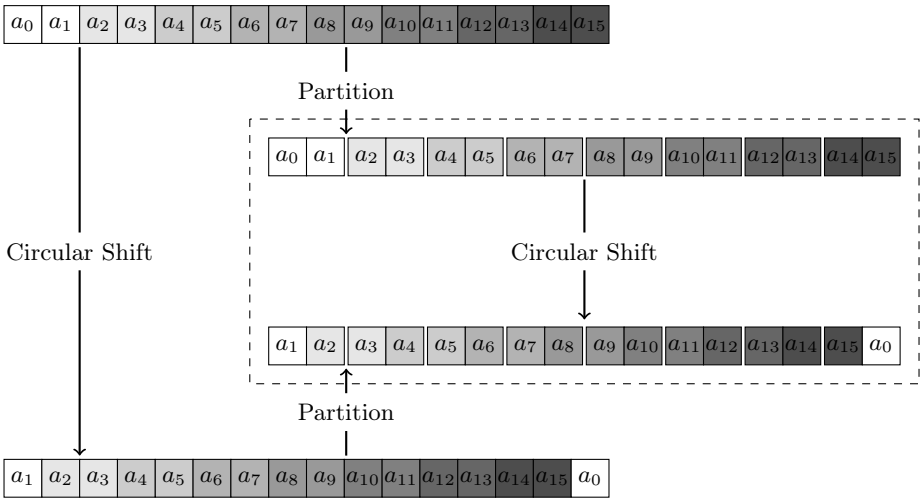

Fig. 2. Left Circular Shift by One Bit Regarding the State Partition

It is clear that left circular shift by i bits just need to apply the proposition i times. However, note that when the shift offset is a multiple of t, the bitwise circular shift is just a piece-wise circular shift. In this situation, division property propagation is a circular shift of the corresponding coordinates of vectors.

Proposition 5 (Circular Shift by One Piece). *Let $\mathbb{X}$ be the input multiset, and let $\boldsymbol{x} = (x_0, x_1, \cdots, x_{\frac{n}{t}-1}) \in \mathbb{X}$ where x_i belongs to $\mathbb{F}_2^t$. The circular shift by one piece creates $\boldsymbol{x^*} = (x_1, x_2, \cdots, x_{\frac{n}{t}-1}, x_0)$. Denote the output multiset by $\mathbb{X}^*$, if the input multiset has division property $\mathcal{D}_{\boldsymbol{k}}^{t,\frac{n}{t}}$ with $\boldsymbol{k} = (k_0, k_1, \cdots, k_{\frac{n}{t}-1})$, the division property of $\mathbb{X}^*$ is $\mathcal{D}_{\boldsymbol{k^*}}^{t,\frac{n}{t}}$ where $\boldsymbol{k^*} = (k_1, k_2, \cdots, k_{\frac{n}{t}-1}, k_0)$.*

Take SIMON48 as an example, we split the state into 4-bit pieces. In this case, circular shift by one bit applies Proposition 4 once, circular shift by eight bits applies Proposition 5 twice and circular shift by two bits applies Proposition 4 twice.

Note that the difference between Proposition 4 and Proposition 5 is Proposition 4 will create approximate (some vectors are invalid when varying the values of each c_i in Equation 2, since the coordinates of each vectors can only take on values from 0 to t) $2^{\frac{n}{t}}$ vectors from one vector while Proposition 5 creates only one vector from a given vector. Furthermore, each vector imposes a constraint

on parity vector $\boldsymbol{u}$, and in general the more vectors, the smaller the size of set Γ_0. In the search algorithm of integral distinguisher we would like to keep Γ_0 shrink as slow as possible along encryption procedure. Thus, we prefer to use Proposition 5. However, the shift offsets of Simon are one, eight and two, and the greatest common divisor is one. If we use only Proposition 5, we need to set the piece size $t = 1$, which corresponds to bit-based division property.

3.2 The Second Partition of the State

In this subsection we present an improved state partition which do not assign consecutive bits into a piece, on the contrary we assign the bits separated by a given interval into a piece as below.

Assume the piece size is t with $t \mid n$, let X and Y denote the left and right halves of the state respectively, $X = a_0a_1\cdots a_{n-1}$ and $Y = b_0b_1\cdots b_{n-1}$ are given as in Subsection 3.1. We cut the state as follows:

$$\begin{aligned} X \overset{cut}{\Longrightarrow} & (a_0 a_{\frac{n}{t}} a_{\frac{2n}{t}} \cdots a_{n-\frac{n}{t}}, \\ & a_1 a_{\frac{n}{t}+1} a_{\frac{2n}{t}+1} \cdots a_{n-\frac{n}{t}+1}, \\ & \cdots, \\ & a_{\frac{n}{t}-1} a_{\frac{2n}{t}-1} a_{\frac{3n}{t}-1} \cdots a_{n-1}) \\ \overset{def}{=} & (x_0, x_1, \cdots, x_{\frac{n}{t}-1}) \end{aligned} \tag{3}$$

$$\begin{aligned} Y \overset{cut}{\Longrightarrow} & (b_0 b_{\frac{n}{t}} b_{\frac{2n}{t}} \cdots b_{n-\frac{n}{t}}, \\ & b_1 b_{\frac{n}{t}+1} b_{\frac{2n}{t}+1} \cdots b_{n-\frac{n}{t}+1}, \\ & \cdots, \\ & b_{\frac{n}{t}-1} b_{\frac{2n}{t}-1} b_{\frac{3n}{t}-1} \cdots b_{n-1}) \\ \overset{def}{=} & (y_0, y_1, \cdots, y_{\frac{n}{t}-1}) \end{aligned} \tag{4}$$

Proposition 6 (Revised Circular Shift by One Bit). *Let $\mathbb{X}$ be an input multiset, $\boldsymbol{x}$ is an element of $\mathbb{X}$ and $\boldsymbol{x} \in \mathbb{F}_2^n$. Cut $\boldsymbol{x}$ as in Equation (3) and denoted by $\boldsymbol{x} \overset{def}{=} (x_0, x_1, \cdots, x_{\frac{n}{t}-1})$. Left circular shift by one bit creates $\boldsymbol{x^*}$ with $\boldsymbol{x^*} = \boldsymbol{x} \lll 1$. Let $\mathbb{X}^*$ denote the output multiset. Cut $\boldsymbol{x^*}$ as in (3) and denoted by $\boldsymbol{x^*} = (x_0^*, x_1^*, \cdots, x_{\frac{n}{t}-1}^*)$. Then we have $x_0^* = x_1, x_1^* = x_2, \cdots, x_{\frac{n}{t}-2}^* = x_{\frac{n}{t}-1}, x_{\frac{n}{t}-1}^* = (x_0 \lll 1)$. Moreover, if the input multiset has division property $\mathcal{D}_{\boldsymbol{k}}^{t,\frac{n}{t}}$ with $\boldsymbol{k} = (k_0, k_1, \cdots, k_{\frac{n}{t}-1})$, then the output multiset has division property $\mathcal{D}_{\boldsymbol{k^*}}^{t,\frac{n}{t}}$ with $\boldsymbol{k^*} = (k_0^*, k_1^*, \cdots, k_{\frac{n}{t}-1}^*)$, where $k_0^* = k_1, k_1^* = k_2, \cdots, k_{\frac{n}{t}-2}^* = k_{\frac{n}{t}-1}, k_{\frac{n}{t}-1}^* = k_0$.*

We give a schematic diagram of this improved state partition, and illustrate the circular shift operation based on the state partition in Fig.3. For the sake of simplicity, in Fig.3 we take Simon32 as an example and the piece size $t = 2$. Similarly, this partition is always feasible for Simon48 and Simon64.

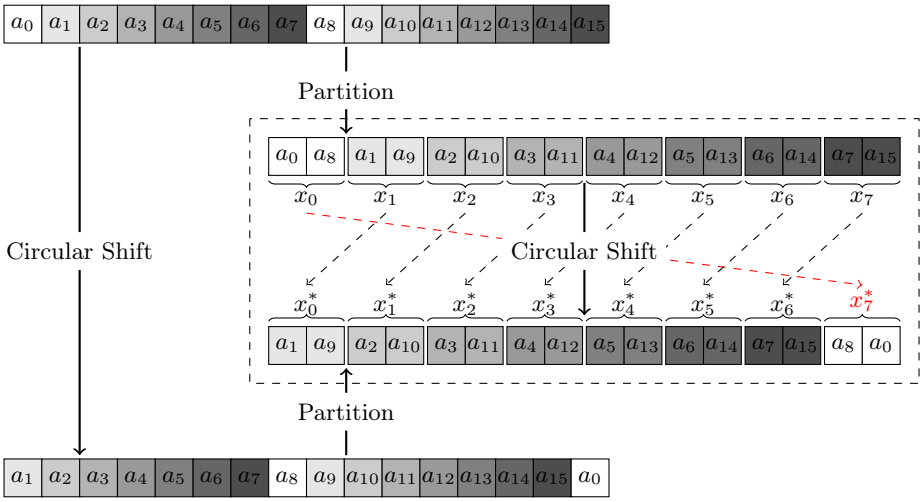

Fig. 3. Left Circular Shift by One Bit Regarding Improved Partition

Under this new state partition, circular shift by i bits just apply Proposition 6 for i times. Based on this partition, division property propagation of circular shift is just a circular shift of coordinates of the corresponding vectors, thus the number of vectors before and after circular shift will retain the same as we have expected. Note that this new state partition does not influence the division property propagations of bitwise AND and bitwise XOR operations, since unlike circular shift, bitwise AND and bitwise XOR operate on corresponding bits among pieces of *two* words. Thus together with Proposition 1, 2 and 3, division property propagation can be studied under this new state partition.

Comparison between two partitions. Thereafter, we denote the first state partition presented in Subsection 3.1 as **PartitionI**, denote the second state partition presented in Subsection 3.2 as **PartitionII**. (1). As described in Subsection 3.1, if the piece size t satisfies $0 < t < n$ in PartitionI, the use of Proposition 4 is unavoidable and this will create lots of extra vectors, however in PartitionII the number of vectors keep unchanged. Thus, the search algorithm under PartitionI is much more computationally expensive than PartitionII. (2). Since each vector adds a constraint on $\boldsymbol{u}$, PartitionI will create more vectors than PartitionII and this makes the set Γ_0 smaller under PartitionI. Thus, we expect that Γ_0 will shrink fast under PartitionI which will result in a shorter integral distinguisher. (3). In the choice of piece size t, PartitionII only needs $t \mid n$, however PartitionI has to consider the shift offsets and makes the partition use fewer times of Proposition 4. From this point of view PartitionII is much better.

3.3 Search Algorithm and Results

So far we have discussed division property propagations of all operations used in SIMON round function. Given a partition and a multiset of chosen inputs values, assume that the division property of this multiset is $\mathcal{D}_{\boldsymbol{k}}^{t,\frac{2n}{t}}$ under the given partition, the division property propagation of one round SIMON can be evaluated with the propagation characteristics introduced in the previous two subsections.

Search Algorithm Now we are ready to give our search Algorithm 1. The algorithm calls four functions:

1. ***RoundEval*** returns the division property of outputs of one round SIMON given the input division property and partition.
2. ***MinSum*** returns the minimal sum of coordinates of the given set of vectors. Since the division property can always propagate if the minimal sum of coordinates of all vectors is greater than one which means all the state bits are balanced. Note that if there exist $\boldsymbol{k}^{(i)}$ and $\boldsymbol{k}^{(j)}$ such that $\boldsymbol{k}^{(i)} \succeq \boldsymbol{k}^{(j)}$, then the sum of coordinates of $\boldsymbol{k}^{(i)}$ is greater than the sum of coordinates of $\boldsymbol{k}^{(j)}$. When the minimal sum of coordinates of all vectors in a given set $\boldsymbol{S}$ is greater than one, it is clear that any vector $\boldsymbol{u}$ with $W(\boldsymbol{u})$ a unit vector will satisfy $W(\boldsymbol{u}) \not\succeq \boldsymbol{k}^{(i)}$, for all $\boldsymbol{k}^{(i)} \in \boldsymbol{S}$. Thus, all state bits are balanced.
3. ***SizeReduce*** eliminates redundant vectors since redundant vectors provide no useful information. A vector $\boldsymbol{k}^{(i)}$ in $\{\boldsymbol{k}^{(0)}, \boldsymbol{k}^{(1)}, \cdots, \boldsymbol{k}^{(q)}\}$ is called redundant if there exists $\boldsymbol{k}^{(i)} \succeq \boldsymbol{k}^{(j)}$ with $j \neq i$.
4. ***MaxCoord*** returns the maximal coordinate of the given set S of vectors. If there exists a vector $\boldsymbol{k}^{(i)}$ with one of the coordinates greater than one after SizeReduce, we can choose a vector $\boldsymbol{u}$ whose weight $W(\boldsymbol{u})$ is a unit vector such that $\boldsymbol{k}^{(i)} \succeq W(\boldsymbol{u})$. In this case, any vector $\boldsymbol{k}^{(j)} \in \boldsymbol{S}$ will satisfy $W(\boldsymbol{u}) \not\succeq \boldsymbol{k}^{(j)}$, otherwise we have $\boldsymbol{k}^{(i)} \succeq \boldsymbol{k}^{(j)}$ which contradicts with the fact that $\boldsymbol{S}$ has been processed by SizeReduce function. Hence we get a nonzero vector $\boldsymbol{u}$ in Γ_0 which results to an even parity.

The most troublesome problems in the search algorithm are the rapid expansion of vectors created by copy function and circular shift (if PartitionI is adopted). Another problem we have met is that the SizeReduce algorithm which deletes the redundant vectors is very costly as mentioned in [16][22]. To solve the rapid expansion of vectors, we carefully choose the piece size t to make sure that the number of all possible vectors is of reasonable size and we store all possible vectors in a table. Each time when calculating one step of division property propagation we first mark all the vectors in such a table by 0, then calculate the division property propagation and mark the resulting vectors by 1, thus we can eliminate duplicated vectors. To solve the second problem we do not perform SizeReduce procedure through the search algorithm until there exists a vector whose coordinates add up to one. We found that when such a vector appears SizeReduce can be very efficient.

Algorithm 1 Integral Distinguisher search of SIMON2n

Require: $n, t, PartitionType$, Initial division property $\mathcal{D}_{\boldsymbol{k}}^{t,\frac{2n}{t}}$.
Ensure: Round r.
  $r \leftarrow 0$
  $\boldsymbol{k}_r^{(\mathbf{0})}, \boldsymbol{k}_r^{(\mathbf{1})}, \cdots, \boldsymbol{k}_r^{(\boldsymbol{q_r})} \leftarrow \boldsymbol{RoundEval}(n, t, ParititonType, \mathcal{D}_{\boldsymbol{k}}^{t,\frac{2n}{t}})$
  while $\boldsymbol{MinSum}(\boldsymbol{k}_r^{(\mathbf{0})}, \boldsymbol{k}_r^{(\mathbf{1})}, \cdots, \boldsymbol{k}_r^{(\boldsymbol{q_r})}) > 1$ **do**
    $r \leftarrow r + 1$
    $\boldsymbol{k}_r^{(\mathbf{0})}, \boldsymbol{k}_r^{(\mathbf{1})}, \cdots, \boldsymbol{k}_r^{(\boldsymbol{q_r})} \leftarrow \boldsymbol{RoundEval}(n, t, ParititonType, \mathcal{D}^{t,\frac{2n}{t}}_{\boldsymbol{k}_{r-1}^{(\mathbf{0})}, \boldsymbol{k}_{r-1}^{(\mathbf{1})}, \cdots, \boldsymbol{k}_{r-1}^{(\boldsymbol{q_{r-1}})}})$
  end while
  $\hat{\boldsymbol{k}}_r^{(\mathbf{0})}, \hat{\boldsymbol{k}}_r^{(\mathbf{1})}, \cdots, \hat{\boldsymbol{k}}_r^{(\boldsymbol{p_r})} \leftarrow \boldsymbol{SizeReduce}(\boldsymbol{k}_r^{(\mathbf{0})}, \boldsymbol{k}_r^{(\mathbf{1})}, \cdots, \boldsymbol{k}_r^{(\boldsymbol{q_r})})$
  while $\boldsymbol{MaxCoord}(\hat{\boldsymbol{k}}_r^{(\mathbf{0})}, \hat{\boldsymbol{k}}_r^{(\mathbf{1})}, \cdots, \hat{\boldsymbol{k}}_r^{(\boldsymbol{p_r})}) > 1$ **do**
    $r \leftarrow r + 1$
    $\boldsymbol{k}_r^{(\mathbf{0})}, \boldsymbol{k}_r^{(\mathbf{1})}, \cdots, \boldsymbol{k}_r^{(\boldsymbol{q_r})} \leftarrow \boldsymbol{RoundEval}(n, t, ParititonType, \mathcal{D}^{t,\frac{2n}{t}}_{\hat{\boldsymbol{k}}_{r-1}^{(\mathbf{0})}, \hat{\boldsymbol{k}}_{r-1}^{(\mathbf{1})}, \cdots, \hat{\boldsymbol{k}}_{r-1}^{(\boldsymbol{p_{r-1}})}})$
    $\hat{\boldsymbol{k}}_r^{(\mathbf{0})}, \hat{\boldsymbol{k}}_r^{(\mathbf{1})}, \cdots, \hat{\boldsymbol{k}}_r^{(\boldsymbol{p_r})} \leftarrow \boldsymbol{SizeReduce}(\boldsymbol{k}_r^{(\mathbf{0})}, \boldsymbol{k}_r^{(\mathbf{1})}, \cdots, \boldsymbol{k}_r^{(\boldsymbol{q_r})})$
  end while
  return r.

Table 1. Results on SIMON32 with $t = 2$

Target	$\log_2(\sharp\text{texts})$									
	r=3	r=4	r=5	r=6	r=7	r=8	r=9	r=10	r=11	r=12
PartitionI	2	3	5	17	25	28	30	-	-	-
PartitionII	-	-	-	2	5	18	22	25	28	30

Results To compare two state partitions proposed above, we run experiments on Simon32, and the results are listed in Table 1.

Table 1 shows that PartitionII gives better results with respect to data and the number of covered rounds. Note that in [18] Todo *et al.* applied bit-based division property on Simon32, thus, we only use Simon32 here as an example to illustrate the influences of the two state partitions. Since PartitionII gives better results, in the following we only consider PartitionII. Table 2 lists the longest integral distinguishers we found by PartitionII compared with previous results.

Table 2. Results on Simon48 and Simon64(PartitionII)

Cipher	[17]		This paper		
	$\log_2(\sharp\text{texts})$	Round	$\log_2(\sharp\text{texts})$	Round	Piece Size
Simon48	47	11	47	12	4
Simon64	63	11	63	12	8

Integral distinguishers found in Table 2 are similar for both Simon48 and Simon64. The input structure of Simon48 (resp. Simon64) keeps the leftmost bit of the state constant and let the remaining 47 (resp. 63) bits take on all possible values. Then the 24 (resp. 32)-bit word of right half of the state are balanced after 12 (resp. 12) rounds. Based on these distinguishers, integral attack can be launched. For Simon48, 18- and 19-round Simon can be attacked with key size 72 bits and 96 bits respectively. For Simon64, 19- and 20-round Simon can be attacked with key size 96 bits and 128 bits respectively. Due to the limit of space, we omit the details here.

Complexity of the Algorithm. Given a state partition with piece size t, division property is indicated by vectors in $\{0, 1, \cdots, t\}^{\frac{2n}{t}}$, and the number of such vectors is $(t+1)^{\frac{2n}{t}}$. Thus, the complexity of the algorithm is upper bounded by $(t+1)^{\frac{2n}{t}}$. For Simon48 we choose $t = 4$, thus the complexity of the algorithm is upper bounded by $5^{12} \approx 2^{27.9}$. Similarly, the complexity for Simon64 is upper bounded by $2^{25.4}$. We note that the methods used in [17] and [18] for searching integral distinguishers of Simon family, can be seen as two limit cases of our method by using a state partition. If we set the piece size $t = 1$, this case corresponds to the bit-based division property. In this case, the complexity of the algorithm is upper bounded by 2^{2n}, which is impractical for Simon with block size larger than 32 bits. On the other hand, if we set $t = n$, that is, the state of Simon is split into two parts: the left half and right half, and this corresponds to the approach used in [17]. In this case the complexity is upper bounded by $(n + 1)^2$, which is rather small even for Simon128.

4 Summary and Discussion

In this paper we showed improved division property of SIMON48 and SIMON64 by studying the division property propagations of SIMON round function operations. We first split the state into small pieces in order to get a trade-off between time-memory complexity and accuracy of the distinguishers, and then we studied the division property propagations of bitwise AND and circular shift which are basic operations used in SIMON. Moreover, we found that the state partition has a great influence on the propagation of division property. To be specific, the propagation of division property against circular shift operation. In this paper we presented two state partitions and experimented on SIMON32 to illustrate this influence, the experimental results showed that different state partitions resulted to different data and the number of covered rounds.

Based on the state partition and these propagation characteristics we presented a search algorithm and found 12-round distinguishers for SIMON48 and SIMON64 which improved Todo's results in [17] by one round for both ciphers.

Furthermore, we also applied the search algorithm to SIMON96 and SIMON128. We could only set $t = 16$ and $t = 32$ for SIMON96 and SIMON128 at the present, respectively. As a result, it seemed that the size of t was too large and we could not get better integral distinguishers for SIMON96 and SIMON128 than the results in [17]. However, we believe that the state partition and division property propagations studied in this can help study the division property of SIMON, and we leave it as a future work.

Acknowledgements. We are very grateful to the anonymous reviewers.

References

1. Abdelraheem, M.A., Alizadeh, J., Alkhzaimi, H.A., Aref, M.R., Bagheri, N., Gauravaram, P.: Improved linear cryptanalysis of reduced-round SIMON-32 and SIMON-48. Cryptology ePrint Archive, Report 2015/988 (2015), http://eprint.iacr.org/
2. Abdelraheem, M.A., Alizadeh, J., Alkhzaimi, H.A., Aref, M.R., Bagheri, N., Gauravaram, P., Lauridsen, M.M.: Improved linear cryptanalysis of reduced-round SIMON. Cryptology ePrint Archive, Report 2014/681 (2014), http://eprint.iacr.org/
3. Albrecht, M.R., Driessen, B., Kavun, E.B., Leander, G., Paar, C., Yalçın, T.: Block ciphers–focus on the linear layer (feat. PRIDE). In: Advances in Cryptology–CRYPTO 2014, pp. 57–76 (2014)
4. Ashur, T.: Improved linear trails for the block cipher SIMON. Cryptology ePrint Archive, Report 2015/285 (2015), http://eprint.iacr.org/
5. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK families of lightweight block ciphers. IACR Cryptology ePrint Archive 2013, 404 (2013)
6. Biryukov, A., Roy, A., Velichkov, V.: Differential analysis of block ciphers SIMON and SPECK. In: Fast Software Encryption. pp. 546–570. Springer (2014)

7. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J., Seurin, Y., Vikkelsoe, C.: PRESENT: An ultra-lightweight block cipher. Springer (2007)
8. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., et al.: PRINCE–A low-latency block cipher for pervasive computing applications. In: Advances in Cryptology–ASIACRYPT 2012, pp. 208–225. Springer (2012)
9. Boura, C., Naya-Plasencia, M., Suder, V.: Scrutinizing and improving impossible differential attacks: Applications to CLEFIA, Camellia, LBlock and SIMON. In: Advances in Cryptology–ASIACRYPT 2014, pp. 179–199. Springer (2014)
10. Chen, H., Wang, X.: Improved linear hull attack on round-reduced Simon with dynamic key-guessing techniques. Cryptology ePrint Archive, Report 2015/666 (2015), http://eprint.iacr.org/
11. Chen, Z., Wang, N., Wang, X.: Impossible differential cryptanalysis of reduced round SIMON. Cryptology ePrint Archive, Report 2015/286 (2015), http://eprint.iacr.org/
12. Daemen, J., Knudsen, L., Rijmen, V.: The block cipher Square. In: Fast Software Encryption. pp. 149–165. Springer (1997)
13. De Canniere, C., Dunkelman, O., Knežević, M.: KATAN and KTANTAN-a family of small and efficient hardware-oriented block ciphers. In: Cryptographic Hardware and Embedded Systems-CHES 2009, pp. 272–288. Springer (2009)
14. Knudsen, L., Wagner, D.: Integral cryptanalysis. In: Fast Software Encryption. pp. 112–127. Springer (2002)
15. Mourouzis, T., Song, G., Courtois, N., Christofii, M.: Advanced differential cryptanalysis of reduced-round SIMON64/128 using large-round statistical distinguishers. Cryptology ePrint Archive, Report 2015/481 (2015), http://eprint.iacr.org/
16. Todo, Y.: Integral cryptanalysis on full MISTY1. In: Advances in Cryptology–CRYPTO 2015, pp. 413–432. Springer (2015)
17. Todo, Y.: Structural evaluation by generalized integral property. In: Advances in Cryptology–EUROCRYPT 2015, pp. 287–314. Springer (2015)
18. Todo, Y., Morii, M.: Bit-based division property and application to SIMON family. Cryptology ePrint Archive, Report 2016/285 (2016), http://eprint.iacr.org/
19. Wang, N., Wang, X., Jia, K., Zhao, J.: Differential attacks on reduced SIMON versions with dynamic key-guessing techniques. Cryptology ePrint Archive, Report 2014/448 (2014), http://eprint.iacr.org/
20. 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 (2014)
21. Wu, W., Zhang, L.: LBlock: a lightweight block cipher. In: Applied Cryptography and Network Security. pp. 327–344. Springer (2011)
22. 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 (2015)
23. 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)

A Proofs of Propositions

A.1 Proof of Proposition 3

Proof. The aim is to prove under what conditions the output parity is always even given the input multiset division property. Let $\boldsymbol{u} = (u_0, u_1, \cdots, u_{\frac{n}{t}-1}) \in (\mathbb{F}_2^t)^{\frac{n}{t}}$ and

$$\begin{aligned}\bigoplus_{\boldsymbol{z}\in\mathbb{Z}} \boldsymbol{\pi_u}(\boldsymbol{z}) &= \bigoplus_{\boldsymbol{z}\in\mathbb{Z}} \prod_{i=0}^{\frac{n}{t}-1} \pi_{u_i}(z_i) \\ &= \bigoplus_{\boldsymbol{x}\in\mathbb{X}} \prod_{i=0}^{\frac{n}{t}-1} \pi_{u_i}(x_i \& x_i^*) \\ &= \bigoplus_{\boldsymbol{x}\in\mathbb{X}} \prod_{i=0}^{\frac{n}{t}-1} \pi_{u_i}(x_i) \prod_{i=0}^{\frac{n}{t}-1} \pi_{u_i}(x_i^*) \\ &= \bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{\pi_{(u,u)}}(\boldsymbol{x}) \ .\end{aligned} \tag{5}$$

In order to get an even parity of $\bigoplus_{\boldsymbol{z}\in\mathbb{Z}} \boldsymbol{\pi_u}(\boldsymbol{z})$, the parity of $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{\pi_{(u,u)}}(\boldsymbol{x})$ must be even. According to the input division property $\mathcal{D}_{\boldsymbol{k}}^{t,\frac{2n}{t}}$, it follows that there must exist $i \in \{0, 1, \cdots, \frac{n}{t}-1\}$ such that $w(u_i) < k_i$ or $w(u_i) < k_i^*$, thus $w(u_i) < \max\{k_i, k_i^*\}$. It's evident to see we need $W(\boldsymbol{u}) \not\succeq \hat{\boldsymbol{k}}$ which completes the proof.

A.2 Proof of Proposition 4

Proof. Let $\boldsymbol{u} = (u_0, u_1, \cdots, u_{\frac{n}{t}-1}) \in (\mathbb{F}_2^t)^{\frac{n}{t}}$, since $\boldsymbol{x^*} = \boldsymbol{x} \lll 1$ we have $\bigoplus_{\boldsymbol{x^*}\in\mathbb{X}^*} \boldsymbol{\pi_u}(\boldsymbol{x^*}) = \bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{\pi_{u \ggg 1}}(\boldsymbol{x})$. Let $\boldsymbol{v} = (v_0, v_1, \cdots, v_{\frac{n}{t}-1}) \in (\mathbb{F}_2^t)^{\frac{n}{t}}$, and $\boldsymbol{v} = \boldsymbol{u} \ggg 1$. It follows that $\bigoplus_{\boldsymbol{x^*}\in\mathbb{X}^*} \boldsymbol{\pi_u}(\boldsymbol{x^*}) = \bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{\pi_v}(\boldsymbol{x})$.

In order to prove for any $W(\boldsymbol{u}) \not\succeq \boldsymbol{k^{(0)}}, \boldsymbol{k^{(1)}}, \cdots, \boldsymbol{k^{(q-1)}}$ the corresponding $\boldsymbol{v} = \boldsymbol{u} \ggg 1$ satisfies $W(\boldsymbol{v}) \not\succeq \boldsymbol{k}$, we can prove for any $W(\boldsymbol{v}) \succeq \boldsymbol{k}$ there exists i such that the corresponding $\boldsymbol{u} = \boldsymbol{v} \lll 1$ satisfies $W(\boldsymbol{u}) \succeq \boldsymbol{k^{(i)}}$.

Write the bit string expression of $\boldsymbol{v}$ as $(v[0]v[1]\cdots v[t-1], v[t]v[t+1]\cdots v[2t-1], \cdots, v[n-t]v[n-t+1]\cdots v[n-1])$ with $v[i]$ the i-th bit of $\boldsymbol{v}$. Since $\boldsymbol{u} = \boldsymbol{v} \lll 1$, it is easy to see that

$$\begin{cases} w(u_0) & = w(v_0) - v[0] + v[t], \\ \vdots & \\ w(u_{\frac{n}{t}-2}) & = w(v_{\frac{n}{t}-2}) - v[n-2t] + v[n-t], \\ w(u_{\frac{n}{t}-1}) & = w(v_{\frac{n}{t}-1}) - v[n-t] + v[0] \ . \end{cases} \tag{6}$$

Since $W(\boldsymbol{v}) \succeq \boldsymbol{k}$, it follows that $w(v_i) \geq k_i$ for any $i \in \{0, 1, \cdots, \frac{n}{t} - 1\}$. Thus we have

$$\begin{cases} w(u_0) & = w(v_0) - v[0] + v[t] \geq k_0 - v[0] + v[t], \\ \vdots & \\ w(u_{\frac{n}{t}-2}) & = w(v_{\frac{n}{t}-2}) - v[n-2t] + v[n-t] \geq k_{\frac{n}{t}-1} - v[n-2t] + v[n-t], \\ w(u_{\frac{n}{t}-1}) & = w(v_{\frac{n}{t}-1}) - v[n-t] + v[0] \geq k_{\frac{n}{t}-1} - v[n-t] + v[0] \ . \end{cases} \tag{7}$$

If the coordinates of $(k_0 - v[0] + v[t], \cdots, k_{\frac{n}{t}-1} - v[n-t] + v[0])$ are between 0 and t, according to (2), there exists i such that $\boldsymbol{k^{(i)}} = (k_0 - v[0] + v[t], \cdots, k_{\frac{n}{t}-1} - v[n-t] + v[0])$. It follows that $W(\boldsymbol{u}) \succeq \boldsymbol{k^{(i)}}$ and we have thus proved the proposition.

However, if the coordinates of $(k_0 - v[0] + v[t], \cdots, k_{\frac{n}{t}-1} - v[n-t] + v[0])$ do not range from 0 to t, we can still find $\boldsymbol{k^{(i)}}$ such that $W(\boldsymbol{u}) \succeq \boldsymbol{k^{(i)}}$. Note that $k_i - v[i*t] + v[(i+1)*t] \leq w(u_i) \leq t$, thus $k_i - v[i*t] + v[(i+1)*t]$ will be invalid only if it happens that $k_i - v[i*t] + v[(i+1)*t] = -1$. If this happens we can deduce that $k_i = 0, v[i*t] = 1$ and $v[(i+1)*t] = 0$, thus bit string $v[0]v[t]\cdots v[\frac{n}{t} - 1]$ can not take on values of all one's or all zero's. We show next how to construct vector $\boldsymbol{k^{(i)}}$ such that $W(\boldsymbol{u}) \succeq \boldsymbol{k^{(i)}}$.

Without loss of generality, we assume that $k_i - v[i*t] + v[(i+1)*t] = -1$ and $v[s*t]v[(s+1)*t]\cdots v[i*t] = 01\cdots 1$ with $s < i$. Denote $(k_0 - v[0] + v[t], \cdots, k_{\frac{n}{t}-1} - v[n-t] + v[0]) = \boldsymbol{a}$, thus we have

$$\begin{cases} a_j & = k_j - v[j*t] + v[(j+1)*t] \quad \forall j \notin \{s, s+1, \cdots, i\}, \\ a_s & = k_s - 0 + 1 = k_s + 1, \\ a_{s+1} & = k_{s+1} - 1 + 1 = k_{s+1}, \\ \vdots & \\ a_i & = k_i - 1 + 0 = -1. \end{cases} \tag{8}$$

We construct $\boldsymbol{b}$ as follows:

$$\begin{cases} b_j & = k_j - v[j*t] + v[(j+1)*t] \quad \forall j \notin \{s, s+1, \cdots, i\}, \\ b_s & = k_s - 0 + 0 = k_s, \\ b_{s+1} & = k_{s+1} - 0 + 0 = k_{s+1}, \\ \vdots & \\ b_i & = k_i - 0 + 0 = k_i = 0. \end{cases} \tag{9}$$

Since $w(u_s) \geq k_s - v[s*t] + v[(s+1)*t] = k_s - 0 + 1 = a_s > b_s = k_s, w(u_{s+1}) \geq k_{s+1} - v[(s+1)*t] + v[(s+2)*t] = k_{s+1} = a_{s+1} = b_{s+1}, \cdots, w(u_i) \geq 0 = b_i$, thus we have constructed vector $\boldsymbol{b}$ such that $W(\boldsymbol{u}) \succeq \boldsymbol{b}$. If the coordinates of $\boldsymbol{b}$ range from 0 to t, $\boldsymbol{b}$ is a solution of (2) and there exists $\boldsymbol{k^{(t)}}$ such that $\boldsymbol{k^{(t)}} = \boldsymbol{b}$, thus we have proved the proposition. However, If these still exits coordinates of $\boldsymbol{b}$ equal to -1, we can repeat the above process to modify $\boldsymbol{b}$ until we get a valid solution.