Integral Attack Against Bit-Oriented Block Ciphers

Huiling Zhang[1,2,3(✉)], Wenling Wu[1,2,3], and Yanfeng Wang[1,2,3]

[1] TCA Laboratory, SKLCS, Institute of Software,
Chinese Academy of Sciences, Beijing, China
{zhanghuiling,wwl}@tca.iscas.ac.cn
[2] State Key Laboratory of Cryptology, P.O.Box 5159, Beijing 100878, China
[3] University of Chinese Academy of Sciences, Beijing, China

Abstract. Integral attack is an extremely important and extensively investigated cryptanalytic tool for symmetric-key primitives. In this paper, we improve the integral attack against bit-oriented ciphers. First, we propose the match-through-the-Sbox technique based on a specific property of the Sbox. Instead of computing the inverse of the Sbox in partial decryption, we independently calculate two Boolean functions which accept less input bits. The time complexity is thus reduced and the number of attacked rounds will be stretched. Second, we devise an easy-to-implement algorithm for construction of the integral distinguisher, which is then proved to be very effective for constructing lower order distinguishers. It shows SIMON 32, 48, 64, 96 and 128 has 13-, 14-, 17-, 21- and 25-round integral distinguisher, respectively, significantly improving the recent results from EUROCRYPT 2015. Finally, our techniques are applied to several ciphers. We attack one more round than the previous best integral attack for PRESENT and first evaluate the securities of SIMON family (except for SIMON 32) and RECTANGLE with integral attack.

Keywords: Bit-oriented block cipher · Integral attack · Meet-in-the-middle · Algebraic normal form · PRESENT · SIMON

1 Introduction

Integral attack was firstly proposed by Daemen et al. to evaluate the security of Square cipher [5] and then formalized by Knudsen and Wagner [7]. It consists of two phases, the integral distinguisher construction and the key recovery. An attacker starts with a set of 2^d plaintexts, which travel all values at d bit positions and take a constant value at others. If he proves that the state after r encryption rounds has a property with probability 1, e.g., the XOR of all values of the state equals to 0 at some bits which are known as *balanced* bits, a d-order integral distinguisher containing r rounds is thus achieved. Then for the second phase, the key space is reduced by checking balanced property. More specifically, the attacker guesses a part of subkeys and computes the balanced bits for every

S. Kwon and A. Yun (Eds.): ICISC 2015, LNCS 9558, pp. 102–118, 2016.
DOI: 10.1007/978-3-319-30840-1_7

Table 1. Summary of integral attack results

Target	#Rounds	Time	Data	Mem	Technique	Ref.
PRESENT-80	6	$2^{41.7}$	$2^{22.4}$	-	Bit-pattern	[14]
	9	2^{60}	$2^{20.3}$	2^{20}	7-round IND	[12]
	10	2^{35}	$2^{21.5}$	$2^{35.9}$	MTTS	Sect. 4.1
PRESENT-128	7	$2^{100.1}$	$2^{24.3}$	2^{80}	Bit-pattern	[14]
	10	$2^{99.3}$	$2^{22.4}$	2^{84}	7-round IND	[12]
	11	$2^{94.8}$	$2^{21.2}$	$2^{32.1}$	MTTS	Sect. 4.1
SIMON$(2n, m)$	19/20/21/	2^{m-1}	2^{2n-1}	$2^c c$	New IND	Sect. 4.2
	23/24/27/					
	28/32/33/34					
RECTANGLE-80	11	$2^{69.5}$	$2^{39.6}$	$2^{45.6}$	New IND and MTTS	Sect. 4.3
RECTANGLE-128	12	$2^{120.7}$	2^{45}	2^{39}	New IND and MTTS	Sect. 4.3

IND: integral distinguisher. MTTS: match-through-the-Sbox technique. $2n$: block size. m: key size. $c = 42/52/76/55/84/56/89/89/130/179$.

ciphertext by the partial decryption. If the XOR of the results is 0, the guessed value is a candidate for the right subkey, otherwise, it must be wrong.

Several techniques were proposed to optimize integral attack. In 2000, Ferguson et al. [6] introduced the *partial-sum technique*, which reduces the complexity of the partial decryption by guessing each subkey byte one after another and timely discarding the redundant data. The *meet-in-the-middle technique* for integral attack against Feistel ciphers was proposed by Sasaki et al. [8,9]. It employs the characteristic of Feistel structure and represents the balanced state by the XOR of two variables. Then, the partial decryption is separated into two independent parts, which greatly diminishes the time complexity. In 2014, Todo and Aoki applied the *FFT technique* to integral attack [11]. As the partial decryption is performed by Fast Walsh-Hadamard Transform, the time complexity does not depend on the number of chosen plaintexts, which is useful in an integral attack with enormous number of chosen plaintexts.

Recently, block ciphers which can be implemented in resource constraint environment, e.g., RFID Tags for a sensor network, have received much attention. This kind of block ciphers are called *lightweight block ciphers*. Lots of them are bit-oriented, such as PRESENT [3], PRINCE [4], PRIDE [1], RECTANGLE [15], as well as EPCBC [13] and SIMON family [2]. Traditional integral attack is less effective for these ciphers, which impels the cryptanalysts to develop new techniques. In FSE 2008, Z'aba et al. introduced a method of constructing integral distinguishers for bit-oriented ciphers [14]. Several bit-wise integral properties, denoted by *bit-patterns*, were defined and after that they showed the propagation of bit-patterns which will indicate the existence of integral distinguishers. This method requires the set of plaintexts to be ordered, and bit-patterns are easily destroyed after a few encryption rounds. Thus, the applications are limited. In EUROCRYPT 2015, Todo proposed a generalized integral property [10], named

division property, which evaluates the sum of the outputs of a parity function. A multi-set Λ has the division property D_k^n if and only if for all Boolean functions, $f: F_2^n \to F_2$, with algebraic degree $< k$, the sum of f on Λ is always 0. It has been pointed out that the propagation characteristic of division property for the nonlinear function totally depends on the algebraic degree. Hence, ciphers with low-degree functions are vulnerable to this analysis. Another generic method is by directly evaluating the algebraic degree, which is often called "higher-order differential attack". It utilizes the facts: (1) any state bit can be computed by a Boolean function taking plaintext and key bits as variables. (2) If the degree of the Boolean function is less than d for any value of the key, this state bit is balanced for 2^d chosen plaintexts. But, unfortunately, the existing approaches of degree evaluation are mostly rough, which will result in the misjudgment of balanced bits. In 2013, Wu et al. discovered that some properties of PRESENT's Sbox help to make a more accurate evaluation of the degree and they extended the integral distinguisher to 7 rounds [12]. However, their improvement is dedicated for PRESENT. The generic approach to improve the accuracy of degree evaluation is still unavailable.

Our Contributions. In this paper, we first attempt to optimize the key recovery phase by the property of Sboxes and propose *match-through-the-Sbox* technique for bit-oriented block ciphers. In previous key recovery, attackers compute the inverse of the Sbox to get the value of balanced bits. We discover that the computation can be divided into two independent parts when the Sbox has a specific property. This leads to a great decrease of the time complexity and furthermore leads to an extension of the number of attacked rounds. Then, we propose an algorithm of constructing integral distinguishers. It is inspired by [12], however our improvement is generic. The algorithm focuses on the terms occurring in the algebraic normal form of the Boolean function mapping plaintext bits to the state bit, which shows a tighter upper-bound of the degree. Therefore, integral distinguishers can be more effectively constructed. Moreover, it can be automatically implemented, that is to say, it does not require the complicated and tedious manual deductions, such as the proof of the 7-round distinguisher for PRESENT in [12]. As applications, we prove 13-, 14-, 17-, 21- and 25-round distinguisher for SIMON 32, 48, 64, 96 and 128, which contains 4, 3, 6, 8 and 12 more rounds than the previous best result, respectively. We also reduce the order of the integral distinguisher for RECTANGLE to 36 from 56, and thus the required number of chosen plaintexts for the integral attack can be decreased 2^{20} times. Finally, our techniques are applied to the integral attacks against PRESENT, SIMON family and RECTANGLE. The comparison of the results to previous integral attacks is summarized in Table 1.

Organization. Section 2 gives a brief review of Boolean function and integral attack. The techniques for improving integral attack against bit-oriented block ciphers are proposed in Sect. 3. In Sect. 4, we apply our techniques to PRESENT, SIMON family and RECTANGLE. Finally, Sect. 5 concludes this paper.

2 Preliminaries

2.1 Boolean Function

A *Boolean function* f on n variables is a mapping from F_2^n to F_2. It can be expressed with *algebraic normal form (ANF)*, that is

$$f(x) = \bigoplus_{\Gamma \in \mathcal{P}(\mathcal{N})} a_\Gamma \prod_{k \in \Gamma} x_k,$$

where $\mathcal{P}(\mathcal{N})$ is the power set of $\mathcal{N} = \{0, 1, \cdots, n-1\}$ and $x = (x_{n-1}, \cdots, x_0) \in F_2^n$. The *algebraic degree* of f, denoted by $deg(f)$, is the number of variables in the highest order term with the nonzero coefficient. It has following properties,

$$deg(fg) \leq deg(f) + deg(g), \tag{1}$$

$$deg(f \oplus g) \leq max\{deg(f), deg(g)\}. \tag{2}$$

A *vectorial Boolean function* F is a mapping from F_2^n into F_2^m. Such function being given, the Boolean functions $f_{m-1}, \cdots, f_0$ defined, at every $x \in F_2^n$, by $F(x) = (f_{m-1}(x), \cdots, f_0(x))$, are called the *coordinate Boolean functions* of F. The *algebraic degree* of F is defined as the highest degree of its coordinate Boolean functions. And the linear combinations, with non all-zero coefficients, of the coordinate functions are called the *component functions* of F.

2.2 Integral Attack

A well-known result from the theory of Boolean functions is that if the algebraic degree of a Boolean function is less than d, then the sum over the outputs of the function applied to all elements of an affine vector space of dimension $\geq d$ is zero. This property allows to exploit the algebraic degree to create integral distinguishers.

In a key-alternating block cipher, the intermediate state X^i is iteratively computed from the plaintext X^0 as:

$$X^i = F(K^{i-1} \oplus X^{i-1}),$$

where F is the round function. We denote the j-th bit of X^i by x_j^i. Assuming the block size is l, then, x_j^i can be expressed as a Boolean function on l plaintext bits, $x_0^0, x_1^0, \cdots, x_{l-1}^0$. To construct a d-order integral distinguisher, we first choose d bits in the plaintext as variables, supposing they are $x_0^0, x_1^0, \cdots, x_{d-1}^0$ for simplicity, and then evaluate the degree of the function (treating the rest of $l - d$ plaintext bits as constants). If the degree is less than d for any key, the sum of the values of x_j^i must be zero for a plaintext set whose elements travel all values of $x_0^0, x_1^0, \cdots, x_{d-1}^0$ and have a fixed value of remaining bits, i.e., x_j^i is a balanced bit.

A generic method of the degree evaluation is by recursion. Supposing the upper-bounds of the degrees for x_t^{i-1}, $0 \leq t < l$, are known, we can evaluate the degree for x_j^i according to property (1), (2).

3 Improvements of Integral Attack

In this section, we first propose the match-through-the-Sbox technique, which is very simple and effective for integral attacks against some bit-oriented ciphers. After that, we scrutinize and improve the integral distinguisher construction described in Sect. 2.2.

3.1 Match-Through-the-Sbox Technique

A property of the Boolean function is firstly defined. Then, the match-through-the-Sbox technique based on this property is developed, which is used in the key recovery phase to reduce the time and memory complexities and even extend the number of attacked rounds.

Definition 1. *Let f be a Boolean function on n variables. If there exist two Boolean functions on less than n variables, denoted by g_1 and g_2, satisfying*

$$f = g_1 \oplus g_2$$

f is said to be a separable Boolean function. In addition, if g_1 and g_2 do not share any variable, f is completely separable.

Example 1. Suppose f is a Boolean function on 4 variables.

- Let $f = x_3x_2x_1 \oplus x_2x_1x_0 \oplus x_3x_0 \oplus 1$. f is not a separable Boolean function.
- Let $f = x_3x_2x_1 \oplus x_2x_1x_0 \oplus 1$. f is separable, while it is not completely separable.
- Let $f = x_3x_2 \oplus x_1x_0 \oplus x_0$. f is completely separable.

Separable property commonly occurs for the component functions of 4-bit Sboxes (or their inverse mappings) in lightweight block ciphers, such as PRESENT, LBlock, PRINCE, etc., because lightweight block ciphers prefer the Sbox with compact algebraic expression for the sake of low cost. We will explain how to optimize the key recovery by using this property, which is called the match-through-the-Sbox technique.

Match-Through-the-Sbox Technique. Assume that y is a balanced bit. Let f be the coordinate Boolean function of S^{-1} such that $y = f(x_3, x_2, x_1, x_0)$, where x_i is the state bit outputted from the Sbox. In previous key recovery, attackers decrypt to the values of y and check whether the sum is zero for a plaintext set (denoted by Λ). However, if f is separable and $f = g_1(x_3, x_2, x_1) \oplus g_2(x_2, x_1, x_0)$ without loss of generality, we can write the checking equation $\bigoplus_{\Lambda} y = 0$ as the equivalent form:

$$\bigoplus_{\Lambda} g_1(x_3, x_2, x_1) = \bigoplus_{\Lambda} g_2(x_2, x_1, x_0). \tag{3}$$

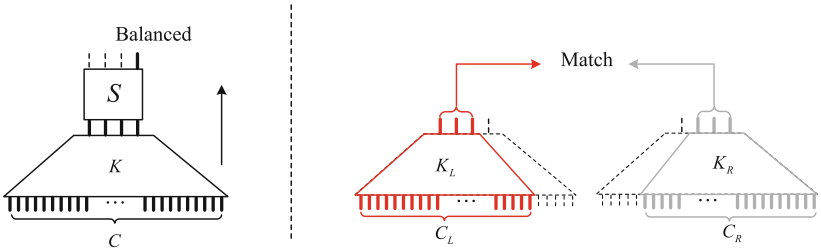

Fig. 1. Left: previous approach; right: match-through-the-Sbox technique

We then compute $x_3||x_2||x_1$ and $x_2||x_1||x_0$, independently, and finally check whether they match each other according to the Eq. (3).

The difference between the previous approach and our match-through-the-Sbox technique is depicted in Fig. 1. We now analyze their complexities without considering any optimal technique, for example, the partial-sum technique. Let K be the subkey information which needs to be guessed to obtain the value of y by partial decryption. Let C be the ciphertext bits involved in the computation of y. The previous attack costs time complexity $2^{|K|+|C|}$. Let K_L and K_R be the subkey that needs to be guessed to obtain the value of $x_3||x_2||x_1$ and $x_2||x_1||x_0$, respectively. And let C_L and C_R be the ciphertext bits involved in two partial decryption phases, respectively. Our method costs time complexity $2^{|K_L|+|C_L|} + 2^{|K_R|+|C_R|}$. Since K is the combination of K_L and K_R, we have $|K_L| < |K|$ and $|K_R| < |K|$. Similarly, $|C_L| < |C|$ and $|C_R| < |C|$ hold. Therefore, the time complexity of our method is much lower than the previous approach. Meanwhile, the cost of storing ciphertext bits will be reduced to $2^{|C_L|}|C_L| + 2^{|C_R|}|C_R|$ from $2^{|C|}|C|$. However, additional $\min\{2^{|K_L|}|K_L|, 2^{|K_R|}|K_R|\}$ bits of memory is required to find the matches.

3.2 Integral Distinguisher Construction

The traditional integral distinguisher construction, as shown in Sect. 2.2, only focuses on the upper-bound of the degree. Here we further pay attention to the terms occurring in the algebraic expression and then propose a searching algorithm.

We assume that $x_0^0, x_1^0, \cdots, x_{d-1}^0$ are chosen as d variables from l plaintext bits. The j-th bit of the state after i encryption rounds, x_j^i, is uniquely represented as a polynomial on these variables with coefficients in F_2, i.e.,

$$x_j^i = \bigoplus_{\Gamma \in \mathcal{S}(\mathcal{D})} \rho_\Gamma(k, c) \left(\prod_{t \in \Gamma} x_t^0 \right), \tag{4}$$

where $\mathcal{S}(\mathcal{D})$ denotes a subset of the power set of $\mathcal{D} = \{0, 1, \cdots, d-1\}$, k denotes the key, c is the constant in plaintext, and $\rho_\Gamma(k, c)$ either takes value 1 or else depends on k and c. The polynomial (hereinafter referred to as the "$Poly(x_j^i)$")

Algorithm 1. Construction of integral distinguishers

INDSearch$(d, l, \{t_0, t_1, \cdots, t_{d-1}\})$
$r = 0$
for $i = 0$ to $l - 1$ **do**
  $A^i \triangleq a^i_{2^d-1}||\cdots||a^i_0 = 0||\cdots||1$
end for
for $i = 0$ to $d - 1$ **do**
  $a^{t_i}_{2^i} = 1$
  $a^{t_i}_0 = 0$
end for
while $a^{l-1}_{2^d-1}\cdots a^1_{2^d-1}a^0_{2^d-1} = 0$ **do**
  for $i = 0$ to $l - 1$ **do**
    $T^i = A^i$
  end for
  $r = r + 1$
  for $i = 0$ to $l - 1$ **do**
    $A^i = EvalFunc(f_i, T^{l-1}, \cdots, T^0)$
  end for
end while
return $r - 1$

can be deduced from the coordinate functions of the round function by recursion, however, it is a tedious procedure since k is unknown. Therefore, we consider a collection of several monomials instead, denoted by $\Omega(x^i_j)$, which contains every term in $Poly(x^i_j)$ no matter which values k and c take. Thus if the highest order term $x^0_{d-1}\cdots x^0_1 x^0_0$ does not occur in $\Omega(x^i_j)$, it certainly has $\deg(x^i_j) < d$. The challenge is how to estimate the set $\Omega(x^i_j)$ as small as possible. We realize it by a straightforward method as follows.

Obviously, $\Omega(x^0_j)$ only contains term x^0_j if x^0_j is a variable, otherwise, it only contains constant term 1. $\Omega(x^1_t)$ can be evaluated from $\Omega(x^0_j)$, $0 \le j < l$, according to the t-th coordinate function of the round function. This process involves two basic operations, XOR and AND, which comply with the rules:

$$\Omega(x \oplus y) = \Omega(x) \cup \Omega(y) \quad \text{and} \quad \Omega(xy) = \{ab | a \in \Omega(x), b \in \Omega(y)\}. \tag{5}$$

where x and y are state or key bits. By noting that $\Omega(x) = \{1\}$ when x is a key bit, we can easily prove Eq. (5). In the recursive manner, $\Omega(x^i_j)$ is evaluated.

Search Algorithm. The basic idea has been explained above. We further describe each term by a d-bit string $a_{d-1}||\cdots||a_1||a_0$, where a_s $(0 \le s < d)$ takes 1 if the variable x^0_s occurs in the term, otherwise it takes 0. Then, $\Omega(x^i_j)$ corresponds to a 2^d-bit string $a_{2^d-1}||\cdots||a_1||a_0$, where s-th bit $(0 \le s < 2^d)$ takes 1 if the term $[s]_2$ (the binary representation of s) is in the set. Thereafter, the construction of the integral distinguisher can be performed by Algorithm 1.

In the algorithm, f_i is the i-th coordinate Boolean function of the round function. And $EvalFunc(f_i, T^{l-1}, \cdots, T^0)$ evaluates $\Omega(x^r_i)$ from $\Omega(x^{r-1}_j)$, $0 \le j < l$, by the rules:

(1) $\Omega(x \oplus y) = (a_{2^d-1}||\cdots||a_0) \vee (b_{2^d-1}||\cdots||b_0)$, where $\vee$ is bit-wise OR.
(2) $\Omega(xy) = a'_{2^d-1}||\cdots||a'_0$, where $a'_{[i]_2 \vee [j]_2} = 1$ if $a_i = 1$ and $b_j = 1$.

for $\Omega(x) = a_{2^d-1}||\cdots||a_0$ and $\Omega(y) = b_{2^d-1}||\cdots||b_0$. The time and memory complexity is 2^{2d} simple computations and $l2^d$ bits, respectively.

Table 2. Integral distinguishers for RECTANGLE and SIMON family.

r	Orders of r-round integral distinguishers											
	RECTANGLE		SIMON32		SIMON48		SIMON64		SIMON96		SIMON128	
	[15]	Ours	[10]	Ours	[10]	Ours	[10]	Ours	[10]	Ours	[10]	Ours
6	-	-	17	**1**	17	-	17	-	17	-	17	-
7	56	36	25	**2**	29	**1**	33	-	33	-	33	-
8	-	-	29	8	39	**3**	49	**1**	57	-	65	-
9	-	-	31	16	44	10	57	**2**	77	-	97	-
10	-	-	-	23	46	21	61	8	87	**1**	113	-
11	-	-	-	28	47	33	63	18	92	**2**	121	-
12	-	-	-	31	-	42	-	31	94	8	125	**1**
13	-	-	-	**31**	-	47	-	44	95	18	127	**2**
14	-	-	-	-	-	**47**	-	54	-	31	-	8
15	-	-	-	-	-	-	-	60	-	46	-	18
16	-	-	-	-	-	-	-	63	-	62	-	31
17	-	-	-	-	-	-	-	**63**	-	76	-	46
18	-	-	-	-	-	-	-	-	-	86	-	62
19	-	-	-	-	-	-	-	-	-	92	-	78
20	-	-	-	-	-	-	-	-	-	95	-	94
21	-	-	-	-	-	-	-	-	-	**95**	-	108
22	-	-	-	-	-	-	-	-	-	-	-	118
23	-	-	-	-	-	-	-	-	-	-	-	124
24	-	-	-	-	-	-	-	-	-	-	-	127
25	-	-	-	-	-	-	-	-	-	-	-	**127**

The generic method loses too much information and thus has a rough estimation, which is improved by our approach. Hence, our algorithm can more effectively construct the integral distinguishers, especially for ciphers with simple confusion components.

Results. We apply Algorithm 1 to construct integral distinguishers for RECTANGLE and SIMON family. Since the complexities grow exponentially with d, we choose small d and get a lower order integral distinguisher, and then extend it by applying the higher order integral method as shown in [16]. The results are displayed in Table 2, where the distinguishers colored red are directly achieved, and others are constructed based on them. Note that, we built the 7-round distinguisher for RECTANGLE by extending a 4-order distinguisher with 4 rounds constructed by Algorithm 1. Besides, the distinguishers marked in bold type are free extensions of previous ones. They choose the plaintext set as $\{(R, F(R) \oplus L) | (L, R) \in \Lambda\}$, where F is the round function and Λ is a plaintext set of the previous distinguisher. Compared with the previous best known results, our distinguishers have much lower order under the same number of rounds, furthermore, the longest distinguisher for each member of SIMON family contains 4/3/6/8/12 more rounds than the distinguisher in [10] which is constructed by the division property.

4 Applications

In this section, we demonstrate several applications of our techniques. Based on the match-through-the-Sbox technique we attack one more round than the

previous best integral attacks against PRESENT for both two versions. Besides, new integral distinguishers are used to launch the attacks against SIMON family and RECTANGLE.

4.1 Application to PRESENT

PRESENT is a 31-round SPN (Substitution Permutation Network) type block cipher with block size 64 bits. It supports 80- and 128-bit master key, which will be denoted by PRESENT-80 and PRESENT-128, respectively. The round function of PRESENT is the same for both versions and consists of standard operations such as subkey XOR, substitution and permutation. At the beginning of each round, 64-bit input is XORed with the subkey. Just after the subkey XOR, 16 identical 4×4 Sboxes are used in parallel as a non-linear substitution layer and finally a bit-wise permutation is performed so as to provide diffusion.

The subkeys K^i for $0 \leq i \leq 31$, where K^{31} is used for post-whitening, are derived from the master key by the key schedule. We provide the key schedule of PRESENT-80: 80-bit master key is stored in a key register and represented as $k_{79}||k_{78}||\cdots||k_0$. At i-th round, the 64 leftmost bits of actual content of the key register are extracted as the subkey K^i, that is, $K^i = k_{79}||k_{78}||\cdots||k_{16}$. After that, the key register is rotated by 61 bit positions to the left, then the Sbox is applied to left-most four bits of the key register and finally the round counter value, which is a different constant for each round, is XORed with $k_{19}k_{18}k_{17}k_{16}k_{15}$. The key schedule of PRESENT-128 is similar with PRESENT-80 except two Sboxes are applied. For more details, please refer to [3].

We denote by X^i the internal state which is the input to the i-th round and denote by Y^i its output after subkey XOR, i.e., $Y^i = X^i \oplus K^i$. We further describe 64 bits inside of X^i as $X^i = X^i[63]||\cdots||X^i[1]||X^i[0]$. A plaintext is loaded into the state X^0 and Y^{31} is produced as the ciphertext.

In 2013, Wu et al. proposed a 7-round integral distinguisher of PRESENT [12], that is, for a set of 2^{16} plaintexts where $X^0[0,\cdots,15]$ are active bits, the rightmost bit of Y^7 is balanced. We adopt this distinguisher in following attacks.

Property of the Sbox. Let $x = (x_3, x_2, x_1, x_0)$ be the input of S^{-1}, and y_0 be the rightmost bit of the output. It has

$$y_0 = x_3x_1 \oplus x_2 \oplus x_0 \oplus 1. \tag{6}$$

Suppose that $g_1 = x_3x_1$ and $g_2 = x_2 \oplus x_0 \oplus 1$. We get $y_0 = g_1 \oplus g_2$, which means the coordinate Boolean function of S^{-1} is completely separable.

Key Recovery Against 10-Round PRESENT-80. Choose $X^0[0,\cdots,15]$ as active bits and then $Y^7[0]$ is balanced. From Eq. (6), it has $Y^7[0] = X^8[48]X^8[16] \oplus X^8[32] \oplus X^8[0] \oplus 1$. Applying the match-through-the-Sbox technique, we need to check the following equation in the key recovery:

$$\bigoplus_{\Lambda} X^8[48]X^8[16] = \bigoplus_{\Lambda}(X^8[32] \oplus X^8[0]). \tag{7}$$

As shown in Fig. 2, the computation of $\bigoplus_{\Lambda} X^8[48]X^8[16]$ and $\bigoplus_{\Lambda}(X^8[32] \oplus X^8[0])$ involves the bits marked with red lines and black lines, respectively. One observation is that we only need to get the rightmost bit of the output from each S^{-1}, which is computed by this form: $(x_3 \oplus k_3)(x_1 \oplus k_1) \oplus (x_2 \oplus k_2) \oplus (x_0 \oplus k_0) \oplus 1$, where $k_0, \cdots, k_3$ are subkey bits. Therefore, we actually require 3-bit subkey information, $k_0 \oplus k_2, k_1, k_3$, instead of 4-bit. The details of the attack are as follows:

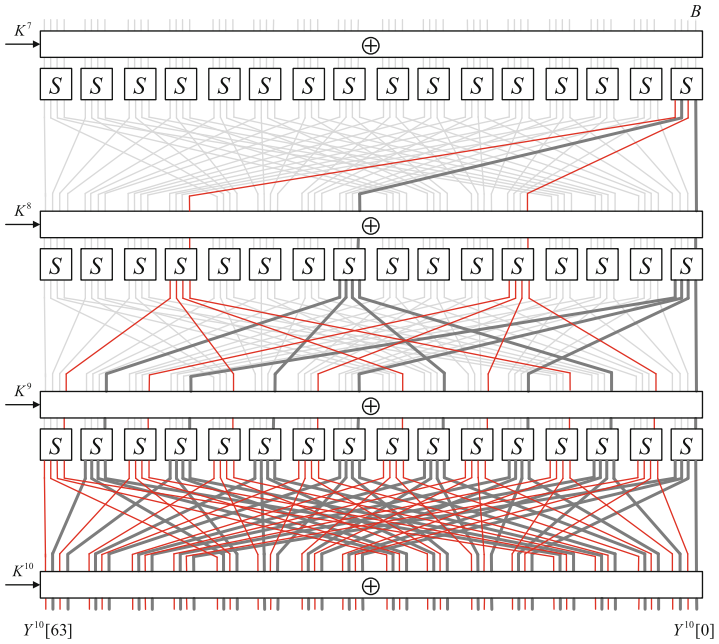

Fig. 2. 10-round key recovery (Color figure online)

1. Choose N plaintext sets, $\Lambda_0, \cdots, \Lambda_{N-1}$, satisfying the integral distinguisher, and get the ciphertexts after 10-round encryption.
2. Compute $\bigoplus_{\Lambda_s} X^8[48]X^8[16]$ for $0 \leq s < N$.
   - We guess 12-bit subkey: $K^{10}[i] \oplus K^{10}[i+32], K^{10}[i+16], K^{10}[i+48]$ for $i \in \{1, 3, 9, 11\}$, and compute the value of $Y^9[4] \oplus Y^9[36]$, $Y^9[12] \oplus Y^9[44]$ for each ciphertext. Count how many times each 14-bit value appears: $Y^9[4] \oplus Y^9[36]$, $Y^9[12] \oplus Y^9[44]$ and $Y^{10}[i] \oplus Y^{10}[i+32], Y^{10}[i+16], Y^{10}[i+48]$ for $i \in \{5, 7, 13, 15\}$, and then save the values which appear odd times in a table.
   - Guess 3 subkey bits, $K^{10}[5] \oplus K^{10}[37], K^{10}[21]$ and $K^{10}[53]$. Compress the data into 2^{12} texts of $Y^9[4] \oplus Y^9[36]$, $Y^9[20]$, $Y^9[12] \oplus Y^9[44]$ and $Y^{10}[i] \oplus Y^{10}[i+32], Y^{10}[i+16], Y^{10}[i+48]$ for $i \in \{7, 13, 15\}$.

- Guess 3 subkey bits, $K^{10}[13] \oplus K^{10}[45], K^{10}[29]$ and $K^{10}[61]$. Compress the data into 2^{10} texts of $Y^9[4] \oplus Y^9[36]$, $Y^9[20, 52]$, $Y^9[12] \oplus Y^9[44]$ and $Y^{10}[i] \oplus Y^{10}[i + 32], Y^{10}[i + 16], Y^{10}[i + 48]$ for $i \in \{7, 15\}$.
- Guess 3 subkey bits, $K^{10}[7] \oplus K^{10}[39], K^{10}[23]$ and $K^{10}[55]$. Compress the data into 2^8 texts of $Y^9[4] \oplus Y^9[36]$, $Y^9[20, 28, 52]$, $Y^9[12] \oplus Y^9[44]$ and $Y^{10}[15] \oplus Y^{10}[47], Y^{10}[31], Y^{10}[63]$.
- Guess 3 subkey bits, $K^{10}[15] \oplus K^{10}[47], K^{10}[31]$ and $K^{10}[63]$. Compress the data into 2^6 texts of $Y^9[4] \oplus Y^9[36]$, $Y^9[20, 28, 52, 60]$, $Y^9[12] \oplus Y^9[44]$.
- Thanks to the key schedule, $K^9[44]$ is obtained due to previous guessed subkey bit $K^{10}[25]$, and $K^9[28] \oplus K^9[60] = K^{10}[9] \oplus K^{10}[41]$. Therefore, we only need to guess 2-bit $K^9[12, 60]$, compute $Y^8[48]$ and compress the data into 2^4 texts of $Y^8[48]$, $Y^9[4] \oplus Y^9[36]$, $Y^9[20, 52]$.
- Similarly, we have $K^9[36] = K^{10}[15]$ and $K^9[20] \oplus K^9[52] = K^{10}[1] \oplus K^{10}[33]$. Hence, we guess 2-bit $K^9[4, 52]$, compute $Y^8[16]$ and compress the data into 2^2 texts of $Y^8[16, 48]$.
- Guess 2-bit value of $K^8[16, 48]$, compute $\bigoplus_{\Lambda_s} X^8[48]X^8[16]$ for $0 \leq s < N$ and save the values of 30-bit guessed subkey, K_1, in a hash table H indexed by the N-bit result.

3. Similar to Step 2, we compute $\bigoplus_{\Lambda_s}(Y^8[32] \oplus Y^8[0])$ (notice that it is $\bigoplus_{\Lambda_s}(X^8[32] \oplus X^8[0])$) for $0 \leq s < N$ by guessing 30-bit subkey K_2. The details are shown in Appendix A. Save K_2 in a hash table H' indexed by the N-bit sum.
4. Check the matches between H and H'. If the indexes match each other and the overlap information between K_1 and K_2 matches, $K_1||K_2$ is a key candidate.

Complexities. We have $|K_1| = 30$ and $|K_2| = 30$. Due to the key schedule, K_1 and K_2 overlap in 3 bits, $K^9[40, 48]$ and $K^9[24] \oplus K^9[56]$. Therefore, the total guessed subkey K contains $30 + 30 - 3 = 57$ bits. 2^{57-N} candidates for K are left after checking the matches. They are then exhaustively searched together with the remaining 23 subkey bits. The time complexity of Step 2 is evaluated as

$$\begin{aligned} &2^{12} \times 2^{16} \times 4 + 2^3 \times 2^{12} \times 2^{14} + \cdots + 2^2 \times 2^{28} \times 2^2 \\ &= 2^{30} + 2^{29} + 2^{30} + 2^{31} + 2^{32} + 2^{32} + 2^{32} + 2^{32} \\ &= 2^{34.3} \end{aligned}$$

computations of the Sbox. The time complexity of Step 3 is 2^{35} as explained in Appendix A. For the trade-off between time and data complexity, we choose $N = 46$. Hence, the attack totally costs $(2^{34.3} + 2^{35}) \times \frac{1}{16} \times \frac{1}{10} N + 2^{80-N} = 2^{35}$ 10-round encryptions. The data complexity is $2^{16}N = 2^{21.5}$ chosen plaintexts. The memory complexity depends on the storage of two hash tables, which is $2 \times (2^{30} \times 30) = 2^{35.9}$ bits.

For the integral attack against 11-round PRESENT-128, we first guess 64-bit K^{11} and decrypt the ciphertexts to Y^{10}. After that, procedures of the partial decryption are similar with the 10-round case. Specifically, we guess 27-bit and 25-bit subkeys to compute $\bigoplus_{\Lambda} X^8[48]X^8[16]$ and $\bigoplus_{\Lambda}(Y^8[32] \oplus Y^8[0])$, respectively,

as K^{11} has been guessed. They only overlap in one bit, $K^8[16] \oplus K^8[48]$, therefore, the total guessed subkey K contains $64 + 27 + 25 - 1 = 115$ bits. We analyze 38 plaintexts sets for the optimization of time complexity. The data complexity is hence $2^{21.2}$ chosen plaintexts. The memory complexity is evaluated by $2^{16} \times 64 \times 38 + 2^{27} \times 27 + 2^{25} \times 25 = 2^{32.1}$ bits, and time complexity is $2^{64} \times (2^{32} + 2^{32}) \times \frac{1}{16} \times \frac{1}{11} \times 38 + 2^{128-38} = 2^{94.8}$ 11-round encryptions.

4.2 Application to SIMON

SIMON is a family of lightweight block ciphers, optimized for performance on hardware devices, proposed by NSA. It is based on a classical Feistel construction operating on two n-bit branches. Denote the state entering i-th round by (L^i, R^i), which is further described as $(L^i[n-1]||\cdots||L^i[0],\ \ R^i[n-1]||\cdots||R^i[0])$. At each round, the round function F transforms the left branch in the following way,

$$F(L^i) = ((L^i <<< 8) \& (L^i <<< 1)) \oplus (L^i <<< 2).$$

The output of F is then XORed with the subkey K^i and with the right branch to form the left input of the next round. There exist in total ten members of the SIMON family, each one characterized by different block and key size. We denote a member of the SIMON family by SIMON$(2n/m)$, where $2n$ is the block size and m is the key size. The key schedule processes different procedures depending on $\frac{m}{n}$. While, it is always linear, and the master key can be derived if any sequence of $\frac{m}{n}$ consecutive subkeys are known. For detailed description, please refer to [2].

Key Recovery Against SIMON Family. We use the longest integral distinguisher shown in Table 2 for each member. Assume that s is the number of rounds of the distinguisher. We append t rounds to the distinguisher and give the key recovery attack against $(s+t)$-round SIMON$(2n/m)$.

We only decrypt to one balanced bit for the sake of less subkey involved. Suppose it is $R^s[b]$. It has $R^s[b] = (R^{s+1}[b-1]R^{s+1}[b-8]) \oplus R^{s+1}[b-2] \oplus L^{s+1}[b] \oplus K^s[b]$. Therefore, we check the following equation in key recovery:

$$\bigoplus (R^{s+1}[b-1]R^{s+1}[b-8]) = \bigoplus (R^{s+1}[b-2] \oplus L^{s+1}[b]). \tag{8}$$

If Eq. (8) holds for a guessed value, it is regarded as a candidate for the right subkey, which will be exhaustively searched together with the remaining key information. To check Eq. (8), we first decrypt to $\bigoplus (R^{s+1}[b-2] \oplus L^{s+1}[b])$ by applying the partial-sum technique. After that, a hash table is achieved, which saves the values of guessed subkey indexed with the result. Then we compute $\bigoplus (R^{s+1}[b-1]R^{s+1}[b-8])$ similarly and finally we find matches in the hash table.

Now we analysis the complexities of our attack. Let K_1 and K_2 be the subkey involved in the computation of $\bigoplus (R^{s+1}[b-2] \oplus L^{s+1}[b])$ and $\bigoplus (R^{s+1}[b-1]R^{s+1}[b-8])$, respectively. Denote the total subkey guessed in key recovery phase by $K_1 \cup K_2$. We count the number of bits in K_1, K_2 and $K_1 \cup K_2$ for each cipher in SIMON family, without considering the key schedule.

Table 3. Size of guessed subkey for t-round key recovery against SIMON family

(2n/m)	s	t	Size of guessed subkey		
			$\|K_1\|$	$\|K_2\|$	$\|K_1 \cup K_2\|$
(32/64)	13	6	42	45	49
(48/72)	14	6	52	59	65
(48/96)	14	7	76	83	89
(64/96)	17	6	55	63	69
(64/128)	17	7	84	93	99
(96/96)	21	6	56	65	73
(96/144)	21	7	89	101	109
(128/128)	25	7	89	101	110
(128/192)	25	8	130	145	155
(128/256)	25	9	179	197	207

The results are summarized in Table 3. Then the memory complexity is calculated by $2^{|K_1|}|K_1|$. The subkey space can be reduced by 1 bit for a plaintext set, furthermore, the master key space is accordingly reduced by 1 bit since the subkeys are related with linear relations. Hence, the time complexity of exhaustive search is 2^{m-1} $(s+t)$-round encryptions, which dominates the time complexity of the entire attack. The data complexity is 2^{2n-1} chosen plaintexts.

4.3 Application to RECTANGLE

RECTANGLE is a 25-round SPN cipher with bit-slice design. The 64-bit state $X^i = (X^i[63], \cdots, X^i[1], X^i[0])$ has equivalent representation as Fig. 3. The round function consists of three steps: AddSubkey, SubColumn, ShiftRow. Denote the state after AddSubkey in i-th round by Y^i, i.e., $Y^i = X^i \oplus K^i$. SubColumn is parallel application of Sboxes to the 4 bits in the same column. ShiftRow is a left rotation by 0, 1, 12 and 13 offset for row 0, 1, 2, and 3, respectively. The key schedule is similar to the encryption with less Sboxes and different rotations. Limited by the space, please refer to [15] for the details.

$$\begin{bmatrix} X[15] & X[14] & \cdots & X[1] & X[0] \\ X[31] & X[30] & \cdots & X[17] & X[16] \\ X[47] & X[46] & \cdots & X[33] & X[32] \\ X[63] & X[62] & \cdots & X[49] & X[48] \end{bmatrix} \Leftrightarrow \begin{bmatrix} X(0f) & X(0e) & \cdots & X(01) & X(00) \\ X(1f) & X(1e) & \cdots & X(11) & X(10) \\ X(2f) & X(2e) & \cdots & X(21) & X(20) \\ X(3f) & X(3e) & \cdots & X(31) & X(30) \end{bmatrix}$$

Fig. 3. Two-dimensional representation of the state

Our search algorithm constructs a 36-order integral distinguisher containing 7 rounds, which improves the 56-order distinguisher proposed by the designers. More specifically, for a set of 2^{36} plaintexts with $X^0(i0, i2, i3, i4, i6, i7, i8, ie, if)$, $0 \leq i < 4$, being active, the state after 7 encryption rounds has 22 balanced bits: $X^7(00, 10, 20, 01, 11, 21, 31, 12, 32, 03, 04, 14, 05, 15, 16, 2b, 2c, 2d, 3d, 3e, 0f, 2f)$.

Key Recovery Against RECTANGLE. Since $Y^7(01)$ and $Y^7(11)$ are balanced bits, $\bigoplus_{\Lambda}(Y^7(01) \oplus Y^7(11)) = 0$ holds. Applying the match-through-the-Sbox technique, we have

$$\bigoplus_{\Lambda} X^8(12)X^8(01) = \bigoplus_{\Lambda} \{X^8(3e)X^8(12) \oplus X^8(3e) \oplus X^8(12)\}. \tag{9}$$

We first prepare N sets of 2^{36} plaintexts satisfying the integral distinguisher. 48 bits of the ciphertext are related to the partial decryption for $X^8(01, 12)$, and also 48 bits of the ciphertext are related to the partial decryption of $X^8(3e, 12)$. Let K_1 and K_2 be the subkey involved in the computation of $X^8(01, 12)$ and $X^8(3e, 12)$, respectively. From the key schedule, $|K_1| = 60$, $|K_2| = 61$ and they overlap in 45 bits. We first guess 20 subkey bits shared by them for saving memory. Then we independently guess the rest of bits in K_1 and K_2 and independently compute the left and right of Eq. (9). Finally, the matches between the results are checked in order to sieve the guessed keys. Due to the limitation of space, we only show the complexities. The total time complexity of the attack is evaluated by $(2^{56}+2^{72.3}+2^{71.7})N \times \frac{1}{16} \times \frac{1}{11} + 2^{80-N} = 2^{65.3}N + 2^{80-N}$ encryptions. Choose $N = 12$, and then the time complexity is optimized to $2^{69.5}$ encryptions. The memory complexity is $2^{36} \times 64N + 2^{60-20} \times 40 + 2^{61-20} \times 41 = 2^{45.6}$ bits. The data complexity is $2^{36}N = 2^{39.6}$. In a similar manner, we can attack 12-round RECTANGLE-128 with the time, memory and data complexity being $2^{120.7}$ 12-round encryptions, 2^{45} bits and 2^{39} chosen plaintexts, respectively.

5 Conclusion

In this paper, we raised the power of integral attack against bit-oriented ciphers in both aspects. We first proposed the match-through-the-Sbox technique, which reduces the time and memory for the key recovery phase by using the separable property of the Sbox. It works similarly to the meet-in-the-middle technique for integral attack against Feistel ciphers, however, its application is not restricted by the structure but by the property of the Sbox. Therefore, it can be applied to SPN ciphers, such as PRESENT and RECTANGLE. Then, we devised a generic algorithm for increasing the accuracy of the degree evaluation and thereby improving the integral distinguisher. Limited by the memory cost, it is suitable for constructing lower order distinguishers, which can be extended to more rounds by the higher order method. The effect of this algorithm was

demonstrated for several ciphers. For instance, it constructed 13-, 14-, 17-, 21- and 25-round integral distinguisher for SIMON 32, 48, 64, 96 and 128, respectively, which are all the best results as we known. Besides, it reduced the order of the distinguisher for RECTANGLE to 36 from 56.

As applications, we launched the integral attack on several ciphers. For PRESENT, we can attack one more round than the previous best integral attack for two versions. Moreover, for SIMON family and RECTANGLE, we first evaluated their securities against integral attack (except for SIMON 32). Although our attacks do not pose a threat to these ciphers, it shows that the integral attack against bit-oriented ciphers has more room to be enhanced.

As we shown, both of our techniques are relevant to the property of the confusion component, for example the Sbox, hence, we conclude that integral attack is also sensitive to the Sbox for bit-oriented ciphers, except for the linear layer. Since the lightweight block ciphers is an actively discussed topic, we hope that this paper returns some useful feedback to future design and analysis.

A Details of Step 3

Compute $\bigoplus_{\Lambda_s}(Y^8[32] \oplus Y^8[0])$ for $0 \leq s < N$.

- We guess 12-bit subkey: $K^{10}[i] \oplus K^{10}[i+32], K^{10}[i+16], K^{10}[i+48]$ for $i \in \{0, 2, 8, 10\}$, and compute the value of $Y^9[0] \oplus Y^9[32]$, $Y^9[8] \oplus Y^9[40]$ for each ciphertext. Count how many times each 14-bit value appears: $Y^9[0] \oplus Y^9[32]$, $Y^9[8] \oplus Y^9[40]$ and $Y^{10}[i] \oplus Y^{10}[i+32], Y^{10}[i+16], Y^{10}[i+48]$ for $i \in \{4, 6, 12, 14\}$. And then pick the values which appear odd times.
- Guess 3 subkey bits, $K^{10}[4] \oplus K^{10}[36], K^{10}[20]$ and $K^{10}[52]$. Compress the data into at most 2^{12} values of $Y^9[0] \oplus Y^9[32]$, $Y^9[16]$, $Y^9[8] \oplus Y^9[40]$ and $Y^{10}[i] \oplus Y^{10}[i+32], Y^{10}[i+16], Y^{10}[i+48]$ for $i \in \{6, 12, 14\}$, which appear odd times.
- Guess 3 subkey bits, $K^{10}[12] \oplus K^{10}[44], K^{10}[28]$ and $K^{10}[60]$. Compress the data into 2^{10} texts of $Y^9[0] \oplus Y^9[32]$, $Y^9[16, 48]$, $Y^9[8] \oplus Y^9[40]$ and $Y^{10}[i] \oplus Y^{10}[i+32], Y^{10}[i+16], Y^{10}[i+48]$ for $i \in \{6, 14\}$.
- Guess 3 subkey bits, $K^{10}[6] \oplus K^{10}[38], K^{10}[22]$ and $K^{10}[54]$. Compress the data into 2^8 texts of $Y^9[0] \oplus Y^9[32]$, $Y^9[16, 24, 48]$, $Y^9[8] \oplus Y^9[40]$ and $Y^{10}[14] \oplus Y^{10}[46], Y^{10}[30], Y^{10}[62]$.
- Guess 3 subkey bits, $K^{10}[14] \oplus K^{10}[46], K^{10}[30]$ and $K^{10}[62]$. Compress the data into 2^6 texts of $Y^9[0] \oplus Y^9[32]$, $Y^9[16, 24, 48, 56]$, $Y^9[8] \oplus Y^9[40]$.
- Guess 3 subkey bits, $K^9[0] \oplus K^{10}[32], K^{10}[16]$ and $K^{10}[48]$. Compress the data into 2^4 texts of $Y^8[0]$, $Y^9[24, 56]$, $Y^9[8] \oplus Y^9[40]$.
- Guess 3 subkey bits, $K^9[8] \oplus K^{10}[40], K^{10}[24]$ and $K^{10}[56]$. Compress the data into 2^2 texts of $Y^8[0, 32]$.
- Compute $\bigoplus_{\Lambda_s}(Y^8[32] \oplus Y^8[0])$ for $0 \leq s < N$ and save the 30-bit guessed subkey K_2 in a hash table H' indexed by the corresponding N-bit result.

The time complexity is $2^{30}+2^{29}+2^{30}+2^{31}+2^{32}+2^{33}+2^{34} = 2^{35}$ computations of the Sbox.

References

1. Albrecht, M.R., Driessen, B., Kavun, E.B., Leander, G., Paar, C., Yalçın, T.: Block ciphers – focus on the linear layer (feat. PRIDE). In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 57–76. Springer, Heidelberg (2014)
2. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK families of lightweight block ciphers. Cryptology ePrint Archive, Report 2013/404 (2013). http://eprint.iacr.org/
3. Bogdanov, A.A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M., Seurin, Y., Vikkelsoe, C.: PRESENT: an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
4. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., et al.: PRINCE – a low-latency block cipher for pervasive computing applications. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 208–225. Springer, Heidelberg (2012)
5. Daemen, J., Knudsen, L.R., Rijmen, V.: The block cipher SQUARE. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 149–165. Springer, Heidelberg (1997)
6. Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D., Whiting, D.L.: Improved cryptanalysis of Rijndael. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 213–230. Springer, Heidelberg (2001)
7. Knudsen, L.R., Wagner, D.: Integral cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 112–127. Springer, Heidelberg (2002)
8. Sasaki, Y., Wang, L.: Comprehensive study of integral analysis on 22-round LBlock. In: Kwon, T., Lee, M.-K., Kwon, D. (eds.) ICISC 2012. LNCS, vol. 7839, pp. 156–169. Springer, Heidelberg (2013)
9. Sasaki, Y., Wang, L.: Meet-in-the-middle technique for integral attacks against feistel ciphers. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 234–251. Springer, Heidelberg (2013)
10. Todo, Y.: Structural evaluation by generalized integral property. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 287–314. Springer, Heidelberg (2015)
11. Todo, Y., Aoki, K.: FFT key recovery for integral attack. In: Gritzalis, D., Kiayias, A., Askoxylakis, I. (eds.) CANS 2014. LNCS, vol. 8813, pp. 64–81. Springer, Heidelberg (2014)
12. Wu, S., Wang, M.: Integral attacks on reduced-round PRESENT. In: Qing, S., Zhou, J., Liu, D. (eds.) ICICS 2013. LNCS, vol. 8233, pp. 331–345. Springer, Heidelberg (2013)
13. Yap, H., Khoo, K., Poschmann, A., Henricksen, M.: EPCBC - a block cipher suitable for electronic product code encryption. In: Lin, D., Tsudik, G., Wang, X. (eds.) CANS 2011. LNCS, vol. 7092, pp. 76–97. Springer, Heidelberg (2011)
14. Z'aba, M.R., Raddum, H., Henricksen, M., Dawson, E.: Bit-pattern based integral attack. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 363–381. Springer, Heidelberg (2008)

15. Zhang, W., Bao, Z., Lin, D., Rijmen, V., Yang, B., Verbauwhede, I.: RECTANGLE: a bit-slice ultra-lightweight block cipher suitable for multiple platforms. Cryptology ePrint Archive, Report 2014/084 (2014). http://eprint.iacr.org/
16. Zhang, W., Su, B., Wu, W., Feng, D., Wu, C.: Extending higher-order integral: an efficient unified algorithm of constructing integral distinguishers for block ciphers. In: Bao, F., Samarati, P., Zhou, J. (eds.) ACNS 2012. LNCS, vol. 7341, pp. 117–134. Springer, Heidelberg (2012)