SECURITY EVALUATION OF LIGHT-WEIGHT BLOCK CIPHERS BY GPGPU

Haruhisa Kosuge, Hidema Tanaka

National Defense Academy of Japan, Yokosuka, Japan

ABSTRACT

Nowadays, general purpose graphical processing unit (GPGPU) has been used in many ares. We use it for security evaluation of light-weight block ciphers. Light-weight block cipher is one of key technologies for small communication devices such as sensor network. To design a light-weight block cipher whose fastness and security are balanced, so that, its security margin should be evaluated exactly. One of security evaluation method, we focus on integral attack which exploits integral distinguisher to recover some round keys. Integral distinguisher is the main factor of integral attack, and it can be obtained by computer experiment. We use GPGPU to accelerate computer experiment. We propose an algorithm to search for upper bound of integral distinguisher by GPGPU. There are theoretical and experimental steps. We specify lower order integral distinguisher from upper bound one in the theoretical step. Such integral distinguisher is tested by computer experiment in the experimental step. By applying the proposal algorithm to HIGHT, TWINE, LBlock, PRESENT and RECTANGLE, we obtain more advantageous results.

KEYWORDS

GPGPU,Chosen plaintext attack, Light-weight block cipher, Integral attack

1. INTRODUCTION

General-purpose graphical processing unit (GPGPU) is a technology to adapt GPU for general purpose computings. GPGPU is used in various fields and it enables us to solve problems which take long time in ordinary platforms. As one of the usage of GPGPU, we consider security evaluation of light-weight block ciphers.

There has been a growing interest in light-weight block cipher which is a key technology to ensure security of communications among small devices such as sensor network and RFID. In ISO/IEC 29192-2, it is being standardized and PRESENT [1] and CLEFIA [2] have already been adopted. In addition, some of light-weight block ciphers are proposed, for example, HIGHT [3], KLEIN [4], LBlock [5], Piccolo [6], LED [7], TWINE [8], SIMON/SPECK [9] RECTANGLE [10] and so on. On the design of light-weight block ciphers, designers must consider trade-off between fastness and security. In order to design fast and secure cipher, security margin of block ciphers must be exactly determined. Therefore, security evaluation methods should be established, and we focus on integral attack which is a necessary tool for evaluation of block ciphers. Note that 64-bit block ciphers are in our scope, since they are common among light-weight block ciphers.

Integral attack is one of the major chosen plaintext attacks against block ciphers. The attack was firstly proposed as SQUARE attack by Daemon et al. [11], and then it was formalized as integral attack by Knudsen et al [12]. Recently, it has been drawing intense research interest because of its effectiveness and broad utility. Especially, full rounds attack on MISTY1 [13] by Todo shows outstanding effectiveness of integral attack [14].

Integral distinguisher is a property obtained by a set of chosen plaintexts, and the attackers can recover some round keys by using it. It is obtained by 2^n chosen plaintexts, where *n* (1 _ *n* _ 63) is the order of integral distinguisher and we call such one as *n*-th order integral distinguisher. A set of 2^n chosen plaintexts is encrypted for multiple rounds to make a set of outputs. An integrated value of the set of outputs is calculated. If there exist bits which are always 0 in such integrated value, we can define integral distinguisher. We call such bits as *balanced bits*. The number of rounds which balanced bits exist and one of balanced bits are parameters to indicate advantage for the attackers.

We can obtain distinguisher which holds in additional rounds by increasing the order [12]. Therefore, upper bound of integral distinguisher is 63rd order. Upper bound of integral distinguisher is necessary for deciding security margin of ciphers. Hence, the overall goal of this paper is to search for 63rd order integral distinguisher.

There are two types of search methods for integral distinguisher, computer experiment and theoretical search. Although there is a restriction in computer resource, we can execute computer experiment in any block ciphers. On the other hand, we can obtain upper bound of integral distinguisher in theoretical search, since there is not such restriction. However, there is a restriction in applicable cipher function and there can be a mismatch between theoretical result and computational one. This paper focuses on search method using computer experiment and aims to improve it. Computer experiment is very effective and easy to apply when the block length is short such as SIMON32 [9][15]. We can obtain the upper bound of integral distinguisher by 31st order integral distinguisher, and their computer experiments can be executed in realistic times. However, 63rd order integral distinguisher itself can not be obtained by computer experiment. Therefore, we propose a new technique to achieve it. Also, we accelerate computer experiment by using GPGPU.

2. PRELIMINARY

2.1. NOTATIONS

We use the notations shown in Table 1.

2.2. INTEGRAL DISTINGUISHER

Integral distinguisher depends on *input condition* which is defined by variable bits. When the attackers chooses *n* bits as variable, he needs to prepare 2^n chosen plaintexts. We call *n* as *order* and such integral distinguisher as *n-th order integral distinguisher*. In 2^n plaintexts, a concatenation of variable bits takes every element of $F^n{}_2$ and one of constant bits takes a constant value. A set of chosen plaintexts satisfies

$$\begin{aligned} &v_0\|v_1\|\ldots\|v_{n-1} = \{0, 1, \ldots, 2^n - 1\}, \\ &c_0\|c_1\|\ldots\|c_{63-n} = const, \end{aligned} \qquad (1)$$

where v$_0$jjv$_1$jj:::jjv$_{n\ 1}$ denotes a concatenation of variable bits, and $c_0$jj$c_1$jj:::jj$c_{N\,n\ 1}$ a concatenation of constant bits. Let *A* be an index set of variable bits, and a set of chosen plaintexts defined by *A* is $X_A = \{X_A^i | X_A^i \in \mathbb{F}_2^{64}, 0 \le i \le 2^n - 1\}$. In the following, we denote input condition by X$_A$.

Let E^γ be γ rounds cipher function and K^γ be a set of round keys used from first to γ-th round encryption. Integration of input condition X_A is defined as follows.

$$\int \mathcal{X}_A = \bigoplus_{i=0}^{2^n-1} E^{\gamma}(X_A^i, K^{\gamma}), \tag{2}$$

Where $\oplus$ denotes **XOR** summation.If there exist at least one bit which is always 0 in $\int X_A$ for any values of constant subblocks and round keys, we can define integral distinguisher. We call such

Table 1: Notation of variables.

Notation	Description
m	bit length of a subblock ($2 \le m \le 32$).
n	order of integral distinguisher which is defined by the number of variable bits ($1 \le n \le 63$).
K^{γ}	set of round keys used for encryption from first to γ-th round ($\lvert K^{\gamma}\rvert = \kappa\gamma$).
E^{γ}	γ rounds cipher function which is defined by iteration of a round function for γ times ($\mathbb{F}_2^{64} \times \mathbb{F}_2^{\kappa\gamma} \to \mathbb{F}_2^{64}$).
$A = \{a_0, a_1, ..., a_{n-1}\}$	index set of variable bits.
$B = \{b_0, b_1, ..., b_{k-1}\}$	index set of balanced bits.
$\mathcal{X}_A$	input condition; set of chosen plaintexts whose variable bits are defined by A ($\mathcal{X}_A = \{X_A^i \mid X_A^i \in \mathbb{F}_2^{64}, 0 \le i \le 2^n - 1\}$).
Y_B	output property; integrated value of outputs of $E^{\gamma}(\mathcal{X}_A, K^{\gamma})$, and balanced bits are denoted by B ($Y_B = \bigoplus_{i=0}^{2^n-1} E^{\gamma}(X_A^i, K^{\gamma})$).
$\mathcal{X}_A \to^{\gamma} Y_B$	integral distinguisher; integrated value has output property Y_B when a set of chosen plaintexts defined by input condition $\mathcal{X}_A$ is encrypted for γ rounds ($Y_B = \bigoplus_{i=0}^{2^n-1} E^{\gamma}(X_A^i, K^{\gamma})$).
$\mathcal{X}_A \Supset \mathcal{X}_{A'}^{\gamma'}$	inclusive relation; $\mathcal{X}_A$ includes $\mathcal{X}_{A'}$. $\mathcal{X}_{A'}$ is constructed in a set of outputs $E^{\gamma'}(\mathcal{X}_A, K^{\gamma'})$.

bits as *balansed bits*, and we define *output property* using them. Let *B* be an index set of balanced bits and integrated value denoted by *B* is $Y_B \in F_2^{64}$.

We define integral distinguisher by $X_A \to^{\gamma} Y_B$. It denotes that an integrated value has output property Y_B when a set of chosen plaintexts defined by input condition X_A is encrypted for γ rounds. Due to the limited space, we omit the integral attack scenario using integral distinguisher, and typical case is shown in [16].

2.3. Property of Integral Distinguisher

Focusing on output property Y_B, it is obtained by multiple input conditions. We call such property as inclusive relation which holds among multiple input conditions, and we define it as follows.

Definition 1 *If two input conditions $\mathcal{X}_A$ and $\mathcal{X}_{A'}$ satisfies any of following conditions, $\mathcal{X}_A$ and $\mathcal{X}_{A'}$ are in inclusive relation and $\mathcal{X}_A$ includes $\mathcal{X}_{A'}$.*
(i) $\mathcal{X}_A \to^{\gamma} Y_B$ and $\mathcal{X}_{A'} \to^{\gamma} Y_{B'}$ hold, where $A \supset A'$ and $B \supset B'$.
(ii) New input condition $\mathcal{X}_{A'}$ is constructed in a set of outputs $E^1(\mathcal{X}_A, K^1)$.

If condition (i) holds, we write $\mathcal{X}_A \Supset \mathcal{X}_{A'}$. If condition ($ii$) holds, we write $\mathcal{X}_A \Supset \mathcal{X}^1_{A'}$. Two inclusive relations $\mathcal{X}_A \Supset \mathcal{X}_{A''}$ and $\mathcal{X}_{A''} \Supset \mathcal{X}_{A'}$ are concatenated to $\mathcal{X}_A \Supset \mathcal{X}_{A'}$. Also, $\mathcal{X}_A \Supset \mathcal{X}^1_{A''}$ and $\mathcal{X}_{A''} \Supset \mathcal{X}^1_{A'}$ are concatenated to $\mathcal{X}_A \Supset \mathcal{X}^2_{A'}$. From definition 1, following property is obtained.

Property 1 *If $A \supset A'$, then $\mathcal{X}_A \Supset \mathcal{X}_{A'}$.*

Proof *Suppose integral distinguisher $\mathcal{X}_A \to^{\gamma} Y_B$ and $\mathcal{X}_{A'} \to^{\gamma} Y_{B'}$ hold, where $|A| = n$ and $|A'| = n'$ ($n > n'$). From Eq.(1), a set of plaintexts $\mathcal{X}_A$ is regarded as a set whose elements are sets of plaintexts which are defined by $\mathcal{X}_{A'}$. Therefore, $\mathcal{X}_A$ is written as $\{X_A^{2^{n'}j+k} | 0 \leq j \leq 2^{n-n'} - 1, 0 \leq k \leq 2^{n'} - 1\}$, where $X_A^{2^{n'}j+k}$ takes every element of $\mathbb{F}_2^{n'}$ for any values of j. Since $\mathcal{X}_{A'} \to^{\gamma} Y_{B'}$ hold, $\sum_{k=0}^{2^{n'}-1} E^{\gamma}(X_A^{2^{n'}j+k}, K^{\gamma}) = Y^j_{B'}$ hold*

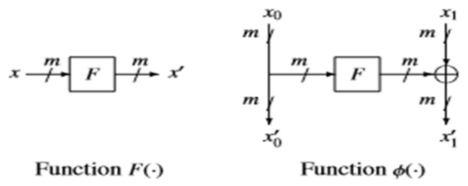

Figure 1: Two bijective functions $F(\cdot)$ and $\phi(\cdot)$.

for any values of j, where $Y^j_{B'}$ is an integrated value whose balanced bits are denoted by B'. An integrated value Y_B is calculated as

$$\begin{aligned} Y_B &= \sum_{i=0}^{2^n-1} E^{\gamma}(X^i_A, K^{\gamma}) \\ &= \sum_{j=0}^{2^{n-n'}-1} \sum_{k=0}^{2^{n'}-1} E^{\gamma}(X_A^{2^{n'}j+k}, K^{\gamma}) \\ &= \sum_{j=0}^{2^{n-n'}-1} Y^j_{B'}. \end{aligned} \tag{3}$$

Since the integrated value Y_B have the same balanced bits with $Y^j_{B'}$, $\mathcal{X}_A \to^{\gamma} Y_{B'}$ holds.

Property 2 *If $\mathcal{X}_{A'} \to^{\gamma} Y_{B'}$ and $\mathcal{X}_A \Supset \mathcal{X}^{\gamma'}_{A'}$ hold, then $\mathcal{X}_A \to^{\gamma+\gamma'} Y_{B'}$ holds.*

Proof *By considering the case when $\gamma' = 1$, we can prove for any values of γ' inductively. Let $\mathcal{X}_{A''}$ be an input condition which satisfies $\mathcal{X}_A \Supset \mathcal{X}_{A''} \Supset \mathcal{X}^1_{A'}$ and $|A''| = |A'|$. From definition of $\mathcal{X}_{A''} \Supset \mathcal{X}^1_{A'}$, we can calculate as $\mathcal{X}_{A'} = E^1(\mathcal{X}_{A''}, K^1)$, where K^1 is a round key of first round. An integrated value $Y_{B''}$ is calculated as*

$$\begin{aligned} Y_{B''} &= \sum_{i=0}^{2^{n''}-1} E^{\gamma+1}(X^i_{A''}, K^{\gamma}) \\ &= \sum_{i=0}^{2^{n''}-1} E^{\gamma}(E^1(X^i_{A''}, K^1), K') \\ &= \sum_{i=0}^{2^{n''}-1} E^{\gamma}(X^i_{A'}, K') \\ &= Y_{B'}, \end{aligned} \tag{4}$$

where $n'' = |A''|$ and $K^{\gamma} = K^1 \| K'$. Therefore, $\mathcal{X}_{A''} \to^{\gamma+1} Y_{B'}$ holds, and $\mathcal{X}_A \to^{\gamma+1} Y_{B'}$ holds from Property 1. The inclusive relation $\alpha_A \supset \alpha^{\gamma'}_{A'}$ is composed of successive relations such as $\alpha_A \supset \alpha^1_{A'}$. Therefore $\alpha_A \to^{\gamma+\gamma'} \beta_{B'}$ holds for any values of γ'.

The above properties are analysed by exploiting bijective characteristics in partial functions.

2.4. BIJECTIVE CHARACTERISTICS OF PARTIAL FUNCTIONS

In a round function, there may be bijective partial functions. Integral distinguisher is searched by exploiting such characteristics. In this paper, we define two bijective functions F(·) and ϕ(·), and show them in Fig.1.

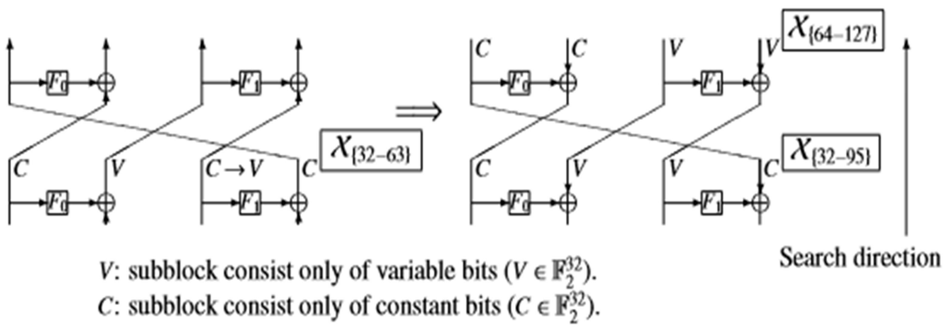
F_0
F_1
C
V
$C \to V$
$X_{\{32-63\}}$
$X_{\{64-127\}}$
$X_{\{32-95\}}$
Search direction
V: subblock consist only of variable bits ($V \in \mathbb{F}_2^{32}$).
C: subblock consist only of constant bits ($C \in \mathbb{F}_2^{32}$).

Figure 2: Outline of the extension of integral distinguisher in CLEFIA

A function $F(\cdot)$ is bijective, s.t., $F : \mathbb{F}_2^m \to \mathbb{F}_2^m$. SPN structures such as PRESENT [1] and RECTANGLE [10] are considered as functions consist of multiple functions $F(\cdot)$. In PRESENT and RECTANGLE, S-boxes are known, however, round keys XORed before S-box operations are unknown. Therefore, they are considered as random bijective functions.

The other bijective function $\phi(\cdot)$ is written as

$$x'_0 \| x'_1 = \phi(x_0 \| x_1),\ s.t.,\ x'_0 = x_0,\ x'_1 = F(x_0) \oplus x_1, \tag{5}$$

where $x_0, x_1, x'_0, x'_1 \in \mathbb{F}_2^m$. Feistel structures such as HIGHT [3], LBlock [5] and TWINE [8] are considered as functions consist of multiple functions $\phi(\cdot)$. Since $F(\cdot)$ is a random bijective function because of unknown round keys, $\phi(\cdot)$ is also a random bijective function.

If x_0 is a constant value in Eq.(5), we define another function ϕ' as

$$C \| x'_1 = \phi'(C \| x_1),\ s.t.,\ x'_1 = F(C) \oplus x_1, \tag{6}$$

where $C \in \mathbb{F}_2^m$ is a constant value. Since $F(C)$ is a random constant value, $\phi'(\cdot)$ is a random bijective function, s.t., $\phi' : \mathbb{F}_2^m \to \mathbb{F}_2^m$, when we foncus on x'_1.

Exploiting bijective characteristics of partial functions of a round function, inclusive relations (see Sec.2.3) which hold in a cipher function can be analysed.

3. CONVENTIONAL ALGORITHM

Knudsen et al. formalized integral attack for applying mainly to block ciphers in which all operations are executed in m-bit unit (subblock). We call the algorithm as conventional algorithm. It is divided into two steps. The first step is a search algorithm for m-th order integral distinguisher. The second step is an algorithm to extend m-th order integral distinguisher to higher order one which holds in additional rounds (extension algorithm). Due to the limited space, we omit the first step, and typical case is shown in[17]. Integral distinguisher can be extended by using inclusive relation (Sec.2.3) and bijective characteristics of partial functions (Sec.2.4). We demonstrate extension algorithm in Sec.3.5 and show a problem of it in Sec.3.6.

3.1. Extension Algorithm

We demonstrate the extension algorithm in 128-bit block cipher CLEFIA[2]. Suppose that following 32nd order integral distinguisher is known (first step).

$$\mathcal{X}_{\{32-63\}} \rightarrow^{6} Y_{\{96-127\}} \tag{7}$$

We extend Eq.(7) to 64th order Integral Distinguisher. Fig.2 shows outline of extension of integral distinguisher. An inverse function of the round function E^{-1} consists of two bijective functions

$\phi(\cdot)$ (see Eq.(5)). Input condition $\mathcal{X}_{\{32-63\}}$ is inputted to E^{-1}. In one of the functions, only constant bits are inputted and random constant bits are outputted. On the other hand, the other function has variable bits in input. To exploit a bijective characteristic of $\phi(\cdot)$, we consider an input condition $\mathcal{X}_{\{32-95\}}$, where $\mathcal{X}_{\{32-95\}} \Supset \mathcal{X}_{\{32-63\}}$ form Property 1. In the function, all elements of $\mathbb{F}_2^{64}$ are inputted without multiplicity. Since $\phi(\cdot)$ is bijective, all elements of $\mathbb{F}_2^{64}$ are outputted without multiplicity. Therefore, new input condition $\mathcal{X}_{\{64-127\}}$ is constructed in the output of E^{-1}. From Definition 1, $\mathcal{X}_{\{64-127\}} \Supset \mathcal{X}^{1}_{\{32-95\}}$ is obtained. Also, $\mathcal{X}_{\{64-127\}} \Supset \mathcal{X}^{1}_{\{32-63\}}$ holds from $\mathcal{X}_{\{32-95\}} \Supset \mathcal{X}_{\{32-63\}}$. From Eq.(7), we have

$$\begin{cases} & \mathcal{X}_{\{32-63\}} \rightarrow^{6} Y_{\{96-127\}} \\ \mathcal{X}_{\{64-127\}} \Supset & \mathcal{X}^{1}_{\{32-63\}} \end{cases}$$
$$\Rightarrow \mathcal{X}_{\{64-127\}} \rightarrow^{7} Y_{\{96-127\}}. \tag{8}$$

In the same way, following 96th order integral distinguisher is obtained by extending Eq.(8) for one more round.

$$\begin{cases} & \mathcal{X}_{\{32-63\}} \rightarrow^{6} Y_{\{96-127\}} \\ \mathcal{X}_{\{0-64,96-12\}} \Supset & \mathcal{X}^{2}_{\{32-63\}} \end{cases}$$
$$\Rightarrow \mathcal{X}_{\{0-64,96-127\}} \rightarrow^{8} Y_{\{96-127\}} \tag{9}$$

In this way, higher order integral distinguisher is obtained by lower order one in extension algorithm.

3.2. Problem of Conventional Algorithm

In the conventional algorithm, upper bound of integral distinguisher is obtained as an extension of m-th order one. However, upper bound of integral distinguisher includes many other integral distinguisher than m-th order one. For example, Shibayama et al. showed 12th order integral distinguisher found by computer experiment and its extension in TWINE [18].

$$\begin{cases} \mathcal{X}_{\{0-3,8-15\}} \rightarrow^{10} Y_{\{4-7,12-15,36-39,44-47,52-55,60-63\}} \\ \mathcal{X}_{\{0-55,60-63\}} \Supset \mathcal{X}^{5}_{\{0-3,8-15\}} \end{cases}$$
$$\Rightarrow \mathcal{X}_{\{0-55,60-63\}} \rightarrow^{15} Y_{\{4-7,12-15,36-39,44-47,52-55,60-63\}} \tag{10}$$

On the other hand, following 4th order integral distinguisher and its extension are obtained in the conventional algorithm.

$$\begin{cases} & \mathcal{X}_{\{4-7\}} \rightarrow^{9} Y_{\{4-7,12-15,52-55,60-63\}} \\ \mathcal{X}_{\{0-55,60-63\}} \Supset & \mathcal{X}^{6}_{\{4-7\}} \end{cases}$$
$$\Rightarrow \mathcal{X}_{\{0-55,60-63\}} \rightarrow^{15} Y_{\{4-7,12-15,52-55,60-63\}} \tag{11}$$

Comparing Eq.(10) with Eq.(11), the number of balanced bits increases in Eq.(10). In this way, upper bound of integral distinguisher can not be obtained if we do not consider other integral distinguisher than m-th order one. Therefore, we conclude that the conventional algorithm is not appropriate in searching for upper bound of integral distinguisher.

4. PROPOSAL ALGORITHM

In the proposal algorithm, we obtain upper bound of integral distinguisher by executing computer experiment effectively. First, we obtain an input condition with lower order from one with 63rd order (theoretical step). Next, we execute computer experiment in such input condition (experimental step). From the above results, we can obtain upper bound of integral distinguisher. We show the outline in Sec.4.7 and the procedure in Sec.4.8.

4.1. OUTLINE OF PROPOSAL ALGORITHM

We consider an input condition with 63rd order as start point, and let X_{A0} be such input condition ($jA_0j = 63$). When we input X_{A0} to E^1, we consider new input condition in a set of such outputs. Let X_{A1} ($jA_0j = jA_1j = 63$) be such input condition, and it satisfies $X_{A0} \Supset X^1_{A1}$ (see Definition 1). If we can not obtain such input condition, we define new input condition $X_{A'0}$, s.t., $X_{A0} \Supset X_{A'0}$ and obtain X_{A1}, s.t., $X_{A'0} \Supset X^1_{A1}$. We repeat the above procedure to obtain input condition in which we can execute computer experiment in realistic time.

We divide E^1 into bijective partial functions such as $F(.)$ and $\phi(.)$. First, we consider the function $F(.)$. When all input bits to $F(.)$ are variable, we can regard all output bits as variable, since all elements of F^m_2 are outputted without multiplicity. Otherwise, we assume that all input bits are constant. Then, all output bits of $F(.)$ become constant and this partial function does not influence the whole integral distinguisher. We call such variable bits assumed to be constant as *redundant variable bits*.

Next, we consider the function $\phi(.)$. Since all input bits to $\phi(.)$ are variable, we can regard all output bits as variable. Also, we consider the function $\phi'(.)$ defined by Eq.(6). If x_0 is constant and x_1 is variable, all elements of F^m_2 are outputted without multiplicity in x_1'. Therefore, variable bits in x_0 become redundant variable bits when x_1 includes only variable and x_0 includes both variable and constant bits. Also, all variable bits become redundant variable bits when x_1 includes both variable and constant bits. We summarize the above procedures in Algorithm 1.

Applying each function of Algorithm 1 to input condition X_{A0}, we can obtain $X_{A'0}$, s.t., $X_{A0} \Supset X_{A'0}$. Then, we obtain X_{A1}, s.t., $X_{A'0} \Supset X^1_{A1}$ by considering positions of input and output bits of these partial functions. Repeating the above procedure, the order decreases gradually. If the order becomes one which is executable in realistic time (about a month) by computer experiment , we end the repetition.

In input condition obtained in the theoretical step, we execute computer experiment in the experimental step. At the experiment, we have to determine the number of times to execute computer experiment in the same input condition. As mentioned in Sec.2.2, integral distinguisher holds for any values of constant bits in input and round keys. By repeating computer experiment by chang-ing constant values, we can eliminate unbalanced bits which become 0 (*false balanced bits*). In integrated values, false balanced bits become 0 with probability 2^{1}. To distinguish false balanced bits with true ones, we execute computer experiment for 10 times by choosing random values for constant bits and round keys (false probability is 2^{10}).

Suppose we obtain $X_{A0} \ni X^{\gamma'}_{A'\,t}$ in the theoretical step and $X_{A'\,t} \rightarrow^{\gamma} Y_B$ in the experimental step. From Property 2, we can obtain upper bound of integral distinguisher $X_{A0} \rightarrow^{\gamma+\gamma'} Y_{B.}$

4.2. PROCEDURE OF PROPOSAL ALGORITHM

We show the whole procedure in Algorithm 2. A function *LowerOrderSearch* is the theoretical step and we call Algorithm 3. Another function *Experiment* is the experimental step and we call Algorithm 4. In the following, we explain them respectively.

4.2.1. MAIN ROUTINE

We suppose 2^{n_t} times encryptions can be executed in a month, where n_t is considered in Sec.5. In the end of the search, all 63rd order integral distinguisher is stored in an array D, e.g., $X_A \rightarrow^{\gamma} Y_B$ is stored as $D[A] = (B, \gamma)$. Let A_0 be an index set of variable bits, s.t., $|A_0| = 63$ and A_t be one with less variable bits ($|A_t| < 63$). We obtain A_t by *LowerOrderSearch* of Algorithm 3, and we execute computer experiment in input condition X_{A_t} in *Experiment* of Algorithm 4. Using Property 2, a result of computer experiment $D'[A_t]$ is stored as $D[A_0] = (D'[A_t], \gamma + \gamma')$.

Algorithm 1 Elimination of redundant variable bits.

```
procedure CutVariableBitF(A)
    if $|A| = m$ then                                ▹ $F(\cdot)$ outputs all elements of $\mathbb{F}_2^m$.
        $A' \Leftarrow A$
    else
        $A' \Leftarrow \{\}$
    end if
return $A'$
end procedure
procedure CutVariableBitϕ(A)
    if $|A| = 2m$ then                               ▹ $\phi(\cdot)$ outputs all elements of $\mathbb{F}_2^{2m}$.
        $A' \Leftarrow A$
    else                                             ▹ $\{A_0, A_1\}$ denotes variable bits in $\{x_0, x_1\}$.
        $\{A_0, A_1\} \Leftarrow A$
        if $|A_0| = m$ then                          ▹ $\phi'(\cdot)$ outputs all elements of $\mathbb{F}_2^m$.
            $A'_0 \Leftarrow A_0, A'_1 \Leftarrow \{\}$
            $A' \Leftarrow \{A'_0, A'_1\}$
        else                                         ▹ Bijection of $\phi$ can not be exploited.
            $A' \Leftarrow \{\}$
        end if
    end if
return $A'$
end procedure
```

Algorithm 2 Search Algorithm for Integral Distinguisher

```
procedure Main($n_t$)
    Let $D$ and $D'$ be arrays with one input and two outputs.
    for $a_c = 0 \rightarrow 63$ do                  ▹ e.g., $D[A] = (B, \gamma)$
        $A_0 = \{0, 1, ..., 63\} \backslash \{a_c\}$
        $(A_t, \gamma) \Leftarrow LowerOrderSearch(A_0, n_t)$
        if $D'[A_t] = NULL$ then
            $D'[A_t] \Leftarrow Experiment(A_t)$
            $D[A_0] \Leftarrow (D'[A_t], \gamma + \gamma')$
        else
            $D[A_0] \Leftarrow (D'[A_t], \gamma + \gamma')$
        end if
    end for                                          ▹ Use Property 2.
    return $D$
end procedure
```

4.2.2. THEORETICAL STEP

In *LowerOrderSearch*, we search for an input condition whose order is less than or equal to n_t. A round function E^1 is partitioned into $\{f_0, f_1, ..., f_{m-1}\}$, and A_0 is also partitioned into $\{A_{f_0}, A_{f_1}, ..., A_{f_{m-1}}\}$ following input bits to each function. Let *CutVariableBit* be a function to eliminate redundant variable bits, and it consists of functions in Algorithm 1. Redundant variable bits are eliminated from each A_{f_i} and substitute it for A'_{f_i}. Let A'_0 be a concatenation of $\{A'_{f_0}, A'_{f_1}, .., A'_{f_{m-1}}\}$, and an index A_t, s.t., $\mathcal{X}_{A_t} = E^1(\mathcal{X}_{A'_0}, K^1)$ is obtained. We define *OutputE1* by a function to obtain index of output variable bits when a set of plaintexts is inputted. Note that, it is not necessary to calculate $E^1(\mathcal{X}_{A_0}, K^1)$, since A_t is easily obtained by considering input-output relation of E^1.

Algorithm 3 Theoretical Step

```
function LowerOrderSearch($A_0, n_t$)
    $\{f_0, f_1, ..., f_{M-1}\} \Leftarrow E^1$                          ▷ Partition $E^1$.
    $\gamma = 0$
    $n = 63$
    while $n > n_t$ do
        $\{A_{f_0}, A_{f_1}, .., A_{f_{M-1}}\} \Leftarrow A_0$                 ▷ Partition $A_0$.
        for $i = 0 \to M - 1$ do                                ▷ Use Algorithm1.
            $A'_{f_i} \Leftarrow CutVariableBit(A_{f_i})$
        end for
        $A'_0 \Leftarrow \{A'_{f_0}, A'_{f_1}, .., A'_{f_{M-1}}\}$
        $A_t \Leftarrow OutputE1(A'_0)$                              ▷ $\mathcal{X}_{A'_0} \Supset \mathcal{X}^1_{A_t}$
        if $|A_t| = 0$ then
            return $(A_0, \gamma)$                        ▷ Output $A_0$ obtained previously.
        end if
        $n \Leftarrow |A_t|$
        $\gamma \Leftarrow \gamma + 1$
        $A_0 \Leftarrow A_t$
    end while
    return $(A_t, \gamma)$
end function
```

While the order is more than n_t, we repeat the routine. The order is gradually reduced, and we obtain an input integral whose order is less than or equal to n_t. By the above repetition, the order decreases and output *At, s.t.,|At|≤nt.*

4.2.3. EXPERIMENTAL STEP

In *Experiment*, we execute computer experiment to obtain output properties from input conditions. We parallelize and accelerate computer experiment by GPGPU. See Sec.5 in the details. First, we prepare a set of chosen plaintexts $\mathcal{X}_{A_t}$. Next, we encrypt them and XOR them with a data $S \in \mathbb{F}_2^{64}$. We parallelize the above procedure, and a total process is divided into T processes. Description **in parrallel do** in Algorithm 4 shows a routine which is parallelized. Note that unit of parallelized process is called *thread*. In each thread, $2^{|A_t|}/T$ chosen plaintexts are encrypted and outputs are XORed with $S_X \in \mathbb{F}_2^{64}$ $(0 \leq X \leq T-1)$. Thereafter, integration of all S_X is calculated as $S = \bigoplus_{X=0}^{T-1} S_X$. We repeat the above procedure for 10 times as mentioned in Sec.4.1, and these outputs are ORed with a data $S' \in \mathbb{F}_2^{64}$ to obtain bits which are always 0 as balanced bits. Finally, an index of balanced bits in S' and its round are outputted as B and γ'.

We execute the above procedure while we have at least one balanced bit with increasing γ'. We initialize $\gamma' = 1$ in Algorithm 4, however, we can substitute any values. By guessing rounds which integral distinguisher holds from previous results, we can set an appropriate initial value in γ' to shorten processing time.

4.3. Comparison with Conventional Algorithm

The conventional algorithm has a problem shown in Sec.3.2, since upper bound of integral distinguisher is obtained as extension of *m*-th order one. On the other hand, we obtain it using relatively higher order integral distinguisher. The input condition with relatively higher order includes many input conditions including one of *m*-th order. Therefore, we can obtain upper bound of integral distinguisher which considers many integral distinguisher.

Algorithm 4 Experimental Step

```
function Experiment($A_t$)
    $\gamma' = 1$                                         ▷ $\gamma'$ is not necessarily be initialized to 1.
    $n = |A_t|$                                           ▷ Let $\hat{A}_t$ be supplement set of $A_t$.
    $\hat{A}_t = \{0, 1, \dots, 63\} \backslash A_t$
    while $S' = 1^{64}$ do                                ▷ $1^{64}$ denotes all elements are 1.
        $S \Leftarrow 0$
        for $count = 0 \rightarrow 9$ do
            $C = \Leftarrow random, K \Leftarrow random$
            for all $X \leftarrow 0$ to $T - 1$ in parallel do
                for $x = 0 \rightarrow 2^n/T - 1$ do
                    $V \Leftarrow T * X + x$                      ▷ $p, v, c \in \mathbb{F}_2$
                    $v_0||v_1||\cdots||v_{n-1} \Leftarrow V$
                    $c_0||c_1||\cdots||c_{63-n} \Leftarrow C$
                    for $i = 0 \rightarrow n - 1$ do
                        $p_{A_t[i]} \Leftarrow v_i$
                    end for                                ▷ $A_t[i]$ denotes $i$-th element.
                    for $j = 0 \rightarrow 63 - n$ do
                        $p_{\hat{A}_t[j]} \Leftarrow c_j$
                    end for
                    $P_X \Leftarrow p_0||p_1||\cdots||p_{63}$
                    $P'_X \Leftarrow E^{\gamma'}(P_X, K^{\gamma})$
                    $S_X \Leftarrow S_X \oplus P'_X$
                end for
                $S \Leftarrow S \oplus S_X$
            end for
            $S' \Leftarrow S' \cup S$                      ▷ Only balanced bits are 0 in $S'$.
        end for
        $\gamma' \Leftarrow \gamma' + 1$
    end while
    Let $B$ be index of bits which are 0 in $S'$.
    return $(B, \gamma')$
end function
```

Table 2: Specification of GPGPU.

CPU	Intel Xenon E5-2620 2.00GHz
Memory	6GB
GPU	NVIDIA Tesla K20Xm×5
CUDA	version 6.00

In addition, we can apply the proposal algorithm to any block ciphers. Hence, the proposal algorithm is more effective and versatile algorithm to search for upper bound of integral distinguisher.

5. Acceleration of Computer Experiment by GPGPU

The more we can accelerate the experiment, the bigger n_t we can take in Algorithm 2. Table 2 shows a specification of our GPGPU environment. In the implementation of block ciphers in GPGPU, we consider three parameters needed for acceleration in Sec.5.1. In Sec.5.2, we execute computer experiment in HIGHT [3], LBlock [5], TWINE [8], PRESENT [1] and RECTANGLE

Table 3: The optimal numbers of **Parameter 2** and **3**.

thread-blocks	total threads	execution time[s]
1	2^{10}	8.705
2	2^{11}	4.350
2^2	2^{12}	2.178
2^3	2^{13}	1.089
2^4	2^{14}	0.544
2^5	2^{15}	0.506
2^6	2^{16}	0.500
2^7	2^{17}	0.479
2^8	2^{18}	0.453
2^9	2^{19}	0.443
$\underline{2^{10}}$	$\underline{2^{20}}$	$\underline{0.441}$
2^{11}	2^{21}	0.441
2^{12}	2^{22}	0.441

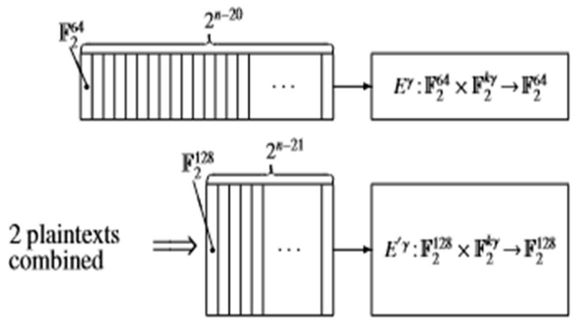
$\mathbb{F}_2^{64}$
2^{n-20}
$E^{\gamma}: \mathbb{F}_2^{64} \times \mathbb{F}_2^{k\gamma} \rightarrow \mathbb{F}_2^{64}$
$\mathbb{F}_2^{128}$
2^{n-21}
2 plaintexts combined
$E'^{\gamma}: \mathbb{F}_2^{128} \times \mathbb{F}_2^{k\gamma} \rightarrow \mathbb{F}_2^{128}$

Figure 3: Abstract of 2 plaintexts are combined into one data.

[10] and show their execution times.

5.1. IMPROVEMENT OF IMPLEMENTATION

We consider three parameters needed for acceleration of experiment.
Parameter 1: the number of threads for 2^n times encryptions.
Parameter 2: the number of plaintexts simultaneously encrypted in a thread.
Parameter 3: the number of S-boxes for optimization of lookup tables (LUT).

Parameter 1 is the number of threads. In CUDA complier, we can set the number of threads. Also, multiple threads share memories in a block called *thread-block*. In the experiment platform shown in Table 2, we can set 1; 024 threads in a thread-block at most. Therefore, we set 1; 024 threads in each thread-block. We determine the optimal number of thread-blocks by measuring execution times needed for 2^{32} times encryptions. To measure execution times, we use a round function of TWINE [8]. And the number of thread-blocks is doubled for each time, since 2^{32} must be divisible. Table 5 shows execution times for each number of thread-blocks. From 1 to 2^{10}, execution time is shortened, however, it is unchanged after 2^{11}. Since the number of thread-blocks is saturated in 2^{10}, we determine the total number of threads as 2^{10} 2^{10} = 2^{20}.

Parameter 2 is the number of plaintexts simultaneously encrypted in each thread. It is possible to accelerate by combining multiple plaintexts into one data and encrypt this data at one time in each thread. Fig.3 illustrates an example that 2 plaintexts are combined into one date in each thread. If *d* plaintexts are combined, the number of encryptions in one thread becomes $\lceil 2^{nt\ 20}=d \rceil$ ($\lceil\ \rceil$ ceiling function). We use a characteristic of processors that processing time does not increase

Table 4: The optimal numbers of **Parameter 2** and **3**.

block cipher	Parameter 2	Parameter 3
HIGHT	2	-
LBlock	2	8
TWINE	2	3
PRESENT	2	2
RECTANGLE	2	-

Table 5: Execution time in 40th order.

cipher	rounds	execution time	execution time/round(t_{40})
		[h:m:s]	[h:m:s]
HIGHT	32	0:24:27	0:00:46
LBlock	31	0:19:08	0:00:37
TWINE	36	0:21:24	0:00:36
PRESENT	31	1:09:29	0:02:14
RECTANGLE	25	0:07:49	0:00:19

in proportion to the number of processing bits. For example, LUT of 8-bit S-box does not require twice as long time as one of 4-bit S-box.

Parameter 3 is the number of S-boxes for optimization of LUT. Since LUT directly influence the whole execution time, the optimal memory arrangement of S-boxes is important. We use two types of memories; register and shared memory. The register is a memory for a thread, and it has lowest latency in memory access. The shared memory is a memory shared in a thread-block. Although the register is smaller than shared memory, we use it as much as possible for acceleration. However, S-boxes have so much capacity that can not be arranged in the register. Therefore, we should store them in the shared memory. Since a memory for a S-box is accessed by multiple threads, resource contention occurs frequently and latency increases. In this paper, we arrange some duplicated S-boxes in each shared memory to mitigate resource contention. However, latency increases by using shared memory too much for S-boxes. Considering such trade-off, we determine the optimal number of S-boxes by executing computer experiment.

In Table 4, we summarize the optimal numbers of **Parameter 2** and **3** in HIGHT, LBlock, TWINE, PRESENT and RECTANGLE.

5.2. Execution Time of Experiment

By considering three parameters shown in Sec.5.1, we execute computer experiment in HIGHT, LBlock, TWINE, PRESENT and RECTANGLE. Table 5 shows times to execute 2^{40} times encryp-tions based on Algorithm 4. Note that we use full rounds cipher function. Since the numbers of rounds are different, we calculate execution times per a round. As for Fesitel structures (HIGHT, LBlock and TWINE), similar execution times are measured.

As for SPN structures (PRESENT and RECTANGLE), RECTANGLE is much faster than PRESENT. As mentioned in [19], PRESENT is not suitable for software implementation. Also, RECTANGLE can be accelerated by using special implementation shown in [10]. S-box operation of RECTANGLE is replaced with AND, OR and XOR operations among 16-bit data.

Let t_{40} be an execution time of a round function in 40th order shown in Table 5. Actual execution time to test n_t-th order input integral with γ rounds cipher function is approximately calculated as *t40 ×γ×2nt−40 ×10[sec]*. In PRESENT, it takes 123 days to execute computer experiment in input condition with 48-th order. However, we use 5 GPGPU machines in our experiment platform, so

Table 6: Previous results of HIGHT, LBlock, TWINE, PRESENT and RECTANGLE

block cipher	order	rounds	balanced bits	citation
HIGHT	56	17	1	[20]
LBlock	60	15	24	[21]
TWINE	60	15	24	[18]
PRESENT	16	7	1	[22]
RECTANGLE	56	7	4	[10]

Table 7: Results of HIGHT, LBlock, TWINE, PRESENT and RECTANGLE by the proposal algorithm.

block cipher	rounds	balanced bits
HIGHT	17	1
LBlock	15	$\underline{32}$
TWINE	15	$\underline{32}$
PRESENT	7	1
RECTANGLE	$\underline{8}$	$\underline{62}$

that, we can reduce it to one fifth. Therefore, we can execute computer experiment in 48th order within a month in all of 5 block ciphers.

6. APPLICATIONS OF PROPOSAL ALGORITHM

We apply the proposal algorithm to light-weight block ciphers, HIGHT [3], LBlock [5], TWINE [8], PRESENT [1] and RECTANGLE [10]. Due to the limited space, we omit the details of each cipher. At first, we summarize previous results in Table 6. We only show the results which are the most advantageous for the attackers. In other word, they are integral distinguisher whose number of rounds and balanced bits are the most. By application of the proposal algorithm, we obtain results shown in Table 7. The same as Table 6, we only show the most advantageous ones for the attackers. Underlined parameters are ones which are advanced from the previous results. In the following, we show the detail of the results, respectively.

HIGHT: Zheng et al. showed following 56th order integral distinguisher [20].

$$X_{\{\overline{34-31}\}} \rightarrow^{17} Y_{\{7\}},\ \ X_{\{\overline{56-63}\}} \rightarrow^{17} Y_{\{3\}} \tag{12}$$

Based on the conventional algorithm, these results are obtained as extensions of 8th order integral distinguisher. On the other hand, we obtain following results by the proposal algorithm.

$$0 \le i \le 7,\ X_{\{\overline{24+i}\}} \rightarrow^{17} Y_{\{7\}},\ \ X_{\{\overline{56+i}\}} \rightarrow^{17} Y_{\{3\}} \tag{13}$$

These results are almost the same as the previous result (Eq.(12)).

LBlock: Shibayama et al. showed following 60th order integral distinguisher [21].

$$\begin{aligned}
&X_{\overline{\{0-3\}}}, X_{\overline{\{12-15\}}} \rightarrow^{15} Y_{\{32-51,56-59\}} \\
&X_{\overline{\{4-7\}}}, X_{\overline{\{28-31\}}} \rightarrow^{15} Y_{\{36-55,60-63\}} \\
&X_{\overline{\{12-15\}}}, X_{\overline{\{20-23\}}} \rightarrow^{15} Y_{\{32-39,44-47,52-63\}} \\
&X_{\overline{\{16-19\}}}, X_{\overline{\{24-27\}}} \rightarrow^{15} Y_{\{32-35,40-43,48-63\}}
\end{aligned} \qquad (14)$$

These results are obtained as extensions of 12th order integral distinguisher searched by computer experiment. On the other hand, we obtain following results by the proposal algorithm.

$$0 \le i \le 31,\ X_{\{\bar{i}\}} \rightarrow^{15} Y_{\{32-63\}} \qquad (15)$$

The number of balanced bits is increased from 24 to 32 compared with the previous results (Eq.(14)).

TWINE Shibayama et al. showed following 60th order integral distinguisher [18].

$$\begin{aligned}
&X_{\overline{\{0-3\}}}, X_{\overline{\{40-43\}}} \rightarrow^{15} Y_{\{4-7,12-15,20-23,28-31,52-55,60-63\}} \\
&X_{\overline{\{8-11\}}}, X_{\overline{\{24-27\}}} \rightarrow^{15} Y_{\{12-15,20-23,28-31,36-39,44-47,60-63\}} \\
&X_{\overline{\{28-31\}}}, X_{\overline{\{48-51\}}} \rightarrow^{15} Y_{\{4-7,12-15,36-39,44-47,52-55,60-63\}} \\
&X_{\overline{\{32-35\}}}, X_{\overline{\{56-59\}}} \rightarrow^{15} Y_{\{4-7,20-23,28-31,36-39,44-47,52-55\}}
\end{aligned} \qquad (16)$$

These results are obtained as extensions of 12th order integral distinguisher searched by computer experiment. On the other hand, we obtain following results by the proposal algorithm.

$$\begin{aligned}
&0 \le i \le 7, 0 \le j \le 3, \\
&X_{\overline{\{8i+j\}}} \rightarrow^{15} Y_{\{4-7,12-15,20-23,28-31,36-39,44-47,52-55,60-63\}}
\end{aligned} \qquad (17)$$

As same as LBlock, the number of balanced bits is increased from 24 to 32 compared with the previous results (Eq.(16)).

PRESENT: Wu et al. showed following 16th order integral distinguisher [22].

$$X_{\{48-63\}} \rightarrow^{7} Y_{\{63\}} \qquad (18)$$

The result is obtained by some algebraic properties of S-box. On the other hand, we obtain following results by the proposal algorithm.

$$0 \le i \le 63,\ X_{\{\bar{i}\}} \rightarrow^{7} Y_{\{63\}} \qquad (19)$$

These results are almost the same as the previous result (Eq.(18)).

However, 16th order integral distinguisher is more advantageous for the attackers, since it needs far less chosen planitexts.

RECTANGLE: Designers of RECTANGLE showed following 56th order integral distinguisher [10].

$$X_{\overline{\{24-27,60-63\}}} \rightarrow^{7} Y_{\{41,44,59,62\}} \tag{20}$$

The result is derived as an extension of 1st order integral distinguisher. On the other hand, we obtain following results by the proposal algorithm.

$$\begin{gathered} 0 \leq i \leq 15, 0 \leq j \leq 3, \\ X_{\overline{\{4i+j\}}} \rightarrow^{8} Y_{\overline{\{23+4i \bmod 64,\ 39+4i \bmod 64\}}}, \end{gathered} \tag{21}$$

where {23 + 4i mod 64, 39 + 4i mod 64} denotes 62 bits are balanced other than 2 bits calculated by substituting i. These results are more advantageous results than the previous result (Eq.(20)), since the number of rounds which balanced bits exist and one of balanced bits increase.

7. Conclusion

We propose a new algorithm to search for integral distinguisher based on computer experiment by GPGPU. In the theoretical step, input condition whose order is less than or equal to n_t is obtained. When all of 63rd order integral distinguisher is searched, we need to execute $64 \times 2^{n_t} \times 10$ times encryptions at most. If we test all possible n_t-th order input conditions, $\binom{64}{n_t} \times 2^{n_t}$ times encryptions are needed. Therefore, we can reduce computational complexity by using the theoretical step. In the experimental step, we accelerate computer experiment by GPGPU. From acceleration by GPGPU and some improvements of implementation, we achieve to test 2^{48} times encryptions of 5 light-weight block ciphers within a month. We apply the proposal algorithm to search for 63rd order integral distinguisher of HIGHT, LBlock, TWINE, PRESENT and RECTANGLE. In all ciphers, we obtain the same or more advantageous upper bound of integral distinguisher compared with previous results.

Designers of block ciphers must consider upper bound of integral distinguisher to decide the security margin of ciphers. There is a possibility that the number of rounds to be attacked increases by upper bound of integral distinguisher. Even if it is not, it can be less difficult to guess all of the secret keys. Therefore, the designers need to consider such vulnerabilities and select stronger cipher function and key schedule. Especially in light-weight block cipher, necessary security margin of cipher function must be obtained, since there must be trade-off between fastness and security.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 24560491.

References

[1] A. Bogdanov, L. Knudsen, G. Leander, C. Paar, A. Poschmann, M. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT: An ultra-lightweight block cipher," in Cryptographic Hardware and Embedded Systems - CHES 2007, ser. Lecture Notes in Computer Science, P. Paillier and I. Verbauwhede, Eds. Springer Berlin Heidelberg, 2007, vol. 4727, pp. 450–466. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-74735-2 31

[2] L. Knudsen, "CLEFIA," in Encyclopedia of Cryptography and Security, H. van Tilborg and S. Jajodia, Eds. Springer US, 2011, pp. 210–211. [Online]. Available: http://dx.doi.org/10.1007/978-1-4419-5906-5 561

[3] D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B.-S. Koo, C. Lee, D. Chang, J. Lee, K. Jeong, H. Kim, J. Kim, and S. Chee, "HIGHT: A new block cipher suitable for low-resource device," in Cryptographic Hardware and Embedded Systems - CHES 2006, ser. Lecture Notes in Computer Science, L. Goubin and M. Matsui, Eds. Springer Berlin Heidelberg, 2006, vol. 4249, pp. 46–59. [Online]. Available: http://dx.doi.org/10.1007/11894063 4

[4] Z. Gong, S. Nikova, and Y. Law, "KLEIN: A new family of lightweight block ciphers," in RFID. Security and Privacy, ser. Lecture Notes in Computer Science, A. Juels and C. Paar, Eds. Springer Berlin Heidelberg, 2012, vol. 7055, pp. 1–18. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-25286-0 1

[5] W. Wu and L. Zhang, "LBlock: A lightweight block cipher," in Applied Cryptography and Network Security, ser. Lecture Notes in Computer Science, J. Lopez and G. Tsudik, Eds. Springer Berlin Heidelberg, 2011, vol. 6715, pp. 327–344. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-21554-4 19

[6] K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai, "Piccolo: an ultra-lightweight blockcipher," in Cryptographic Hardware and Embedded Systems-CHES 2011. Springer, 2011, pp. 342–357.

[7] J. Guo, T. Peyrin, A. Poschmann, and M. Robshaw, "The LED block cipher," in Cryptographic Hardware and Embedded Systems - CHES 2011, ser. Lecture Notes in Computer Science, B. Preneel and T. Takagi, Eds. Springer Berlin Heidelberg, 2011, vol. 6917, pp. 326–341. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-23951-9 22

[8] T. Suzaki, K. Minematsu, S. Morioka, and E. Kobayashi, "TWINE: A lightweight block cipher for multiple platforms," in Selected Areas in Cryptography, ser. Lecture Notes in Computer Science, L. Knudsen and H. Wu, Eds. Springer Berlin Heidelberg, 2013, vol. 7707, pp. 339–354. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-35999-6 22

[9] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "The SIMON and SPECK families of lightweight block ciphers." IACR Cryptology ePrint Archive, vol. 2013, p. 404, 2013.

[10] W. Zhang, Z. Bao, D. Lin, V. Rijmen, B. Yang, and I. Verbauwhede, "RECTANGLE: A bit-slice ultra-lightweight block cipher suitable for multiple platforms," Cryptology ePrint Archive, Report 2014/084, 2014. [Online]. Available: http://eprint.iacr.org/

[11] J. Daemen, L. Knudsen, and V. Rijmen, "The block cipher Square," in Fast Software Encryption, ser. Lecture Notes in Computer Science, E. Biham, Ed. Springer Berlin Heidelberg, 1997, vol. 1267, pp. 149–165. [Online]. Available: http://dx.doi.org/10.1007/BFb0052343

[12] L. Knudsen and D. Wagner, "Integral cryptanalysis," in Fast Software Encryption, ser. Lecture Notes in Computer Science, J. Daemen and V. Rijmen, Eds. Springer Berlin Heidelberg, 2002, vol. 2365, pp. 112–127. [Online]. Available: http://dx.doi.org/10.1007/3-540-45661-9 9

[13] M. Matsui, "New block encryption algorithm MISTY," in Fast Software Encryption, ser. Lecture Notes in Computer Science, E. Biham, Ed. Springer Berlin Heidelberg, 1997, vol. 1267, pp. 54–68. [Online]. Available: http://dx.doi.org/10.1007/BFb0052334

[14] Y. Todo, "Integral cryptanalysis on full MISTY1," in Advances in Cryptology – CRYPTO 2015, ser. Lecture Notes in Computer Science, R. Gennaro and M. Robshaw, Eds. Springer Berlin Heidelberg, 2015, vol. 9215, pp. 413–432. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-47989-6 20

[15] Q. Wang, Z. Liu, K. Varıcı, Y. Sasaki, V. Rijmen, and Y. Todo, "Cryptanalysis of reduced-round simon32 and simon48," in Progress in Cryptology–INDOCRYPT 2014. Springer, 2014, pp. 143–160.

[16] N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, and D. Whiting, "Improved cryptanalysis of Rijndael," in Fast Software Encryption, ser. Lecture Notes in Computer Science, G. Goos, J. Hartmanis, J. van Leeuwen, and B. Schneier, Eds. Springer Berlin Heidelberg, 2001, vol. 1978, pp. 213–230. [Online]. Available: http://dx.doi.org/10.1007/3-540-44706-7 15

[17] Sony Corporation, "The 128-bit blockcipher CLEFIA security and performance evaluations revision 1.0," http://www.sony.net/Products/cryptography/clefia/download/data/clefia/eval/ 1.0.pdf, 2007.

[18] N. Shibayama and T. Kaneko, "New higher order differential property of TWINE," in Sym-posium on Cryptography and Information Security, SCIS2015,1D2-2, 2015.

[19] R. Benadjila, J. Guo, V. Lomne,´ and T. Peyrin, "Implementing lightweight block ciphers on x86 architectures," in Selected Areas in Cryptography–SAC 2013. Springer, 2014, pp. 324–351.

[20] P. Zhang, B. Sun, and C. Li, "Saturation attack on the block cipher HIGHT," in Cryptology and Network Security, ser. Lecture Notes in Computer Science, J. Garay, A. Miyaji, and A. Otsuka, Eds. Springer Berlin Heidelberg, 2009, vol. 5888, pp. 76–86. [Online]. Available:

http://dx.doi.org/10.1007/978-3-642-10433-6 6

[21] N. Shibayama and T. Kaneko, “A new higher order differential of LBlock,” in Information Theory and its Applications (ISITA), 2014 International Symposium on. IEEE, 2014, pp. 488– 492.

[22] S. Wu and M. Wang, “Integral attacks on reduced-round PRESENT,” in Information and Communications Security, ser. Lecture Notes in Computer Science, S. Qing, J. Zhou, and D. Liu, Eds. Springer International Publishing, 2013, vol. 8233, pp. 331–345. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-02726-5 24