

Improved Integral Attack on HIGHT

Yuki Funabiki¹(✉), Yosuke Todo², Takanori Isobe³, and Masakatu Morii¹

¹ Kobe University, 1-1 Rokkodai-cho, Nada-ku, Kobe, Hyogo 657-8501, Japan

funabiki@stu.kobe-u.ac.jp, mmorii@kobe-u.ac.jp

² NTT Secure Platform Laboratories, 3-9-11 Midori-cho,

Musashino, Tokyo 180-8585, Japan

todo.yosuke@lab.ntt.co.jp

³ University of Hyogo, 7-1-28 Minatojima-minamimachi,

Chuo-ku, Kobe, Hyogo 650-0047, Japan

takanori.isobe@ai.u-hyogo.ac.jp

Abstract. HIGHT is a lightweight block cipher with 64-bit block length and 128-bit security, and it is based on the ARX-based generalized Feistel network. HIGHT became a standard encryption algorithm in South Korea and also is internationally standardized by ISO/ICE 18033-3. Therefore, many third-party cryptanalysis against HIGHT have been proposed. Especially, impossible differential and integral attacks are applied to reduced-round HIGHT, and the current best attack under the single-key setting is 27 rounds using the impossible differential attack. In this paper, we propose an improved integral attack against HIGHT. We first propose new 19-round integral characteristics by using the propagation of the division property, and they are improved by two rounds compared with previous integral characteristics. Finally, we can attack 28-round HIGHT by appending 9-round key recovery. Moreover, we can attack 29-round HIGHT if the full code book is used, and it improves by two rounds compared with previous best attack.

Keywords: Block cipher · HIGHT · Integral attack · Division property · Partial-sum technique · Bitwise partial-sum technique · Meet-in-the-middle technique

1 Introduction

The lightweight cryptography is one of the most actively discussed topics in the community of symmetric-key cryptographers. The motivation of the lightweight symmetric-key cryptography is to design high-performance and secure symmetric-key ciphers under the area-constraining environments. Such ciphers are expected to be proper for radio frequency identification (RFID), sensor network, and Internet of Things (IoT). Nowadays, a huge number of such ciphers have been proposed, and please refer to [2], where a list of lightweight ciphers is well summarized.

The generalized Feistel network (GFN) is suited to the design of lightweight block ciphers because each F-function is very small. LBlock [23] and TWINE [19]

Table 1. Comparison of attack results on HIGHT.

Model	Attack	#Rounds	Data	Time	Reference
Single key	Imp. Diff	18	$2^{46.8}$	$2^{109.2}$	[7]
	Imp. Diff	26	2^{61}	$2^{119.53}$	[12]
	Imp. Diff	26	$2^{61.6}$	$2^{114.35}$	[3]
	Imp. Diff	27	2^{58}	$2^{126.6}$	[3]
	Integral	16	2^{42}	2^{51}	[7]
	Integral	22	2^{62}	$2^{118.71}$	[25]
	Integral	22	2^{62}	$2^{102.35}$	[14]
	Integral	26	2^{57}	$2^{120.55}$	[15]
	Integral	28	2^{63}	2^{127}	Sect. 4
	Integral	29	2^{64}	2^{126}	Sect. 5
Related key	Imp. Diff	31	2^{64}	$2^{127.28}$	[12]
	rectangle	32 (full)	$2^{57.84}$	$2^{125.83}$	[10]

are examples of such ciphers. HIGHT, which was proposed by Hong et al. at CHES 2006 [7], is also a lightweight block cipher adopting the GFN. Moreover, HIGHT was standardized by ISO/IEC 18033-3 [8]. HIGHT only consists of three operations, i.e., modular additions over 256, bitwise rotations, and bitwise XOR. Such structure is often called ARX, and HIGHT is regarded as an ARX-based generalized Feistel network. Some ARX-based ciphers have been proposed, but there are many unsolved problems in the security analysis compare with the S-box-based ciphers. Therefore, HIGHT standardized by ISO/IEC is one of the most attractive ARX-based ciphers and is well analyzed.

In the related-key setting, the full HIGHT was already broken using the related-key rectangle attack [10]. On the other hand, impossible differential and integral attacks have been often applied to HIGHT under the single-key setting, but the full HIGHT has not been attacked yet. The current best attack is proposed by Chen using the impossible differential attack and 27-round HIGHT is attacked [3] (Table 1).

In this paper, we propose the current best attack by using the improved integral attacks. The integral attack consists of two parts; an integral characteristic and key recovery. In the integral characteristic, attackers first prepare a set of chosen plaintexts, where the XOR of the part of all corresponding states after several encryption rounds is always 0 for all secret keys. Then, in the key recovery, they guess round keys used in the last several rounds and evaluate whether the XOR of partially decrypted texts is 0 or not. If the correct key is guessed, the XOR is 0 because of the integral characteristic. Therefore, if the XOR is not 0, the guessed round key is discarded.

The integral cryptanalysis on HIGHT was first evaluated by the designers [7]. They showed 12-round integral characteristics with 2^8 chosen plaintexts, and 16-round HIGHT is attacked by using the characteristic. However, the error of

this 12-round characteristics was pointed out by Zhang et al., and they showed that the correct integral characteristics with 2^8 chosen plaintexts cover only 11 rounds [25]. Moreover, they improved the 11-round characteristic to 17-round one by using the higher-order integral characteristics. As a result, 22-round HIGHT is attacked by using the 17-round characteristic. Then, the key recovery part is dramatically improved by Sasaki and Wang. They first proposed the meet-in-the-middle technique for the key recovery of the integral attack [14], which is useful to reduce the time complexity. Moreover, they proposed the bitwise partial-sum technique and optimized the key recovery [15]. As a result, 26-round HIGHT is attacked. Note that both improvements use the same 17-round characteristic by Zhang et al.

In this paper, we first show new 19-round integral characteristics, which is improved by two rounds than previous 17-round one. Our new characteristic is found by the propagation of the division property [21]. The division property is a general technique to find integral characteristics and recently applied to a wide range of block ciphers. New 18-round integral characteristics with 2^{63} chosen plaintexts are found by the propagation of the division property, and 18-round characteristics are extended to 19-round ones. Then, we show that 28-round HIGHT can be attacked by using this extended 19-round characteristic. Moreover, we show that 29-round HIGHT can be attacked by using the same characteristic if the full code book is used. Since the previous best attack is up to 27 rounds, our new attacks are the current best attack under the single-key setting.

2 Preliminaries

2.1 Specification of HIGHT

HIGHT is a block-cipher proposed at CHES 2006 by Hong et al. [7]. The block size is 64 bits and the key size is 128 bits. It adopts the type-2 generalized Feistel network with 8 branches and 32 rounds. Please refer to [7] for details. Note that a figure with an incorrect subkey order is showed in [7], and the designers later fixed the problem [1].

Encryption. The 64-bit plaintext and ciphertext are considered as concatenations of 8 bytes and denoted by $P = P_7 \| P_6 \| \dots \| P_0$ and $C = C_7 \| C_6 \| \dots \| C_0$, respectively. The input of the $(r + 1)$ -th round function is represented as $X^r = X_7^r \| X_6^r \| \dots \| X_0^r$ for $r = 0, 1, \dots, 32$. At first, the plaintext is loaded into an internal state $X_7^0 \| X_6^0 \| \dots \| X_0^0$ as follows.

$$\begin{aligned} X_0^0 &= P_0 \boxplus WK_0, & X_1^0 &= P_1, & X_2^0 &= P_2 \oplus WK_1, & X_3^0 &= P_3, \\ X_4^0 &= P_4 \boxplus WK_2, & X_5^0 &= P_5, & X_6^0 &= P_6 \oplus WK_3, & X_7^0 &= P_7, \end{aligned}$$

where WK_i denotes 8-bit whitening keys for $i = 0, 1, \dots, 7$. The operation \boxplus denotes addition mod 2^8 . Then, the value $X_7^r \| X_6^r \| \dots \| X_0^r$ is updated as Fig. 1

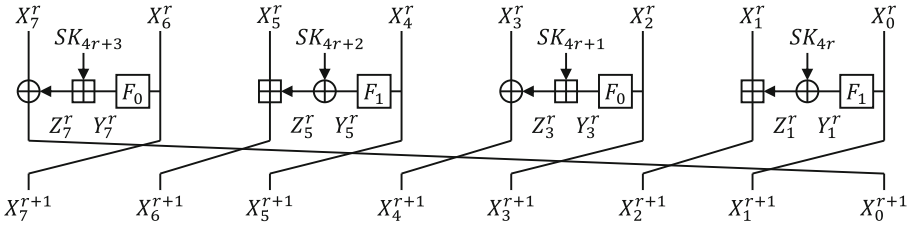


Fig. 1. Round function procedure of HIGHT

for $r = 0, 1, \dots, 31$, where $F_0(x) = (x \lll 1) \oplus (x \lll 2) \oplus (x \lll 7)$ and $F_1(x) = (x \lll 3) \oplus (x \lll 4) \oplus (x \lll 6)$. The operation about $(x \lll s)$ denotes an s -bit left rotation of an 8-bit value x , and SK_i denotes the i -th 8-bit subkey for $i = 0, 1, \dots, 127$. The swap of the byte position is omitted in the last round. The internal state between F and the key addition is defined by $Y_1^r, Y_3^r, Y_5^r, Y_7^r$, and the internal state after the key addition is defined by $Z_1^r, Z_3^r, Z_5^r, Z_7^r$. Finally, the ciphertext is generated from X^{32} by applying the post whitening as follows.

$$\begin{aligned}
 C_0 &= X_0^{32} \boxplus WK_4, & C_1 &= X_1^{32}, & C_2 &= X_2^{32} \oplus WK_5, & C_3 &= X_3^{32}, \\
 C_4 &= X_4^{32} \boxplus WK_6, & C_5 &= X_5^{32}, & C_6 &= X_6^{32} \oplus WK_7, & C_7 &= X_7^{32}.
 \end{aligned}$$

Decryption. The decryption process is explained in the similar to the encryption process. This operation is identical to an operation for encryption apart from the following two modifications.

1. All \boxplus operations are replaced by \boxminus operations except for the \boxplus operations connecting SK_i and outputs of F_0 , where the operation about \boxminus denotes subtraction mod 2^8 .
2. The order in which the keys WK_i and SK_i are applied is reversed.

Key Schedule. The 128-bit master key is considered as a concatenation of 16 bytes and denoted by $K = K_{15} \parallel K_{14} \parallel \dots \parallel K_0$. In the key schedule, 4 whitening keys for plaintexts are first generated from the master key as $(WK_0, WK_1, WK_2, WK_3) = (K_{12}, K_{13}, K_{14}, K_{15})$, and 4 whitening keys for ciphertexts are generated from the master key as $(WK_4, WK_5, WK_6, WK_7) = (K_0, K_1, K_2, K_3)$. Moreover, the 128 subkeys are generated as

$$\begin{cases}
 SK_{16 \cdot i + j} = K_{j-i \pmod 8} \boxplus \delta_{16 \cdot i + j}, \\
 SK_{16 \cdot i + j + 8} = K_{(j-i \pmod 8) + 8} \boxplus \delta_{16 \cdot i + j + 8},
 \end{cases}$$

where δ_i is a constant.

2.2 Integral Characteristics and Division Property

The integral attack was first proposed by Daemen et al. to evaluate the security of SQUARE [5], and then it was formalized by Knudsen and Wagner [9].

The integral attack consists of two parts; construction of an integral characteristic and key recovery. In this subsection, we focus on the first part, and the second part is described in the next subsection.

The most common integral characteristic exploits the set of chosen plaintexts such that the sum of chosen bits in texts encrypted a certain number of rounds is always 0 for all secret keys. Assume that m -bit encrypted texts hold this characteristic in the target block cipher. Then, since the probability that the ideal block cipher holds this characteristic is 2^{-m} , the distinguishing attack is directly derived from the integral characteristic.

Division Property. The division property, which was recently proposed in [21, 22], is a general method to find integral characteristics, and it is defined as follows.

Definition 1 (Division Property [21, 22]). Let \mathbb{X} be a multiset whose elements take a value of \mathbb{F}_2^n . When the multiset \mathbb{X} has the division property $\mathcal{D}_{\mathbb{K}}^{1^n}$, where \mathbb{K} denotes a set of m -dimensional vectors whose i -th element takes 0 or 1, it fulfills the following conditions:

$$\bigoplus_{\mathbf{x} \in \mathbb{X}} \mathbf{x}^{\mathbf{u}} = \begin{cases} \text{unknown} & \text{if there exist } \mathbf{k} \in \mathbb{K} \text{ s.t. } \mathbf{u} \succeq \mathbf{k}, \\ 0 & \text{otherwise,} \end{cases}$$

where $\mathbf{x}^{\mathbf{u}} = \prod_{i=1}^n x[i]^{u[i]}$, and $\mathbf{u} \succeq \mathbf{k}$ if $u[i] \geq k[i]$ for all i . Here, $x[i]$ denotes the i -th bit of x from the least significant bit (lsb).

Todo and Morii showed the propagation rules of the division property for three basic operations; copy, xor, and and [22].

Let $I = \{i_1, i_2, \dots, i_{|I|}\}$ be the index of active plaintext bits. Then, the division property of such chosen plaintexts becomes $\mathcal{D}_{\mathbf{k}}^{1^n}$, where $k_i = 1$ if $i \in I$ and $k_i = 0$ otherwise. Then, to search for integral characteristics, *division trail* is evaluated.

Definition 2 (Division Trail [24]). Let us consider the propagation of the division property

$$\{\mathbf{k}\} \stackrel{\text{def}}{=} \mathbb{K}_0 \rightarrow \mathbb{K}_1 \rightarrow \mathbb{K}_2 \rightarrow \dots \rightarrow \mathbb{K}_r,$$

where $\mathcal{D}_{\mathbb{K}_i}$ be the division property after i -round propagation. Moreover, for any vector $\mathbf{k}_{i+1}^* \in \mathbb{K}_{i+1}$, there must exist a vector $\mathbf{k}_i^* \in \mathbb{K}_i$ such that \mathbf{k}_i^* can propagate to \mathbf{k}_{i+1}^* by the propagation rule of the division property. Furthermore, for $(\mathbf{k}_0, \mathbf{k}_1, \dots, \mathbf{k}_r) \in (\mathbb{K}_0 \times \mathbb{K}_1 \times \dots \times \mathbb{K}_r)$ if \mathbf{k}_i can propagate to \mathbf{k}_{i+1} for all $i \in \{0, 1, \dots, r-1\}$, we call $(\mathbf{k}_0 \rightarrow \mathbf{k}_1 \rightarrow \dots \rightarrow \mathbf{k}_r)$ an r -round division trail.

Let E_k be the target r -round block cipher. Then, if there is no division trail $\mathbf{k}_0 \xrightarrow{E_k} \mathbf{k}_r = \mathbf{e}_i$, the i -th bit of r -round ciphertexts is always balanced. In [21], [20], and [22], all possible division trails are evaluated by using a breadth-first search. Unfortunately, it is practically infeasible to apply this method to block ciphers whose block length exceeds 32 because the size of \mathbb{K}_i is extremely large.

MILP-Aided Propagation Search. A mixed-integer linear programming (MILP) was introduced to cryptanalysis by Mouha et al. in [11]. Then, the MILP has been successfully applied to various cryptanalyses [4, 13, 17, 17, 18, 24]. The MILP is an optimization or feasibility program where variables are restricted to integers. An MILP model \mathcal{M} consists of variables $\mathcal{M}.var$, constraints $\mathcal{M}.con$, and an objective function $\mathcal{M}.obj$, and the following is an example of MILP.

Example 1.

$$\begin{aligned} \mathcal{M}.var &\leftarrow x, y, z \text{ as binary.} \\ \mathcal{M}.con &\leftarrow x + 2y + 3z \leq 4 \quad \text{and} \quad x + y \geq 1 \\ \mathcal{M}.obj &\leftarrow \text{maximize } x + y + 2z \end{aligned}$$

The answer of the model \mathcal{M} is 3, where $(x, y, z) = (1, 0, 1)$.

MILP solver can solve such optimization program, and it returns *infeasible* if there is no feasible solution. Moreover, if there is no objective function, the MILP solver only evaluates whether this model is feasible or not.

Xiang et al. showed that all division trails are efficiently evaluated by using the MILP in [24], where three division trails for basic operations are modeled as follows.

Proposition 1 (MILP model for COPY). *Let $a \xrightarrow{COPY} (b_1, b_2, \dots, b_m)$ be a division trail of COPY, where one bit is copied to m bits. The following inequalities are sufficient to describe the propagation of the division property for copy.*

$$\begin{cases} \mathcal{M}.var \leftarrow a, b_1, b_2, \dots, b_m \text{ as binary.} \\ \mathcal{M}.con \leftarrow a = b_1 + b_2 + \dots + b_m \end{cases}$$

Proposition 2 (MILP model for XOR). *Let $(a_1, a_2, \dots, a_m) \xrightarrow{XOR} b$ be a division trail of XOR, where the XOR of m bits is computed. The following inequalities are sufficient to describe the propagation of the division property for xor.*

$$\begin{cases} \mathcal{M}.var \leftarrow a_1, a_2, \dots, a_m, b \text{ as binary.} \\ \mathcal{M}.con \leftarrow a_1 + a_2 + \dots + a_m = b \end{cases}$$

Proposition 3 (MILP model for 2-bit AND). *Let $(a_1, a_2) \xrightarrow{AND} b$ be a division trail of AND, where the AND of 2 bits is computed. The following inequalities are sufficient to describe the propagation of the division property for and.*

$$\begin{cases} \mathcal{M}.var \leftarrow a_1, a_2, b \text{ as binary.} \\ \mathcal{M}.con \leftarrow b \geq a_i \text{ for all } i \in \{1, 2\} \end{cases}$$

In [24], an additional constraint $b - a_1 - a_2 \leq 0$ is used, but it is redundant. Namely, even if the redundant constraint is not used, it does not affect the result of MILP.

We first create the MILP model for a target block cipher by using Proposition 1, 2, and 3. Then, the division property of plaintexts is constrained according to the index I of active plaintext bits. Moreover, the division property of the i -th bit of ciphertexts is constrained to 1 when the i -th bit of ciphertexts is evaluated, and the division property of the other bits is constrained to 0. If the MILP solver judges that the model is infeasible, the i -th bit of ciphertexts is balanced. Please refer to [24] in detail.

2.3 Key Recovery and Bitwise Partial-Sum Technique

Supposing that κ -bit secret key is involved to evaluate the integral characteristic with $2^{|I|}$ texts from ciphertexts, the trivial key recovery requires $2^{|I|+\kappa}$ time complexity. Ferguson et al. proposed the partial-sum technique to reduce the time complexity in [6]. In this technique, we first store the frequency of ciphertexts into a memory, ciphertexts are partially decrypted by guessing the part of involved keys, and reduce the size of the memory. Since the complexity is the product of the memory size and the partially guessed key size, the attacker can reduce the whole complexity by partial decryption and compressing the data size step by step.

Sasaki and Wang proposed the bitwise partial-sum technique, which improves the complexity of the partial-sum technique for ARX designs [15]. Suppose that n -bit variables X, Y and n -bit unknown key K . Also suppose that 2^{2n} pairs of (X, Y) are given to the attacker, and the goal of the attacker is to compute Z by exhaustively guessing K , where the following two operations are considered.

$$Z = (X \oplus K) \boxplus Y, \quad Z = (X \boxplus K) \oplus Y.$$

The complexity to compute Z is $2^{2n} \cdot 2^n = 2^{3n}$ operations. The bitwise partial-sum can reduce the complexity to $n \cdot 2^{2n+1}$ by computing Z bit by bit.

In practice, we need to evaluate the complexity for mod subtraction because of analyzing on decryption. At first, n -bit variable \bar{Y} and \bar{K} denote inverse elements corresponding to Y and K , respectively. Then, the following equations can be easily derived.

$$(X \oplus K) \boxminus Y = (X \oplus K) \boxplus \bar{Y}, \quad (X \boxminus K) \oplus Y = (X \boxplus \bar{K}) \oplus Y.$$

Hence, we can consider that the mod subtraction is equivalent to the mod addition are equivalent as far as guessing all values of \bar{Y} and \bar{K} , and use same procedure shown by [15]. The complexities to compute the above equations with the bytewise and bitwise partial-sum is given in Table 2.

Table 2. Summary of the complexity of the bytewise and bitwise partial-sum

Target equation	Bytewise partial-sum	Bitwise partial-sum
$Z = (X \oplus K) \boxminus Y$	2^{3n}	$n \cdot 2^{2n+1}$
$Z = (X \boxminus K) \oplus Y$	2^{3n}	$n \cdot 2^{2n+1}$

3 New Integral Characteristics on HIGHT

3.1 Previous 17-Round Integral Characteristics

Zhang et al. first showed 11-round integral characteristics with 2^8 chosen plain-texts in [25]. Moreover, they extended the characteristics to 17-round ones by using the higher-order integral as

$$\begin{aligned} (\mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{C}) &\xrightarrow{17R} (\mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{B}_0, \mathcal{U}, \mathcal{U}, \mathcal{U}), \\ (\mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{C}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}) &\xrightarrow{17R} (\mathcal{B}_0, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}), \end{aligned}$$

where \mathcal{B}_0 denotes that the lsb of the byte is balanced [25]. Moreover, \mathcal{A} denotes that every value appears the same number in the multiset, \mathcal{C} denotes that the value is fixed to a constant for all texts in the multiset and \mathcal{U} denotes that the multiset is indistinguishable from one of n -bit random values.

3.2 New Integral Characteristics Based on Division Property

We first propose some new 18-round integral characteristics, which are found by the propagation of the division property. As the unique structure of HIGHT, there are modular constant additions and modular additions of two values. Such additions are represented by the combination of half and full adders, and we generate the MILP model by simulating these adders by three propagation rules.

MILP Model for Modular Additions. We first consider the MILP model for half and full adders. In the half adder, the input is two bits a and b , and the output is the sum s and the carry c . Then, s and c are computed as

$$c = a \wedge b, \quad s = a \oplus b.$$

In the full adder, the input is three bits a , b , and x , and the output is the sum s and the carry c . Then, s and c are computed as

$$c = (a \wedge b) \oplus (x \wedge (a \oplus b)), \quad s = a \oplus b \oplus x.$$

`halfAdder` and `fullAdder` in Algorithm 1 generates the MILP model of the division property for `halfAdder` and `fullAdder`, respectively. Here, `halfAdder` consists of 6 $\mathcal{M}.vars$ and 5 $\mathcal{M}.cons$, and `fullAdder` consists of 13 $\mathcal{M}.vars$ and 10 $\mathcal{M}.cons$. Moreover, `modAdd` in Algorithm 1 shows the MILP model of the division property for modular addition of two n -bit values, where $(6 + 13 \times (n - 2) + 1)$ $\mathcal{M}.vars$ and $(5 + 10 \times (n - 2) + 1)$ $\mathcal{M}.cons$ are used. Constant round keys are modular added to the state in HIGHT, and `modAddConst` in Algorithm 1 shows the MILP model of the division property. In the constant addition, corresponding division property is always 0. Therefore, additions of the lsb and msb are simply represented, and it is enough to use `halfAdder` for additions of other bits. Therefore, $2 + 6 \times (n - 2) + 1$ $\mathcal{M}.vars$ and $1 + 5 \times (n - 2) + 1$ $\mathcal{M}.cons$ are used.

Algorithm 1. MILP model of division property for modular addition of two values.

<pre> 1: procedure halfAdder(\mathcal{M}, a, b) 2: $\mathcal{M}.var \leftarrow a_s, b_s, a_c, b_c, s, c$ * 3: $\mathcal{M}.con \leftarrow a = a_s + a_c$ 4: $\mathcal{M}.con \leftarrow b = b_s + b_c$ 5: $\mathcal{M}.con \leftarrow s = a_s + b_s$ 6: $\mathcal{M}.con \leftarrow c \geq a_c$ and $c \geq b_c$ 7: return (\mathcal{M}, s, c) </pre>	<pre> 1: procedure modAdd(\mathcal{M}, a, b, n) 2: (\mathcal{M}, s_1, c_1) = halfAdder(\mathcal{M}, a_1, b_1) 3: for $i = 2$ to $n - 1$ do 4: (\mathcal{M}, s_i, c_i) = fullAdder($\mathcal{M}, a_i, b_i, c_{i-1}$) 5: $\mathcal{M}.var \leftarrow s_n$ 6: $\mathcal{M}.con \leftarrow s_n = a_n + b_n + c_{n-1}$ 7: return (\mathcal{M}, s) </pre>
<pre> 1: procedure fullAdder(\mathcal{M}, a, b, x) 2: $\mathcal{M}.var \leftarrow a_s, b_s, a_u, b_u, a_v, b_v$ * 3: $\mathcal{M}.var \leftarrow x_s, x_w, s, c$ * 4: $\mathcal{M}.con \leftarrow a = a_s + a_u + a_v$ 5: $\mathcal{M}.con \leftarrow b = b_s + b_u + b_v$ 6: $\mathcal{M}.con \leftarrow x = x_s + x_w$ 7: $\mathcal{M}.con \leftarrow s = a_s + b_s + x_s$ 8: $\mathcal{M}.var \leftarrow u, v, w$ * 9: $\mathcal{M}.con \leftarrow u \geq a_u$ and $u \geq b_u$ 10: $\mathcal{M}.con \leftarrow v = a_v + b_v$ 11: $\mathcal{M}.con \leftarrow w \geq x_w$ and $w \geq v$ 12: $\mathcal{M}.con \leftarrow c = u + w$ 13: return (\mathcal{M}, s, c) </pre>	<pre> 1: procedure modAddConst(\mathcal{M}, a, n) 2: $\mathcal{M}.var \leftarrow s_1, c_1$ 3: $\mathcal{M}.con \leftarrow a_1 = s_1 + c_1$ 4: for $i = 2$ to $n - 1$ do 5: (\mathcal{M}, s_i, c_i) = halfAdder($\mathcal{M}, a_i, c_{i-1}$) 6: $\mathcal{M}.var \leftarrow s_n$ 7: $\mathcal{M}.con \leftarrow s_n = a_n + c_{n-1}$ 8: return (\mathcal{M}, s) </pre>

* means each variant are defined as binary.

New 18-Round Integral Characteristics. We implemented an MILP model of the division property for HIGHT. The algorithm to search for integral characteristics is described in Algorithm 2 of Appendix A. To find the longest integral characteristics, we choose one constant bit from 64 plaintext bits, i.e., 64 sets of 2^{63} chosen plaintexts are tried out. As a result, we found six 18-round integral characteristics as

$$\begin{aligned}
 \text{IC1} & \quad (\mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}_0) \xrightarrow{18R} (\mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{B}_0, \mathcal{U}, \mathcal{U}, \mathcal{U}), \\
 \text{IC2} & \quad (\mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}_0, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}) \xrightarrow{18R} (\mathcal{B}_0, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}), \\
 \text{IC3} & \quad (\mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}_0, \mathcal{A}, \mathcal{A}) \xrightarrow{18R} (\mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{B}_{1,0}, \mathcal{U}, \mathcal{U}, \mathcal{U}), \\
 \text{IC4} & \quad (\mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}_1, \mathcal{A}, \mathcal{A}) \xrightarrow{18R} (\mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{B}_{1,0}, \mathcal{U}, \mathcal{U}, \mathcal{U}), \\
 \text{IC5} & \quad (\mathcal{A}, \mathcal{A}_0, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}) \xrightarrow{18R} (\mathcal{B}_{1,0}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}), \\
 \text{IC6} & \quad (\mathcal{A}, \mathcal{A}_1, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}) \xrightarrow{18R} (\mathcal{B}_{1,0}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}),
 \end{aligned}$$

where \mathcal{A}_i denotes seven bits except for i -th bit are active and i -th bit is constant, and $\mathcal{B}_{1,0}$ denotes that the lsb and the second lsb are balanced¹.

¹ Sun et.al. also independently proposed 18-round integral characteristics in [16]. However, they presented only two characteristics as IC1 and IC2.

3.3 Extended 19-Round Integral Characteristics

We propose how to extend six 18-round integral characteristics to 19-round ones by appending one round to the plaintext side. Especially, we do not need to guess secret keys for extensions from IC1, IC2, IC3 and IC5, and it does not require the use of the full code book. Unfortunately, other two extensions requires guessing the part of secret keys, but we can easily append one round by using the full code book.

Extending IC1 and IC2. We consider the extension from IC1 and IC2, where the round function using F_0 is involved (see Fig. 2). Then, the lsb of the left half of the output is represented as

$$L[0] \oplus (F_0(R) \boxplus SK)[0] = L[0] \oplus F_0(R)[0] \oplus SK[0].$$

When the lsb of the left half of the plaintext takes a value $\mathcal{X} = F_0(R)[0]$, the lsb of the left half of the output is always constant. As a result, we can get two 19-round integral characteristics as

$$\begin{aligned} \text{IC1}' & \quad (\mathcal{A}^7 \parallel \mathcal{X}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}) \xrightarrow{19R} (\mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{B}_0, \mathcal{U}, \mathcal{U}, \mathcal{U}), \\ \text{IC2}' & \quad (\mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}^7 \parallel \mathcal{X}, \mathcal{A}, \mathcal{A}, \mathcal{A}) \xrightarrow{19R} (\mathcal{B}_0, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}), \end{aligned}$$

without guessing any bit of secret key, where \mathcal{A}^i denotes that i bits are active.

Extending IC3 and IC5. We consider the extension from IC3 and IC5, where the round function using F_1 is involved (see Fig. 3). Then, the lsb of the left half of the output is represented as

$$L[0] \boxplus (F_1(R) \oplus SK)[0] = L[0] \oplus F_1(R)[0] \oplus SK[0].$$

When the lsb of the left half of the plaintext takes a value $\mathcal{X} = F_1(R)[0]$, the lsb of the left half of the output is always constant. As a result, we can get two 19-round integral characteristics as

$$\begin{aligned} \text{IC3}' & \quad (\mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}^7 \parallel \mathcal{X}, \mathcal{A}) \xrightarrow{19R} (\mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{B}_{1,0}, \mathcal{U}, \mathcal{U}, \mathcal{U}), \\ \text{IC5}' & \quad (\mathcal{A}, \mathcal{A}, \mathcal{A}^7 \parallel \mathcal{X}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}) \xrightarrow{19R} (\mathcal{B}_{1,0}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}), \end{aligned}$$

without guessing any bit of secret key.

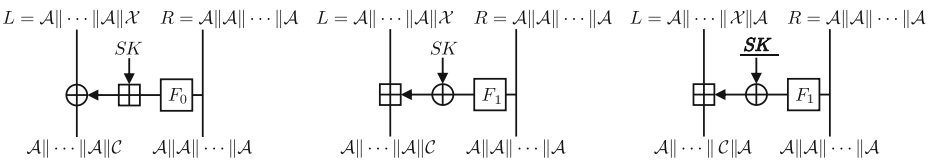


Fig. 2. IC1, 2

Fig. 3. IC3, 5

Fig. 4. IC4, 6

Extending IC4 and IC6. We consider the extension from IC4 and IC6 (see Fig. 4). The second lsb is constant in these integral characteristics instead of the lsb. Then, the second lsb of the left half of the output is represented as

$$L[1] \oplus F_1(R)[1] \oplus SK[1] \oplus (L[0] \times (F_1(R)[0] \oplus SK[0])).$$

When the second lsb of the left half of the plaintext takes a value $\mathcal{X} = F_1(R)[1] \oplus (L[0] \times (F_1(R)[0] \oplus SK[0]))$, the output second lsb is constant. Then,

$$\begin{aligned} \text{IC4}' & \quad (\mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}^6 \|\mathcal{X}\| \mathcal{A}^1, \mathcal{A}) \xrightarrow{19R} (\mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{B}_{1,0}, \mathcal{U}, \mathcal{U}, \mathcal{U}), \\ \text{IC6}' & \quad (\mathcal{A}, \mathcal{A}, \mathcal{A}^6 \|\mathcal{X}\| \mathcal{A}^1, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}) \xrightarrow{19R} (\mathcal{B}_{1,0}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}, \mathcal{U}). \end{aligned}$$

Unfortunately, these extension requires guessing $SK[0]$ and the full code book.

These integral characteristics could not be detected by MILP-aided tool. In our procedure that how to extend integral characteristics, we have to compose the set of chosen plaintexts which include some non-linear part, represented as \mathcal{X} . On the other hand, the division property provide the completely linear and generalized set of chosen plaintexts. As a result, the MILP-aided tool using the division property can find 18-round integral characteristics but cannot find 19-round ones.

4 28-Round Attack on HIGHT Without Full Code Book

4.1 Whitening Key Addition to Integral Characteristics

In Sect. 3.3, we showed new 19-round integral characteristics, but we cannot use each characteristic directly because whitening key is added to plaintexts at first. In this section, we propose how to add the whitening to six 19-round integral characteristics.

First, we add the whitening to IC1' or IC2', where the XOR is used as the addition. Then, the whitening key is linearly involved to the lsb of the left half of the output. Therefore, even if there is the whitening, we can use IC1' without guessing the key.

Next, we add the whitening to IC3' or IC5', where the modular addition is used as the addition. Then, the whitening key is nonlinearly involved to the lsb of the left half of the output. Unfortunately, this requires guessing the whitening key, and the use of the full code book is required to add the whitening. Similarly, the full code book is required to add the whitening to IC4' or IC6'.

As a result, we can add the whitening to IC1' and IC2' without using the full code book. Other four integral characteristics can be added the whitening when the full code book is used. Hereafter, we only use IC1' and IC2' to avoid the use of the full code book in this section.

4.2 Meet-in-the-Middle Technique

Let us consider the integral attack using IC1'. Then, $X_3^{19}[0]$ is balanced and can be written as a linear combination of two variables $X_4^{20}[0]$ and $Z_3^{19}[0]$, where $X_i^r[j]$ denotes the j -th bit of the X_i^r . In the meet-in-the-middle technique [14], each sum is independently computed from ciphertexts, and secret keys satisfying $\bigoplus Z_3^{19}[0] = \bigoplus X_4^{20}[0]$ are recovered by using the computation like the meet-in-the-middle attack. Furthermore, for HIGHT, this concept is extended by exploiting more linearity inside the round function in [15]. Since the complexity for computing $\bigoplus Z_3^{19}[0]$ is much bigger than the one for $\bigoplus X_4^{20}[0]$, we reduce the number of subkeys involved $\bigoplus Z_3^{19}[0]$. We focus on that $Z_3^{19}[0]$ is computed by $SK_{77}[0] \boxplus Y_3^{19}[0]$, and this is represented as $Z_3^{19}[0] = SK_{77}[0] \oplus Y_3^{19}[0]$ because the lsb of the modular addition is a XOR. Therefore, $SK_{77}[0]$ can be removed, namely $\bigoplus X_4^{20}[0] = \bigoplus Y_3^{19}[0]$. Furthermore, by utilizing the linearity of F_0 , i.e., $Y_3^{19}[0] = X_3^{20}[7] \oplus X_3^{20}[6] \oplus X_3^{20}[1]$, we can move more subkey bits, and finally get the following equation.

$$\bigoplus (X_4^{20}[0] \oplus X_4^{21}[7] \oplus X_4^{21}[6] \oplus X_4^{21}[1]) = \bigoplus (Z_3^{20}[7] \oplus Z_3^{20}[6] \oplus Z_3^{20}[1]). \quad (1)$$

Unfortunately, 13-byte keys are involved to the right half of Eq. (1), and we cannot append 9 rounds like [15]. Moreover, 14-byte keys are involved when IC2' is used. Alternatively, we attack 28-round HIGHT from the 2-nd round to 29-th round with whitening keys. Then, 12-byte keys and 13-byte keys are involved when IC1' and IC2' are used, respectively. We detail the analyses of the involved keys about each Z_i^r in Table 5 of Appendix B.

We prepare the 28-round HIGHT from the 2-nd to 29-th round and apply IC1' to 19-round between 2-nd and 20-th round. Then, $X_3^{20}[0]$ is balanced, and we finally get the following equation.

$$\bigoplus (X_4^{21}[0] \oplus X_4^{22}[7] \oplus X_4^{22}[6] \oplus X_4^{22}[1]) = \bigoplus (Z_3^{21}[7] \oplus Z_3^{21}[6] \oplus Z_3^{21}[1]). \quad (2)$$

4.3 Attack Procedure

We use the relationship between the whitening key, subkey and master key in Table 3.

Since the computation for the right-hand side of Eq. (2) requires much more complexity than the left-hand side, we only explain the procedure to obtain the right-hand side of Eq. (2) and evaluate the time complexity. The partial decryption for obtaining $\bigoplus (Z_3^{21}[7] \oplus Z_3^{21}[6] \oplus Z_3^{21}[1])$ is shown in Fig. 5. We first describe our procedure with the bitwise partial-sum technique as following steps:

1. The analysis starts from at most 2^{64} ciphertexts of (C_0, \dots, C_7) .
2. K_1 and K_2 are guessed and the data is compressed into 2^{56} texts.
3. K_2 has been already guessed, so K_3 is guessed and the data is converted into 2^{56} texts.

Table 3. Relationship between Whitening key, Subkey and Master key from 1-st to 29-th round

Round	RK_7	RK_5	RK_3	RK_1	Round	RK_7	RK_5	RK_3	RK_1
W	K_{15}	K_{14}	K_{13}	K_{12}	16	K_{12}	K_{11}	K_{10}	K_9
1	K_3	K_1	K_1	K_0	17	K_7	K_6	K_5	K_4
2	K_7	K_6	K_5	K_4	18	K_3	K_2	K_1	K_0
3	K_{11}	K_{10}	K_9	K_8	19	K_{15}	K_{14}	K_{13}	K_{12}
4	K_{15}	K_{14}	K_{13}	K_{12}	20	K_{11}	K_{10}	K_9	K_8
5	K_2	K_1	K_0	K_7	21	K_6	K_5	K_4	K_3
6	K_6	K_5	K_4	K_3	22	K_2	K_1	K_0	K_7
7	K_{10}	K_9	K_8	K_{15}	23	K_{14}	K_{13}	K_{12}	K_{11}
8	K_{14}	K_{13}	K_{12}	K_{11}	24	K_{10}	K_9	K_8	K_{15}
9	K_1	K_0	K_7	K_6	25	K_5	K_4	K_3	K_2
10	K_5	K_4	K_3	K_2	26	K_1	K_0	K_7	K_6
11	K_9	K_8	K_{15}	K_{14}	27	K_{13}	K_{12}	K_{11}	K_{10}
12	K_{13}	K_{12}	K_{11}	K_{10}	28	K_9	K_8	K_{15}	K_{14}
13	K_0	K_7	K_6	K_5	29	K_4	K_3	K_2	K_1
14	K_4	K_3	K_2	K_1	W	K_3	K_2	K_1	K_0
15	K_8	K_{15}	K_{14}	K_{13}					

When Round is W, RK_i denotes the round key corresponding WK . Otherwise, RK_i denotes the round key corresponding SK

4. K_{15} is guessed and the data is compressed into 2^{48} texts.
5. K_3 has been already guessed, so K_4 is guessed and the data is converted into 2^{48} texts.
6. K_8 is guessed and the data is converted into 2^{48} texts.
7. K_{11} is guessed and the data is compressed into 2^{40} texts.
8. K_1 has been already guessed, so K_0 is guessed and the data is converted into 2^{40} texts.
9. K_9 is guessed and the data is converted into 2^{40} texts.
10. K_{12} is guessed and the data is converted into 2^{40} texts.
11. K_7 is guessed and the data is compressed into 2^{32} texts.
12. K_3 has been already guessed, so the data is converted into 2^{32} texts.
13. K_0 has been already guessed, so the data is converted into 2^{32} texts.
14. K_8 has been already guessed, so the data is converted into 2^{32} texts.
15. K_{13} is guessed and the data is compressed into 2^{24} texts.
16. K_4 has been already guessed, so the data is compressed into 2^{16} texts.
17. K_{12} has been already guessed, so the data is compressed into 2^8 texts.
18. K_0 has been already guessed, so the data is compressed into 1 text of Z_3^{21} .
Then, we can calculate the value of $\bigoplus(Z_3^{21}[7] \oplus Z_3^{21}[6] \oplus Z_3^{21}[1])$.

This procedure and its time complexity evaluation is summarized in Table 4. Step 11 and 15 requires the dominant time complexity, where 2^{128} round

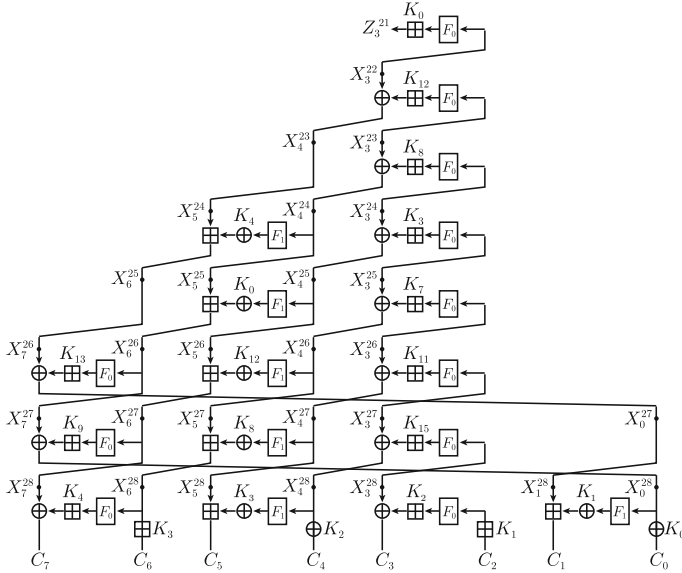


Fig. 5. Partial decryption for $\bigoplus(Z_3^{21}[7] \oplus Z_3^{21}[6] \oplus Z_3^{21}[1])$ on 28-round attack

function computations is used for the bitwise partial sum. We apply the bitwise partial sum to Step 11 and 15 to reduce the complexities. Step 11 starts from 2^{40} texts of $(X_0^{27}, X_3^{26}, X_4^{26}, X_5^{26}, X_7^{27})$, and the goal is obtaining 2^{32} texts of $(X_0^{27}, X_3^{25}, X_5^{26}, X_7^{27})$ with guessing K_7 . We then apply the bitwise partial sum to guess K_7 . Referring Table 2, its time complexity is reduced to $n \cdot (2^{80+39+1}) = 2^{123}$ round functions, where $n = 8$. In Step 15, we can also reduce the time complexity about 2^{128} to $n \cdot (2^{88+31+1}) = 2^{123}$ round functions, similarly. Finally, the time complexity in the key recovery is estimated by Step 11 and 15 because the complexities of other steps are negligible compared with 2^{123} . Hence, the time complexity of the key recovery is about 2^{124} round functions(RF).

Since only one integral characteristic with one balance bit is used, this key recovery only reduces the 1-bit of information on the master key. Therefore, we finally exhaustively searches 2^{127} master keys. As a result, the whole complexity of our attack is 2^{124} RF + 2^{127} Enc $\approx 2^{127}$ Enc.

5 29-Round Attack on HIGHT with Full Code Book

When the use of the full code book is acceptable, we can attack 29-round HIGHT, where one round is added to the plaintext side from the 28-round attack. Therefore, while 28-round HIGHT from the 2-nd to 29-th round is attacked in Sect. 4, the natural 29-round HIGHT is attacked.

We briefly describe the overview of our 29-round attack. We first prepare the set of chosen texts for the input of the 2-nd round function such that it brings 19-round integral characteristics, i.e., $X_3^{20}[0]$ is balanced, and it is the same as the 28-round attack. Next, we guess the round key in the 1-st round and whitening keys, and get the set of corresponding plaintexts. Since the set of plaintexts depends on the guessed keys, 29-round attack requires the use of the full code book. Moreover, the position of the guessed keys depends on each characteristic, and Appendix C shows it in detail. Finally, we execute the key recovery that is the same as that for the 28-round attack.

Table 4. Summary of the computation for $\bigoplus(Z_3^{21}[7] \oplus Z_3^{21}[6] \oplus Z_3^{21}[1])$.

Step	Guessed keys	Data size	Texts need to be analyzed	Complexity (byte-wise)	Complexity (bit-wise)
1	—	2^{64}	$(C_0, C_1, C_2, C_3, C_4, C_5, C_6, C_7)$	2^{64} MA	—
2	K_1, K_2	2^{56}	$(C_0, C_1, X_3^{28}, C_4, C_5, C_6, C_7)$	$2^{16} \cdot 2^{64} = 2^{80}$ RF	—
3	$K_3(, K_2)$	2^{56}	$(C_0, C_1, X_3^{28}, X_4^{28}, X_5^{28}, C_6, C_7)$	$2^{16} \cdot 2^8 \cdot 2^{56} = 2^{80}$ RF	—
4	K_{15}	2^{48}	$(C_0, C_1, X_3^{27}, X_5^{28}, C_6, C_7)$	$2^{24} \cdot 2^8 \cdot 2^{56} = 2^{88}$ RF	—
5	$K_4(, K_3)$	2^{48}	$(C_0, C_1, X_3^{27}, X_5^{28}, X_6^{28}, X_7^{28})$	$2^{32} \cdot 2^8 \cdot 2^{48} = 2^{88}$ RF	—
6	K_8	2^{48}	$(C_0, C_1, X_3^{27}, X_4^{27}, X_5^{27}, X_7^{28})$	$2^{40} \cdot 2^8 \cdot 2^{48} = 2^{96}$ RF	—
7	K_{11}	2^{40}	$(C_0, C_1, X_3^{26}, X_5^{27}, X_7^{28})$	$2^{48} \cdot 2^8 \cdot 2^{48} = 2^{104}$ RF	—
8	$K_0(, K_1)$	2^{40}	$(X_0^{28}, X_0^{27}, X_3^{26}, X_5^{27}, X_7^{28})$	$2^{56} \cdot 2^8 \cdot 2^{40} = 2^{104}$ RF	—
9	K_9	2^{40}	$(X_0^{27}, X_3^{26}, X_5^{27}, X_6^{27}, X_7^{27})$	$2^{64} \cdot 2^8 \cdot 2^{40} = 2^{112}$ RF	—
10	K_{12}	2^{40}	$(X_0^{27}, X_3^{26}, X_4^{26}, X_5^{26}, X_7^{27})$	$2^{72} \cdot 2^8 \cdot 2^{40} = 2^{120}$ RF	—
11	K_7	2^{32}	$(X_0^{27}, X_3^{25}, X_5^{26}, X_7^{27})$	$2^{80} \cdot 2^8 \cdot 2^{40} = 2^{128}$ RF	$n \cdot (2^{80+39+1}) = 2^{123}$ RF
12	(K_3)	2^{32}	$(X_0^{27}, X_3^{24}, X_5^{26}, X_7^{27})$	$2^{88} \cdot 2^0 \cdot 2^{32} = 2^{120}$ RF	—
13	(K_0)	2^{32}	$(X_0^{27}, X_3^{24}, X_5^{25}, X_7^{27})$	$2^{88} \cdot 2^0 \cdot 2^{32} = 2^{120}$ RF	—
14	(K_8)	2^{32}	$(X_0^{27}, X_3^{23}, X_5^{25}, X_7^{27})$	$2^{88} \cdot 2^0 \cdot 2^{32} = 2^{120}$ RF	—
15	K_{13}	2^{24}	$(X_3^{23}, X_5^{25}, X_6^{25})$	$2^{88} \cdot 2^8 \cdot 2^{32} = 2^{128}$ RF	$n \cdot (2^{88+31+1}) = 2^{123}$ RF
16	(K_4)	2^{16}	(X_3^{23}, X_4^{23})	$2^{96} \cdot 2^0 \cdot 2^{24} = 2^{120}$ RF	—
17	(K_{12})	2^8	(X_3^{22})	$2^{96} \cdot 2^0 \cdot 2^{16} = 2^{112}$ RF	—
18	(K_0)	1	$\bigoplus(Z_3^{21}[7] \oplus Z_3^{21}[6] \oplus Z_3^{21}[1])$	$2^{96} \cdot 2^0 \cdot 2^8 = 2^{104}$ RF	—

MA and RF stand for memory access and round function, respectively

We first try to execute 29-round attack using the 19-round integral characteristic IC1'. To prepare the set of plaintexts, we have to guess the value of K_{14} and K_2 . Unfortunately, K_{14} is not involved to the key recovery shown in Table 4. Therefore, the complexity of each step always requires 2^8 times, and the time complexity in Step 11 and 15 is over 2^{128} even if the bit-wise partial sum is applied. As a result, we cannot use IC1'. We next try to execute 29-round attack using the 19-round integral characteristic IC3'. Note that the position of balanced byte is the same as that by IC1', i.e., $X_3^{20}[0]$ is balanced. Therefore, we can use the same procedure for the key recovery. To prepare the set of plaintexts, we have to guess the value of K_{15} and K_3 , which are already guessed in Step 4 and 3 in the key recovery, respectively. Hence, we add two bytes to the guessed keys in Step 1–2 and one byte to them in Step 3, and the complexity does not change after Step 4. Therefore, even if IC3' is used, the complexity of the key recovery is still 2^{124} RF because the dominant part is in Step 11 and 15. Moreover, when IC4' is used, we have to guess the value of not only K_{15} and K_3 but also the lsb of K_4 and K_{12} , but all additional guessing keys have been already guessed in the key recovery. Therefore, similarly to the key recovery using IC3', the dominant complexity is still 2^{124} RF. Since we can execute the key recovery using both IC3' and IC4' in the same time, 2 bits of information of the master key is recovered. As a result, the whole complexity of our attack is 2^{124} RF + 2^{126} Enc $\approx 2^{126}$ Enc.

6 Conclusions

In this paper, we first proposed 19-round integral characteristics by using the propagation of the division property. These characteristics are improved by two rounds compared with previous ones. Then, we showed the attack against 28-round HIGHT by appending 9-round key recovery. We attacked 28-round HIGHT with 2^{63} data size and 2^{127} time complexity. Moreover, we showed another attack on 29-HIGHT with 2^{64} data size and 2^{126} time complexity. These attacks are the best known attack against HIGHT under the single-key setting.

A Detailed MILP Model for HIGHT

In this appendix, the detailed algorithm to search for integral characteristics on HIGHT is described.

As a result of running the Algorithm 2 with our machine (CPU: i5-6500 @ 3.20 GHz, 3.20 GHz/RAM: 8.00 GB/64-bit operating system, x64 base processor), it took about 50 min.

Algorithm 2. MILP model of division property for R -round HIGHT.

```

1: procedure funcF( $\mathcal{M}, \mathbf{x}, r_1, r_2, r_3$ )
2:    $\mathcal{M}.var \leftarrow y[j], x_1[j], x_2[j], x_3[j]$  for  $j \in \{0, 1, \dots, 7\}^*$ 
3:   for  $j = 0$  to 7 do
4:      $\mathcal{M}.con \leftarrow x[j] = x_1[j] + x_2[j] + x_3[j]$ 
5:      $\mathcal{M}.con \leftarrow y[j] = x_1[(j - r_1) \bmod 8] + x_2[(j - r_2) \bmod 8] + x_3[(j - r_3) \bmod 8]$ 
6:   return ( $\mathcal{M}, \mathbf{y}$ )
1: procedure roundFunc1( $\mathcal{M}, \mathbf{l}, \mathbf{r}$ )
2:    $\mathcal{M}.var \leftarrow x[j], s[j]$  for  $j \in \{0, 1, \dots, 7\}^*$ 
3:   for  $j = 0$  to 7 do
4:      $\mathcal{M}.con \leftarrow r[j] = x[j] + s[j]$ 
5:   ( $\mathcal{M}, \mathbf{y}$ ) = funcF( $\mathcal{M}, \mathbf{x}, 3, 4, 6$ )
6:    $\mathbf{t} = \text{modAdd}(\mathcal{M}, \mathbf{l}, \mathbf{y}, 8)$ 
7:   return ( $\mathcal{M}, \mathbf{s}, \mathbf{t}$ )
1: procedure roundFunc0( $\mathcal{M}, \mathbf{l}, \mathbf{r}$ )
2:    $\mathcal{M}.var \leftarrow x[j], s[j]$  for  $j \in \{0, 1, \dots, 7\}^*$ 
3:   for  $j = 0$  to 7 do
4:      $\mathcal{M}.con \leftarrow r[j] = x[j] + s[j]$ 
5:   ( $\mathcal{M}, \mathbf{y}$ ) = funcF( $\mathcal{M}, \mathbf{x}, 1, 2, 7$ )
6:    $\mathbf{z} = \text{modAddConst}(\mathcal{M}, \mathbf{y}, 8)$ 
7:    $\mathcal{M}.var \leftarrow t[j]$  for  $j \in \{0, 1, \dots, 7\}^*$ 
8:   for  $j = 0$  to 7 do
9:      $\mathcal{M}.con \leftarrow t[j] = l[j] + z[j]$ 
10:  return ( $\mathcal{M}, \mathbf{s}, \mathbf{t}$ )
1: procedure HIGHT(round  $R$ , index  $I$ , target bit  $t$ )
2:   create MILP model  $\mathcal{M}$ 
3:    $\mathcal{M}.var \leftarrow x_i^0[j]$  for  $(i, j) \in \{(0, 0), (0, 1), \dots, (7, 7)\}^*$ 
4:   for  $(i, j) = (0, 0)$  to  $(7, 7)$  do
5:     if  $8 \times i + j$  is included in  $I$  then  $\mathcal{M}.con \leftarrow x_i^0[j] = 1$ 
6:     else  $\mathcal{M}.con \leftarrow x_i^0[j] = 0$ 
7:   for  $r = 0$  to  $R - 1$  do
8:     ( $\mathcal{M}, \mathbf{x}_1^{r+1}, \mathbf{x}_2^{r+1}$ ) = roundFunc1( $\mathcal{M}, \mathbf{x}_1^r, \mathbf{x}_0^r$ )
9:     ( $\mathcal{M}, \mathbf{x}_3^{r+1}, \mathbf{x}_4^{r+1}$ ) = roundFunc0( $\mathcal{M}, \mathbf{x}_3^r, \mathbf{x}_2^r$ )
10:    ( $\mathcal{M}, \mathbf{x}_5^{r+1}, \mathbf{x}_6^{r+1}$ ) = roundFunc1( $\mathcal{M}, \mathbf{x}_5^r, \mathbf{x}_4^r$ )
11:    ( $\mathcal{M}, \mathbf{x}_7^{r+1}, \mathbf{x}_0^{r+1}$ ) = roundFunc0( $\mathcal{M}, \mathbf{x}_7^r, \mathbf{x}_6^r$ )
12:   for  $(i, j) = (0, 0)$  to  $(7, 7)$  do
13:     if  $8 \times i + j = t$  then  $\mathcal{M}.con \leftarrow x_i^r[j] = 1$ 
14:     else  $\mathcal{M}.con \leftarrow x_i^r[j] = 0$ 
15:   solve this MILP model  $\mathcal{M}$ 
16:   if  $\mathcal{M}$  is infeasible then return the target bit is balanced.
17:   else return the target bit is unknown.

```

* means each variant are defined as binary.

B Involved Key Size in Key Recovery

In this appendix, the detailed analyses of the involved key size in the calculation of Z_i^r is described.

Table 5. The number of involved key bytes in calculation of Z_i^r

Rounds of Key Recovery	Target	Type of using IC	Involved Key Size	Ignored Keys
20 – 28 R	Z_3^{20}	IC1', IC3', IC4'	13 bytes	K_5, K_6, K_{10}
20 – 28 R	Z_7^{20}	IC2', IC5', IC6'	14 bytes	K_4, K_{12}
21 – 29 R	Z_3^{21}	IC1', IC3', IC4'	12 bytes	K_5, K_6, K_{10}, K_{14}
21 – 29 R	Z_7^{21}	IC2', IC5', IC6'	13 bytes	K_7, K_8, K_{12}
22 – 30 R	Z_3^{22}	IC1', IC3', IC4'	13 bytes	K_{10}, K_{13}, K_{14}
22 – 30 R	Z_7^{22}	IC2', IC5', IC6'	13 bytes	K_8, K_{11}, K_{12}
23 – 31 R	Z_3^{23}	IC1', IC3', IC4'	13 bytes	K_5, K_{13}, K_{14}
23 – 31 R	Z_7^{23}	IC2', IC5', IC6'	13 bytes	K_7, K_8, K_{15}
24 – 32 R	Z_3^{24}	IC1', IC3', IC4'	13 bytes	K_4, K_5, K_9
24 – 32 R	Z_7^{24}	IC2', IC5', IC6'	14 bytes	K_7, K_{11}

C Detailed Addition of Whitening Layer

In this appendix, we described detailed procedure how to add the whitening layer to three 19-round integral characteristics. The first 2-round procedure on HIGHT is shown in Fig. 6. Please refer to Table 3 in order to know the relationship between the round keys and the master keys.

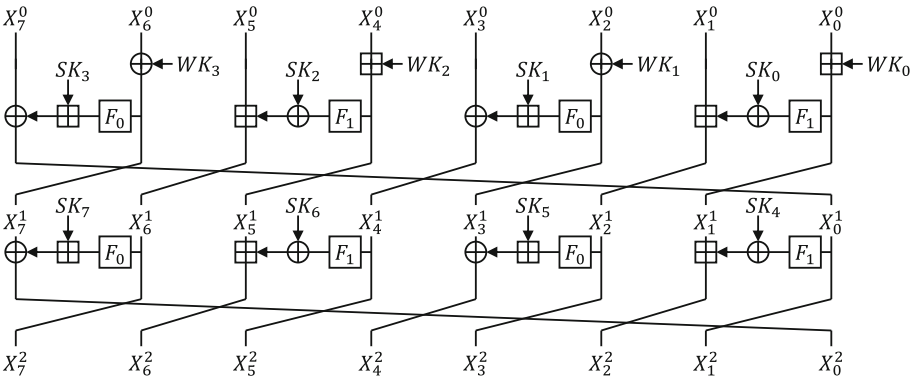


Fig. 6. 1-st and 2-nd round procedure of HIGHT

We consider the extension from IC1' and the lsb of X_0^2 is represented as

$$\begin{aligned} X_0^2[0] &= X_7^1[0] \oplus SK_7[0] \oplus F_0(X_6^1)[0] \\ &= X_6^0[0] \oplus WK_3[0] \oplus SK_7[0] \oplus F_0(F_1(X_4^0 \boxplus WK_2) \oplus SK_2 \boxplus X_5^0)[0]. \end{aligned}$$

We can ignore the value of $WK_3[0]$ and $SK_7[0]$ because $X_0^2[0]$ is added linearly by them using XOR. But we cannot ignore that this extension requires guessing the value of K_{14} and K_2 as WK_2 and SK_2 , respectively. Next, we consider the extension from IC3'. In case of considering the lsb, we can regard the modular addition as the XOR. So the lsb of X_2^2 is represented as

$$\begin{aligned} X_2^2[0] &= X_1^1[0] \oplus SK_4[0] \oplus F_1(X_0^1)[0] \\ &= X_0^0[0] \oplus WK_0[0] \oplus SK_4[0] \oplus F_1(F_0(X_6^0 \oplus WK_3) \boxplus SK_3 \oplus X_7^0)[0]. \end{aligned}$$

This extension requires guessing the value of K_{15} and K_3 as WK_3 and SK_3 , respectively. Finally, we consider the extension from IC4' and the second lsb is represented as

$$X_2^2[1] = X_1^1[1] \oplus F_1(X_0^1)[1] \oplus SK_4[1] \oplus (X_1^1[0] \times (F_1(X_0^1)[0] \oplus SK_4[0])),$$

where each X^1 are represented as follows:

$$\begin{aligned} X_1^1[1] &= X_0^0[1] \oplus WK_0[1] \oplus (X_0^0[0] \times WK_0[0]), \\ X_1^1[0] &= X_0^0[0] \oplus WK_0[0], \\ X_0^1 &= F_0(X_6^0 \oplus WK_3) \boxplus SK_3 \oplus X_7^0. \end{aligned}$$

This extension requires guessing the value of $K_4[0], K_{12}[0], K_{15}$ and K_3 as $SK_4[0], WK_0[0], WK_3$ and SK_3 , respectively.

References

1. Agency, K.I.S.: Hight algorithm specification (2009)
2. Biryukov, A., Perrin, L.: Lightweight cryptography lounge (2015). http://cryptolux.org/index.php/Lightweight_Cryptography
3. Chen, J., Wang, M., Preneel, B.: Impossible differential cryptanalysis of the lightweight block ciphers TEA, XTEA and HIGHT. In: Mitrokovska, A., Vaudenay, S. (eds.) AFRICACRYPT 2012. LNCS, vol. 7374, pp. 117–137. Springer, Heidelberg (2012). doi:10.1007/978-3-642-31410-0_8
4. Cui, T., Jia, K., Fu, K., Chen, S., Wang, M.: New automatic search tool for impossible differentials and zero-correlation linear approximations (2016). <http://eprint.iacr.org/2016/689>
5. Daemen, J., Knudsen, L., Rijmen, V.: The block cipher Square. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 149–165. Springer, Heidelberg (1997). doi:10.1007/BFb0052343
6. Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D., Whiting, D.: Improved cryptanalysis of Rijndael. In: Goos, G., Hartmanis, J., Leeuwen, J., Schneier, B. (eds.) FSE 2000. LNCS, vol. 1978, pp. 213–230. Springer, Heidelberg (2001). doi:10.1007/3-540-44706-7_15

7. Hong, D., et al.: HIGHT: a new block cipher suitable for low-resource device. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 46–59. Springer, Heidelberg (2006). doi:[10.1007/11894063_4](https://doi.org/10.1007/11894063_4)
8. ISO/IEC: JTC1: ISO/IEC 18033–3: Information technology - security techniques - encryption algorithms - part 3: Block ciphers (2010)
9. Knudsen, L., Wagner, D.: Integral cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 112–127. Springer, Heidelberg (2002). doi:[10.1007/3-540-45661-9_9](https://doi.org/10.1007/3-540-45661-9_9)
10. Koo, B., Hong, D., Kwon, D.: Related-key attack on the Full HIGHT. In: Rhee, K.-H., Nyang, D.H. (eds.) ICISC 2010. LNCS, vol. 6829, pp. 49–67. Springer, Heidelberg (2011). doi:[10.1007/978-3-642-24209-0_4](https://doi.org/10.1007/978-3-642-24209-0_4)
11. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: Wu, C.-K., Yung, M., Lin, D. (eds.) Inscrypt 2011. LNCS, vol. 7537, pp. 57–76. Springer, Heidelberg (2012). doi:[10.1007/978-3-642-34704-7_5](https://doi.org/10.1007/978-3-642-34704-7_5)
12. Özen, O., Varıcı, K., Tezcan, C., Kocair, Ç.: Lightweight block ciphers revisited: cryptanalysis of reduced round PRESENT and HIGHT. In: Boyd, C., González Nieto, J. (eds.) ACISP 2009. LNCS, vol. 5594, pp. 90–107. Springer, Heidelberg (2009). doi:[10.1007/978-3-642-02620-1_7](https://doi.org/10.1007/978-3-642-02620-1_7)
13. Sasaki, Y., Todo, Y.: New impossible differential search tool from design and cryptanalysis aspects (2016). <http://eprint.iacr.org/2016/1181>. This paper is accepted in Eurocrypt 2017
14. 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). doi:[10.1007/978-3-642-35999-6_16](https://doi.org/10.1007/978-3-642-35999-6_16)
15. Sasaki, Y., Wang, L.: Bitwise partial-sum on HIGHT: A New tool for integral analysis against ARX designs. In: Lee, H.-S., Han, D.-G. (eds.) ICISC 2013. LNCS, vol. 8565, pp. 189–202. Springer, Cham (2014). doi:[10.1007/978-3-319-12160-4_12](https://doi.org/10.1007/978-3-319-12160-4_12)
16. Sun, L., Wang, W., Liu, R., Wang, M.: Milp-aided bit-based division property for ARX-based block cipher. IACR Cryptology ePrint Archive 2016, 1101 (2016). <http://eprint.iacr.org/2016/1101>
17. Sun, S., Hu, L., Wang, M., Wang, P., Qiao, K., Ma, X., Shi, D., Song, L.: Towards finding the best characteristics of some bit-oriented block ciphers and automatic enumeration of (related-key) differential and linear characteristics with predefined properties (2014). <http://eprint.iacr.org/2014/747>
18. Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (Related-key) differential characteristic search: application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 158–178. Springer, Heidelberg (2014). doi:[10.1007/978-3-662-45611-8_9](https://doi.org/10.1007/978-3-662-45611-8_9)
19. Suzaki, T., Minematsu, K., Morioka, S., Kobayashi, E.: *TWINE*: a lightweight block cipher for multiple platforms. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 339–354. Springer, Heidelberg (2013). doi:[10.1007/978-3-642-35999-6_22](https://doi.org/10.1007/978-3-642-35999-6_22)
20. Todo, Y.: Integral cryptanalysis on full MISTY1. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 413–432. Springer, Heidelberg (2015). doi:[10.1007/978-3-662-47989-6_20](https://doi.org/10.1007/978-3-662-47989-6_20)
21. 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). doi:[10.1007/978-3-662-46800-5_12](https://doi.org/10.1007/978-3-662-46800-5_12)

22. Todo, Y., Morii, M.: Bit-based division property and application to SIMON family. In: Peyrin, T. (ed.) FSE 2016. LNCS, vol. 9783, pp. 357–377. Springer, Heidelberg (2016). doi:[10.1007/978-3-662-52993-5_18](https://doi.org/10.1007/978-3-662-52993-5_18)
23. Wu, W., Zhang, L.: LBlock: a lightweight block cipher. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 327–344. Springer, Heidelberg (2011). doi:[10.1007/978-3-642-21554-4_19](https://doi.org/10.1007/978-3-642-21554-4_19)
24. Xiang, Z., Zhang, W., Bao, Z., Lin, D.: Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10031, pp. 648–678. Springer, Heidelberg (2016). doi:[10.1007/978-3-662-53887-6_24](https://doi.org/10.1007/978-3-662-53887-6_24)
25. Zhang, P., Sun, B., Li, C.: Saturation attack on the block cipher HIGHT. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 76–86. Springer, Heidelberg (2009). doi:[10.1007/978-3-642-10433-6_6](https://doi.org/10.1007/978-3-642-10433-6_6)