# Meet-in-the-Middle Technique for Integral Attacks against Feistel Ciphers

Yu Sasaki[1] and Lei Wang[2]

[1] NTT Secure Platform Laboratories, NTT Corporation
3-9-11 Midori-cho, Musashino-shi, Tokyo 180-8585 Japan
`sasaki.yu@lab.ntt.co.jp`
[2] The University of Electro-Communications
1-5-1 Choufugaoka, Choufu-shi, Tokyo, 182-8585 Japan

**Abstract.** In this paper, an improvement for integral attacks against Feistel ciphers is discussed. The new technique can reduce the complexity of the key recovery phase. This possibly leads to an extension of the number of attacked rounds. In the integral attack, an attacker guesses a part of round keys and performs the partial decryption. The correctness of the guess is judged by examining whether the XOR sum of the results becomes 0 or not. In this paper, it is shown that the computation of the XOR sum of the partial decryptions can be divided into two independent parts if the analysis target adopts the Feistel network or its variant. Then, correct key candidates are efficiently obtained with the meet-in-the-middle approach. The effect of our technique is demonstrated for several Feistel ciphers. Improvements on integral attacks against LBlock, HIGHT, and CLEFIA are presented. Particularly, the number of attacked rounds with integral analysis is extended for LBlock.

**Keywords:** Integral attack, Meet-in-the-middle, Feistel, Partial-sum, LBlock, HIGHT, CLEFIA.

## 1 Introduction

The integral attack is a cryptanalytic technique for symmetric-key primitives, which was firstly proposed by Daemen *et al.* to evaluate the security of SQUARE cipher [1], and was later unified as integral attack by Knudsen and Wagner [2]. The crucial part is a construction of an *integral distinguisher*: an attacker prepares a set of plaintexts which contains all possible values for some bytes and has a constant value for the other bytes. All plaintexts in the set are passed to the encryption oracle. Then, the corresponding state after a few rounds has a certain property, *e.g.* the XOR of all texts in the set becomes 0 with probability 1. Throughout the paper, this property is called *balanced*.

A key recovery attack can be constructed by using this property. An attacker appends a few rounds to the end of the distinguisher. After she obtains a set of the ciphertexts, she guesses a part of round keys and performs the partial decryption up to the balanced state. If the guess is correct, the XOR sum of the results always becomes 0. Hence, the key space can be reduced.

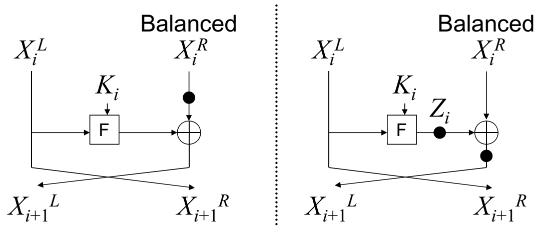

**Fig. 1.** Left: previous approach, Right: our approach

The application of the integral attack to AES [3, 4] and AES based ciphers is widely known. Moreover, for AES, Ferguson *et al.* proposed an improved technique called *partial-sum* [5], which utilizes the property of an MDS multiplication *e.g.* the MixColumns (MC) operation in AES. It observes that each output byte of the MC operation is a linear combination of four input bytes, $(x_0, x_1, x_2, x_3)$. Therefore the sum of the output value $\bigoplus \mathrm{MC}(x_0, x_1, x_2, x_3)$ can be computed in byte-wise independently, *i.e.* $\bigoplus(a_0 \cdot x_0) \oplus \bigoplus(a_1 \cdot x_1) \oplus \bigoplus(a_2 \cdot x_2) \oplus \bigoplus(a_3 \cdot x_3)$, where $a_0, a_1, a_2, a_3$ are some coefficients.

Another popular block-cipher construction is the Feistel network, which separates the state $X_i$ to the left half $X_i^L$ and the right half $X_i^R$. It updates the state by $X_{i+1}^R \leftarrow X_i^L$ and $X_{i+1}^L \leftarrow X_i^R \oplus F(X_i^L, K_i)$, where $F$ is called a round function and $K_i$ is a round key for updating $i$-th round. Variants of the Feistel network, *e.g.*, generalized or modified Feistel network are also popular designs.

Several papers have already applied the integral attack to ciphers with the Feistel network or its variant. In this paper, we call such ciphers *Feistel ciphers.* Examples are the attacks on Twofish [6], Camellia [7–10], CLEFIA [11, 12], SMS4 [13], Zodiac [14], HIGHT [15], and LBlock [16].

## Our Contributions

In this paper, an improvement for integral attacks against Feistel ciphers is discussed. The new technique can reduce the complexity of the key recovery phase. This possibly leads to an extension of the number of attacked rounds. The observation is described in the right-hand side of Fig. 1, which is very simple, but can improve many of previous integral attacks. Assume that the balanced state appears on the state $X_i^R$, thus an attacker examines if $\bigoplus(X_i^R) = 0$ or not. Due to the linearity of the computation, this can be transformed as $\bigoplus Z_i = \bigoplus X_{i+1}^L$, where $Z_i$ is the state after the round function is applied. Finally, we can compute the left-hand side and right-hand side of this equation independently, and the key candidates that result in the balanced state of $X_i^R$ are identified by checking the matches between two values. The match can be done with the meet-in-the-middle technique [17–19]. Therefore, the efficiency of the attack can be improved.

**Table 1.** Comparison of attack results

| Target | Key size | Approach | #Rounds | Data | Time | Memory (bytes) | Reference |
|---|---|---|---|---|---|---|---|
| LBlock | 80 bits | Imp. Diff. | 21 | $2^{62.5}$ | $2^{73.7}$ | $2^{55.5}$ | [21] |
| | | RK Imp. Diff. | 23 | $2^{40}$ | $2^{70}$ | – | [22] |
| | | Integral | 18 | $2^{62}$ | $2^{36}$ | $2^{20}$ | [16] |
| | | Integral | 18 | $2^{62}$ | $2^{12}$ | $2^{16}$ | This paper |
| | | Integral | 20 | $2^{63.6}$ | $2^{39.6}$ | $2^{35}$ | This paper |
| HIGHT | 128 bits | Imp. Diff. | 27 | $2^{58}$ | $2^{126.6}$ | $2^{120}$ | [23] |
| | | RK Diff. | 32 | $2^{57.84}$ | $2^{123.17}$ | – | [24] |
| | | Integral | 18 | $2^{62}$ | $2^{36}$ | $2^{20}$ | [16] |
| | | Integral | 22 | $2^{62}$ | $2^{118.71}$ | $2^{64}$ | [15] |
| | | Integral | 22 | $2^{62}$ | $2^{102.35}$ | $2^{64}$ | This paper |
| CLEFIA (-128) | 128 bits | Improb. Diff. | 13 | $2^{126.83}$ | $2^{126.83}$ | $2^{105.32}$ | [25] |
| | | Imp. Diff. | 13 | $2^{117.8}$ | $2^{121.2}$ | – | [26] |
| | | Integral | 12 | $2^{115.7}$ | $2^{116.7}$ | $2^{100}$ | [11] |
| | | Integral | 12 | $2^{115.7}$ | $2^{103.1}$ | $2^{75.2}$ | This paper |

The complexity for the integral attacks is only for recovering partial key bits and does not include the one for processing the data, while the complexity for the impossible/improbable differential attacks is for the full key recovery.
18-round attacks on LBlock recovers only 16 bits, and the exhaustive search on remaining 64 bits takes $2^{64}$ computations. However, we can avoid it by iterating the attack for other balanced bytes and recover more key bits.

Moreover, our technique can be combined with the partial-sum technique, which exploits another aspect of the independence in some computation[1].

We demonstrate the effect of our technique by applying it to several Feistel ciphers. The results are summarized in Table 1. The complexities for recovering partial key bits are compared for the integral attacks because this paper mainly focuses on the improvement of the key recovery phase inside the integral attack and does not pay attentions to the trivial additional exhaustive search of remaining key bits. The first application is a block-cipher LBlock [16]. We first show an improvement of the 18-round attack by [16]. [16] claimed that the attack could be extended up to 20 rounds. However, we show that the attack is flawed. Then, we construct a first successful integral attack against 20-round LBlock by using our technique. Moreover, we further reduce the complexity by applying the partial-sum technique. The second application is a block-cipher HIGHT [15]. We first show that the previous 22-round attack can be trivially improved, and then, the complexity is further reduced by using our technique. The last application is a block-cipher CLEFIA [12], which uses the generalized Feistel network, and its round function takes the SP function. We combine the partial-sum technique with our approach, and improve 12-round attack on CLEFIA-128.

[1] The same strategy is used in the dedicated attack on TWINE [20].

Note that for Feistel ciphers, an impossible/improbable differential attack is often the best analysis in the single-key setting. In fact, our approach only works for fewer rounds than those attacks. Nevertheless, we believe that presenting a new approach for improving integral attacks is useful, because integral attacks have already been established as a basic tool of the cryptanalysis. In Table 1, we also list the complexity of the current best related-key attack for comparison.

### Paper Outline

The organization of this paper is as follows. Section 2 gives preliminaries. Section 3 explains our basic idea to improve the integral analysis for Feistel ciphers. Section 4 applies our technique to LBlock, HIGHT, and CLEFIA-128. Finally, we conclude this paper in Section 5.

## 2 Preliminaries

### 2.1 Notations for Integral Attack

The integral attack is a cryptanalytic technique for symmetric-key primitives, which was firstly proposed by Daemen *et al.* to evaluate the security of the SQUARE cipher [1]. Its brief description has already given in Section 1. To discuss integral distinguishers, the following notations are used in this paper.

**"$A$ (Active)"** : all values appear exactly the same number in the set of texts.
**"$B$ (Balanced)"** : the XOR of all texts in the set is 0.
**"$C$ (Constant)"** : the value is fixed to a constant for all texts in the set.

### 2.2 Partial-Sum Technique

The partial-sum technique was introduced by Ferguson *et al.* [5] in order to improve the complexity of the key recovery phase in the integral attack. The original attack target was AES. In the key recovery phase of the AES, the partial decryption involves 5 bytes of the key and 4 bytes of the ciphertext. Suppose that the number of data to be analyzed, $n$, is $2^{32}$ and the byte position $b$ of each ciphertext is denoted by $C_{b,n}$. Then, the equation can be described as follows.

$$\bigoplus_{n=1}^{2^{32}} \Big[ S_4 \Big( S_0(c_{0,n} \oplus k_0) \oplus S_1(c_{1,n} \oplus k_1) \oplus S_2(c_{2,n} \oplus k_2) \oplus S_3(c_{3,n} \oplus k_3) \oplus k_4 \Big) \Big]. \quad (1)$$

With a straightforward method, the analysis takes $2^{32+40} = 2^{72}$ partial decryptions, while the partial-sum technique can perform this computation only with $2^{48}$ partial decryptions. The idea is partially computing the sum by guessing each key byte one after another.

The analysis starts from $2^{32}$ texts $(c_{0,n}, c_{1,n}, c_{2,n}, c_{3,n})$. First, two key bytes $k_0$ and $k_1$ are guessed, and $S_0(c_{0,n} \oplus k_0) \oplus S_1(c_{1,n} \oplus k_1)$ is computed for each guess. Let $x_{i,n}$ be $\bigoplus_{p=0}^{i}(S_p(c_{p,n} \oplus k_p))$[2]. Then, $S_0(c_{0,n} \oplus k_0) \oplus S_1(c_{1,n} \oplus k_1)$ can be represented by $x_{1,n}$, and Eq. (1) becomes

$$\bigoplus_{n=1}^{2^{32}} \Big[ S_4\Big( x_{1,n} \oplus S_2(c_{2,n} \oplus k_2) \oplus S_3(c_{3,n} \oplus k_3) \oplus k_4 \Big) \Big].$$

The original set includes $2^{32}$ texts, but now only 3-byte information $(x_1, c_2, c_3)$ is needed. Hence, by counting how many times each of 3-byte values $(x_1, c_2, c_3)$ appears and by only picking the values that appear odd times, the size of the data set is compressed into 3 bytes. For the second step, a single key byte $k_2$ is guessed, and the size of the data set becomes 2 bytes $(x_2, c_3)$. For the third step, a single key byte $k_3$ is guessed, and the size of the data set becomes 1 byte $(x_3)$. Finally, a single byte $k_4$ is guessed and Eq. (1) is computed for each guess.

The complexity for the guess of $k_0, k_1$ is $2^{16} \times 2^{32} = 2^{48}$, for the guess of $k_2$ is $2^{16} \times 2^{8} \times 2^{24} = 2^{48}$. Similarly, the complexity is preserved to be $2^{48}$ until the last computation.

### 2.3 Previous Integral Attack for Feistel Ciphers

Assume that the right half of the state in round $i$, denoted by $X_i^R$, has the balanced property. To recover the key, many of previous attacks use all information that relates to $X_i^R$. This is illustrated in the left-hand side of Fig. 1. Let $\#K(X)$ be the number of key bits that need to be guessed to obtain the value of $X$ by the partial decryption. Similarly, let $\#C(X)$ be the number of ciphertext bits that are used to obtain $X$ by the partial decryption. Many of previous attacks spend $2^{\#K(X_i^R)+\#C(X_i^R)}$ computations to obtain $\bigoplus(X_i^R)$.

## 3 Meet-in-the-Middle Technique for Integral Attacks

In this section, we explain our idea that improves the time complexity and the amount of memory to be used in the key recovery phase. The observation is very simple, but can improve many of previous integral attacks on Feistel ciphers.

Let $n$ be the number of texts. $\bigoplus_n(X_{i,n}^R)$ can be described as $\bigoplus_n(Z_{i,n} \oplus X_{i+1,n}^L)$, where $Z_i$ is the state after the round function is applied. We only use the notation $n$ to show that the sum of the value is later computed. The structure is illustrated in the right-hand side of Fig. 1. Due to the linear computation, the sum of each term can be computed independently. Hence, the equation $\bigoplus_n(X_{i,n}^R) = 0$, can be written as

$$\bigoplus_n Z_{i,n} = \bigoplus_n X_{i+1,n}^L. \tag{2}$$

Then, we compute $\bigoplus_n Z_{i,n}$ for all guesses of $\#K(Z_i)$ and store the result in a table, and independently compute $\bigoplus_n X_{i+1,n}^L$ for all guesses of $\#K(X_{i+1}^L)$ and

[2] Notation $x_{i,n}$ is somehow confusing. $x_{i,n}$ represents the sum of $i$ S-box outputs.

store the result in a table. Finally, the key values that result in the balanced state can be identified with the same manner as the meet-in-the-middle attack *i.e.* by checking the matches between two tables. The time complexity of the attack can be reduced into

$$\max\{2^{\#K(Z_i)+\#C(Z_i)}, 2^{\#K(X^L_{i+1})+\#C(X^L_{i+1})}\}. \tag{3}$$

Note that if the key bits to compute $Z_i$ and $X^L_{i+1}$ have some overlap, we can apply the three subset meet-in-the-middle attack [17] *i.e.*, the shared bits are firstly guessed, and for each guess, the other bits are independently computed. Let $K_s$ be a set of bits for the shared key, and $|K_s|$ is the bit number of $K_s$. Then, the memory complexity of the attack can be reduced into

$$\max\{2^{\#K(Z_i)+\#C(Z_i)-|K_s|}, 2^{\#K(X^L_{i+1})+\#C(X^L_{i+1})-|K_s|}\}. \tag{4}$$

Due to the structure of the Feistel network, the first item is always bigger than the second item. Thus, the time and memory complexity is simply written as

$$(\text{Time}, \text{Memory}) = \left(2^{\#K(Z_i)+\#C(Z_i)}, 2^{\#K(Z_i)+\#C(Z_i)-|K_s|}\right). \tag{5}$$

## 4 Applications of Our Technique

In this section, we demonstrate several applications of our technique by improving previous integral attacks against several ciphers. The goal of this section is to show that our technique can be applied to a wide range of Feistel ciphers. Therefore, we do not optimize each attack by looking inside of the key schedule. We omit its description, and assume that each round key is independent.

### 4.1 LBlock

LBlock is a light-weight block-cipher proposed at ACNS 2011 by Wu and Zhang [16]. The block size is 64-bits and the key size is 80 bits. It adopts a modified Feistel structure with 32 rounds, and its round function consists of the key addition, an S-box layer, and a permutation of the byte positions. The plaintext is loaded into an internal state $X^L_0 \| X^R_0$. The state $X^L_i \| X^R_i$ is updated by using a round key $K_i$ and the round function described in Fig. 2. We denote the $j$-th byte (1 byte is 4 bits for LBlock) of a 32-bit word $X$ by $X[j]$, where 0-th byte is the right most byte in the figure.

**Previous 18-Round Attack.** The designers showed a 15-round integral distinguisher. For a set of $2^{60}$ plaintexts with the form of $(AAAC\ AAAA\ AAAA\ AAAA)$, the state after 15 rounds, $(X^L_{15} \| X^R_{15})$, has the form of $(????\ ????\ ?B?B\ ?B?B)$. By using this property, the designers showed an 18-round key recovery attack. The key recovery phase is illustrated in Fig. 3. The attacker guesses a part of round keys, and decrypts the ciphertexts up to the fourth byte of $X^R_{15}$ and checks if its sum is 0 or not. As shown in Fig. 3, five bytes of the ciphertext ($X^L_{18}[0,6]$ and $X^R_{18}[1,4,6]$)

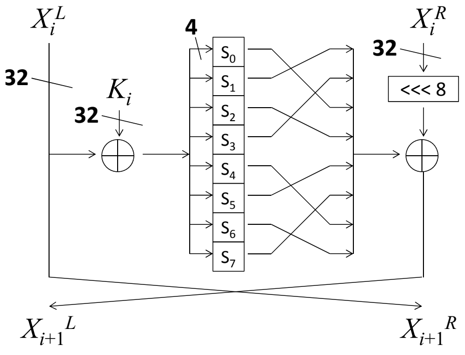

**Fig. 2.** LBlock Round function

and four bytes of keys ($K_{17}[1,4], K_{16}[4]$, and $K_{15}[4]$) relate to the partial decryption for $X_{15}^R[4]$. The attacker first counts how many times each of 5-byte values $X_{18}^L[0,6], X_{18}^R[1,4,6]$ appears and only picks values that appear odd times. Hence, at most $2^{4*5} = 2^{20}$ values are stored in a memory. Then, for each guess of four key bytes, she computes the corresponding $X_{15}^R[4]$ and computes the sum. The attack complexity is $2^{20} \times 2^{16} = 2^{36}$ partial decryptions.

**Improved 18-Round Attack.** The attack complexity can be improved by applying our technique. The condition $\bigoplus X_{15}^R[4] = 0$ can be written as

$$\bigoplus Z_{15}[6] = \bigoplus X_{16}^L[6]. \tag{6}$$

As shown in Fig. 3, the computation of $Z_{15}[6]$ involves three bytes of the ciphertext ($X_{18}^L[6]$ and $X_{18}^R[4,6]$) and three bytes of round keys ($K_{17}[4], K_{16}[4]$, and $K_{15}[4]$), which are denoted by numbers in red square brackets. Similarly, the computation of $X_{16}^L[6]$ involves two bytes of the ciphertext ($X_{18}^L[0]$ and $X_{18}^R[1]$) and a single byte of a round key ($K_{17}[1]$), which are denoted by numbers in blue round brackets. The attack procedure is as follows.

1. Query $2^{60}$ plaintexts which has the form of ($AAAC$ $AAAA$ $AAAA$ $AAAA$).
2. Prepare the memory which stores how many times each three-byte value $X_{18}^L[6], X_{18}^R[4,6]$ appears, and pick the values which appear odd times. Do the same for two-byte values $X_{18}^L[0], X_{18}^R[1]$.
3. For all three-byte keys $K_{17}[4], K_{16}[4]$, and $K_{15}[4]$, compute $Z_{15}[6]$ for all three-byte values $X_{18}^L[6], X_{18}^R[4,6]$ and store its sum in a list $L_{Z_{15}}$.
4. For all values of a single-byte key $K_{17}[1]$, compute $X_{16}^L[6]$ for all two-byte values $X_{18}^L[0], X_{18}^R[1]$ stored in the memory and store its sum in a list $L_{X_{16}^L}$.
5. Check the matches between $L_{Z_{15}}$ and $L_{X_{16}^L}$. If the matches are found, output the corresponding 4-byte keys as correct key candidates.

Step 2 requires a memory to store $2^{12}$ 3-byte values. Step 3 requires a complexity of $2^{12} \times 2^{12} = 2^{24}$ partial decryptions and a memory to store $2^{12}$ sums. Step 4

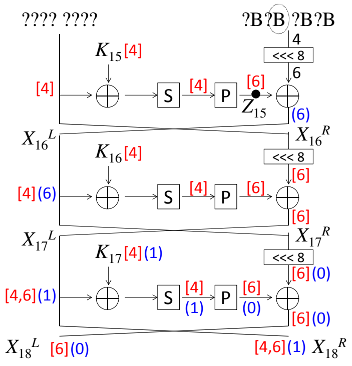

**Fig. 3.** 18-round attack on LBlock

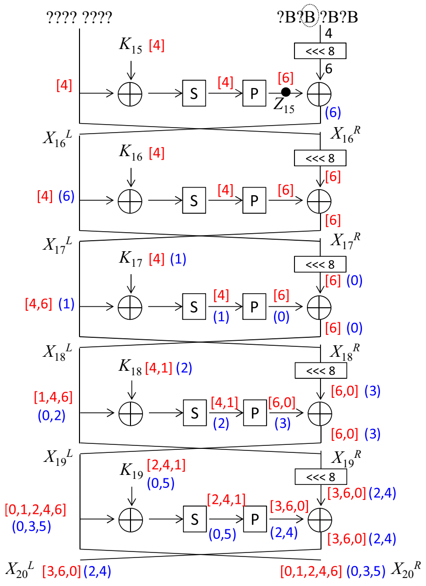

**Fig. 4.** 20-round attack on LBlock

requires a complexity of $2^4 \times 2^8 = 2^{12}$ partial decryptions and a memory to store $2^4$ sums. Step 5 is performed with $2^{12}$ table look-ups, and $2^{16} \times 2^{-4} = 2^{12}$ values are output as correct key candidates.

By iterating the above steps 3 times, a single key candidate is obtained, which requires $2^{24}$ 18-round LBlock computations and the memory to store $2^{12}$ LBlock state. This is faster than the previous 18-round attack. The data complexity is the same as the previous attack, which is $4 \times 2^{60} = 2^{62}$ chosen plaintexts.

**Further Improvement with the Partial-Sum Technique.** The attack complexity can be further improved with the partial-sum technique. the computation of $\bigoplus Z_{15}[6]$ can be written as follows:

$$\bigoplus S\Big[S\Big(S(X_{18}^R[4] \oplus K_{17}[4]) \oplus X_{18}^L[6] \oplus K_{16}[4]\Big) \oplus X_{18}^R[6] \oplus K_{15}[4]\Big]. \quad (7)$$

In the previous section, this was computed with $2^{24}$ computations, but we can compute it only with $2^{16}$ computations with the partial-sum technique. The analysis starts from 3-byte tuple $(X_{18}^L[6], X_{18}^R[4], X_{18}^R[6])$ with $2^{12}$ data. Firstly a single key byte $K_{17}[4]$ is guessed, and $S(X_{18}^R[4] \oplus K_{17}[4]) \oplus X_{18}^L[6]$ is computed for each guess. Let $y_1$ be the result. Then, the data can be compressed into 2-byte tuple $(y_1, X_{18}^R[6])$. Secondly, $K_{16}[4]$ is guessed and $S(y_1 \oplus K_{16}[4]) \oplus X_{18}^R[6]$ is computed for each guess. Let $y_2$ be the result. Then, the data can be compressed into 1-byte $y_2$. Finally, for each guess of $K_{15}[4]$, $\bigoplus Z_{15}[6]$ is computed by $S(y_2 \oplus K_{15}[4])$.

The guess of $K_{17}[4]$ requires $2^4 \cdot 2^{12} = 2^{16}$ computations. The guess of $K_{16}[4]$ requires $2^4 \cdot 2^4 \cdot 2^8 = 2^{16}$ computations. Finally, the guess of $K_{15}[4]$ requires $2^4 \cdot 2^4 \cdot 2^4 \cdot 2^4 = 2^{16}$ computations. In summary, the time complexity of the attack is reduced into $2^{16}$.

**Remarks.** The 18-round attack only recovers 16 bits of subkeys. Hence, the exhaustive search on the remaining 64 bits costs $2^{64}$, which is much more expensive. This can be avoided by performing the above procedure on the other balanced bytes. We stress that the queried data can be shared among the analysis for different balanced bytes. Hence, the data complexity keeps unchanged and only the time and memory complexity increases linearly.

**Extension to 20-Round Attack.** Wu and Zhang [16] claimed that the attack could be extended up to 20 rounds because only 12 key bytes relate to the partial decryption for $X^R_{15}[4]$. However, we show that this attack is flawed. It is true that 12 key bytes relate to $X^R_{15}[4]$, *i.e.*, $\#K(X^R_{15}[4]) = 48$. However, they did not consider the increase of the number of bytes in the ciphertext that relate to $X^R_{15}[4]$. The analysis is given in Fig. 4. It shows that $\#C(X^R_{15}[4])$ is 48 (12 bytes). Hence, their attack requires $2^{48+48} = 2^{96}$ partial decryptions, which is more expensive than the brute force attack.

In our approach, as shown in Fig. 4, both of $\#K(Z_{15}[6])$ and $\#C(Z_{15}[6])$ are 32 (8 bytes). Hence, the complexity to analyze a single set is reduced to $2^{32+32} = 2^{64}$, which is faster than the brute force attack. For each analysis, the key space becomes $2^{-4}$. Hence, by repeating the analysis 12 times, 12 bytes of the key space is reduced to 1. Moreover, similarly to the 18-round attack, the partial-sum technique can be applied to compute $\bigoplus Z_{15}[6]$. The details are omitted due to the limited space. For each guess of a single-key byte, the data is compressed by 1 byte. Hence, the final complexity becomes $2^4 \cdot 2^{32} = 2^{36}$.

Note that, the attack outputs 11 bytes of the key candidates ($2^{44}$) as a result of the first analysis. To store these candidates, more memory than for storing $\#C(Z_{15}[6])$ is necessary. This can be avoided by analyzing 4 sets of plaintexts simultaneously, and thus the effect of the matching part with the meet-in-the-middle approach becomes 4 times. As a result, the key space after the result of the first analysis becomes 8 bytes, which is the same size as $\#C(Z_{15}[6])$.

In summary, our attack is the first successful integral attack against 20-round LBlock with approximately $12 * 2^{36} \approx 2^{39.6}$ LBlock computations, $8 * 2^{32}$ bytes of memory, and $12 * 2^{60} \approx 2^{63.6}$ chosen plaintexts. The previous attack evaluated that 13 sets of plaintexts are necessary to recover the key with a high success probability. Under the same philosophy, our attack also requires $13 * 2^{36} \approx 2^{39.7}$ LBlock computations, and $2^{63.7}$ chosen plaintexts.

**Table 2.** Key schedule for the keys that relate to the key recovery phase

| $SK_{69}^0$ | $SK_{73}$ | $SK_{76}^0$ | $SK_{77}$ | $SK_{80}$ | $SK_{81}$ | $SK_{84}$ | $SK_{85}$ | $SK_{87}^0$ | $WK_5$ | $WK_6$ | $WK_7$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $MK_1^0$ | $MK_{13}$ | $MK_8^0$ | $MK_9$ | $MK_3$ | $MK_4$ | $MK_7$ | $MK_0$ | $MK_2^0$ | $MK_1$ | $MK_2$ | $MK_3$ |

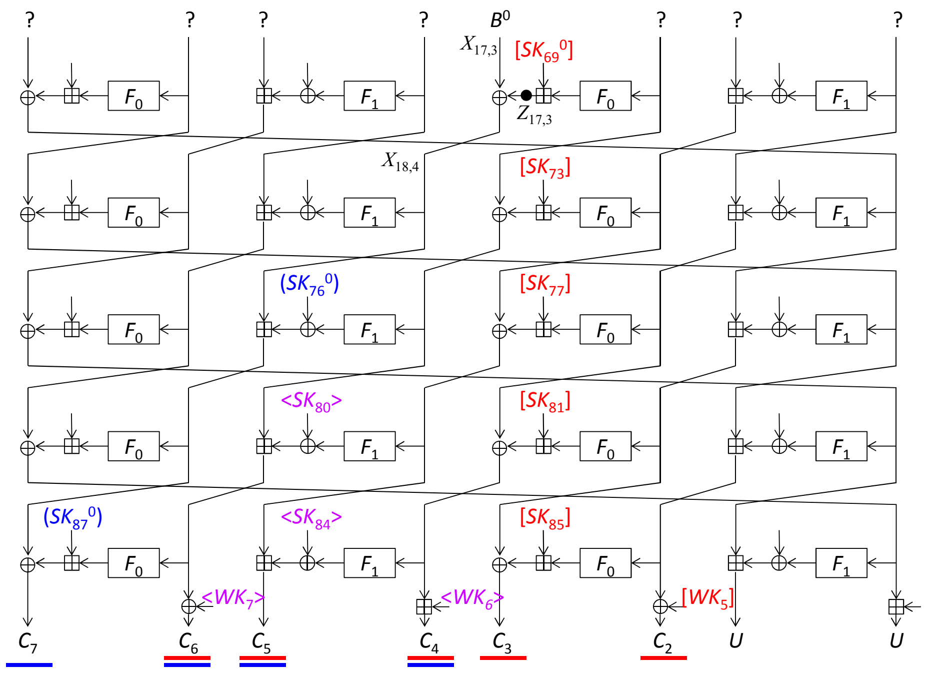

**Fig. 5.** Key recovery phase for 22-round HIGHT

## 4.2 HIGHT

HIGHT is a light-weight block-cipher proposed at CHES 2006 by Hong *et al.* [27]. The block size is 64 bits and the key size is 128 bits. It adopts the generalized Feistel structure with 8 branches and 32 rounds. The round function consists of the ARX structure. The plaintext is loaded into an internal state $X_{0,7} \| X_{0,6} \| \cdots \| X_{0,0}$. The state $X_{i,7} \| X_{i,6} \| \cdots \| X_{i,0}$ is updated by using round keys $SK_{4i}, SK_{4i+1}, SK_{4i+2}, SK_{4i+3}$ and the round function. We denote the $k$-th bit of a byte $X_{i,j}$ by $X_{i,j}^k$. Note that each round key is a copy of a part of the original secret key $K$, and which part is used is defined in the specification. Because the previous attack by Zheng *et al.* [15] has already exploited the relation of subkeys, we need to consider it to improve the attack.

**Previous 22-Round Attack.** Zheng *et al.* showed a 17-round integral distinguisher. For a set of $2^{56}$ plaintexts with the form of $(A, A, A, A, A, A, A, C)$, the state after 17 rounds, $(X_{17,7} \| X_{17,6} \| \cdots \| X_{17,0})$, has the form of $(?, ?, ?, ?, B^0, ?, ?, ?)$, where $B^0$ stands for the balanced state with respect to the

0-th bit. By using this property, Zheng *et al.* showed a 22-round key recovery attack. The key recovery phase for the 22-round attack is illustrated in Fig. 5.

The attacker prepares a set of $2^{56}$ plaintexts that satisfies the above form. As shown in Fig. 5, 48 bits of the ciphertext $(C_7, C_6, C_5, C_4, C_3, C_2)$ and 75 bits of round keys $(SK_{69}^0, SK_{73}, SK_{76}^0, SK_{77}, SK_{80}, SK_{81}, SK_{84}, SK_{85}, SK_{87}^0, WK_5, WK_6, WK_7)$ relate to the partial decryption for $X_{17,3}^0$. The related 75 bits of the round keys have some overlap with respect to the original secret key. The key schedule for these keys is given in Table 2. Here, $MK$ stands for "Master Key", which is the original secret key.

By considering Table 2, the number of bits that need to be guessed is 65. Zheng *et al.* guessed all 65 key bits, and applied the partial decryption for all ciphertexts. Therefore, they concluded that the attack complexity to analyze one set was $2^{56} \times 2^{65} = 2^{121}$ partial decryptions. As a result of analyzing one set, the key space can be reduced by 1 bit. Hence, the attack is repeated for 65 sets. Zheng *et al.* showed that the complexity for 65 iterations was $2^{56}(2^{65} + 2^{64} + \cdots + 2^1) \approx 2^{122}$ partial decryptions, which is equivalent to $2^{118.71}$ 22-round HIGHT encryptions.

**Simple Improvement of the Previous Attack.** We show that the attack by Zheng *et al.* can be improved very simply. Because the partial decryption involves only 48 bits of the ciphertext, the data to be analyzed can be reduced into $2^{48}$ ciphertexts. This is done by counting how many times each of 48-bit values appears, and only picking values which appear odd times.

Moreover, for the 7th byte of the ciphertext, $C_7$, only the least significant bit, $C_7^0$, is needed to compute $X_{17,3}^0$. Therefore, the data to be analyzed can be further reduced into $2^{41}$ ciphertexts.

In summary, the attack complexity becomes $2^{41}(2^{65} + 2^{64} + \cdots + 2^1) \approx 2^{107}$ partial decryptions, which is equivalent to $2^{103.71}$ 22-round HIGHT encryptions.

**Application of Our Technique.** By using our technique, the complexity can be further improved. The condition for $\bigoplus X_{17,3}^0 = 0$ is written as $\bigoplus X_{18,4}^0 = \bigoplus Z_{17,3}^0$. The partial decryption for $Z_{17,3}^0$ involves 73 bits of round keys and 40 bits of ciphertexts. The partial decryption for $X_{18,4}^0$ involves 34 bits of round keys and 25 bits of ciphertexts. If the key schedule is considered, $\#K(Z_{17,3}^0)$ is 64 and $\#K(X_{18,4}^0)$ is 26 respectively, while 24 key bits are overlapped. The dominant complexity is the computations for $\bigoplus Z_{17,3}^0$. According to Eq. (5), the time complexity is $2^{40+64} = 2^{104}$ partial decryptions to analyze a single set.

To reduce the key space into 1, the attack is iterated 65 times. However, the attack complexity cannot be evaluated as $2^{40}(2^{64} + 2^{63} + \cdots + 2^1)$ with the meet-in-the-middle approach. This is because the discarded key candidates are uniformly distributed in the key space. Hence, our strategy is applying the meet-in-the-middle approach 2 times to reduce the key space from $2^{65}$ to $2^{63}$, and then perform the previous attack method to further reduce the key space into 1. Finally, the attack complexity is $2^{104} + 2^{104} + 2^{41} \cdot (2^{63} + 2^{62} + \cdots + 2^1) \approx 2^{106}$ partial decryptions.

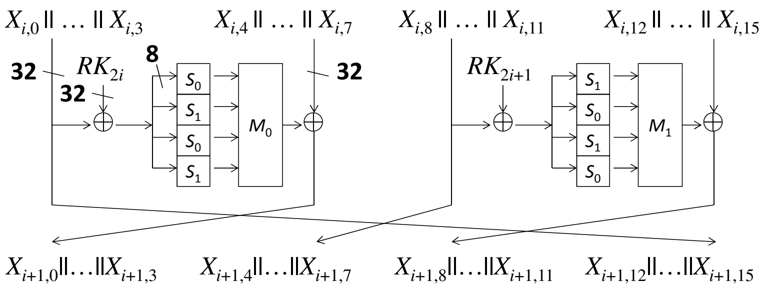
$X_{i,0} \| \dots \| X_{i,3}$
$X_{i,4} \| \dots \| X_{i,7}$
$X_{i,8} \| \dots \| X_{i,11}$
$X_{i,12} \| \dots \| X_{i,15}$
32
$RK_{2i}$
32
8
$S_0$
$S_1$
$S_0$
$S_1$
$M_0$
32
$RK_{2i+1}$
$S_1$
$S_0$
$S_1$
$S_0$
$M_1$
$X_{i+1,0} \| \dots \| X_{i+1,3}$
$X_{i+1,4} \| \dots \| X_{i+1,7}$
$X_{i+1,8} \| \dots \| X_{i+1,11}$
$X_{i+1,12} \| \dots \| X_{i+1,15}$

**Fig. 6.** Round function of CLEFIA

The previous work compared the complexity for one partial decryption and one HIGHT encryption by counting the number of $F$ functions to be calculated. 7 $F_0/F_1$ functions are involved in the partial decryption and $22 * 4 = 88$ $F_0/F_1$ functions are involved in the 22-round HIGHT encryptions. Hence, $2^{106}$ partial decryptions are equivalent to $2^{106} \times 7/88 \approx 2^{102.35}$ 22-round HIGHT encryptions. Note that $2^{64}$ key candidates are output as a result of the first analysis. To store these values, the memory to store $2^{64}$ keys is necessary.

In summary, the attack complexity becomes $2^{102.35}$ 22-round HIGHT encryptions, the memory to store $2^{64}$ keys, and $65 * 2^{56} \approx 2^{62}$ chosen plaintexts.

## 4.3 CLEFIA

CLEFIA is a block-cipher proposed at FSE 2007 by Shirai *et al.* [12]. The block size is 128 bits and the key size can be chosen from 128 bits, 192 bits, or 256 bits. It adopts the generalized Feistel structure with 4 branches and 18 rounds for a 128-bit key. The round function consists of the key addition, S-box application, and multiplication by an MDS matrix. The plaintext is loaded into an internal state $X_{0,0} \| X_{0,1} \| \cdots \| X_{0,15}$. The state $X_{i,0} \| X_{i,1} \| \cdots \| X_{i,15}$ is updated by using round keys $RK_{2i}, RK_{2i+1}$ and the round function described in Fig. 6.

Li *et al.* showed a 9-round integral distinguisher for CLEFIA [11]. A set of $2^{112}$ plaintexts should have the form of $(AAAA\ AAAA\ A'_0A'_1A'_2A'_3\ AAAA)$, where $A'_0$ is $v \oplus w$, $A'_1$ is $2v \oplus 8w$, $A'_2$ is $4v \oplus 2w$, $A'_3$ is $6v \oplus aw$, and $v$ and $w$ are two active bytes. Then, the state after 9 rounds, $(X_{9,0} \| X_{9,1} \| \cdots \| X_{9,15})$, has the form of $(????\ BBBB\ ????\ ????)$. By using this property, Li *et al.* showed a 11-round basic attack and a 12-round extended attack.

**Previous 11-Round Attack.** The key recovery phase is described in Fig. 7. An equivalent transformation is applied to the 10th round. $M_0^{-1}(X_{9,4}, X_{9,5}, X_{9,6}, X_{9,7})$ is still a balanced state because $M_0$ is a linear operation (an MDS matrix multiplication). The attacker guesses round keys and aims to detect if the sum of the 0-th byte of $M_0^{-1}(X_{9,4}, X_{9,5}, X_{9,6}, X_{9,7})$ is 0 or not. The equation that the attacker computes can be written as follows:

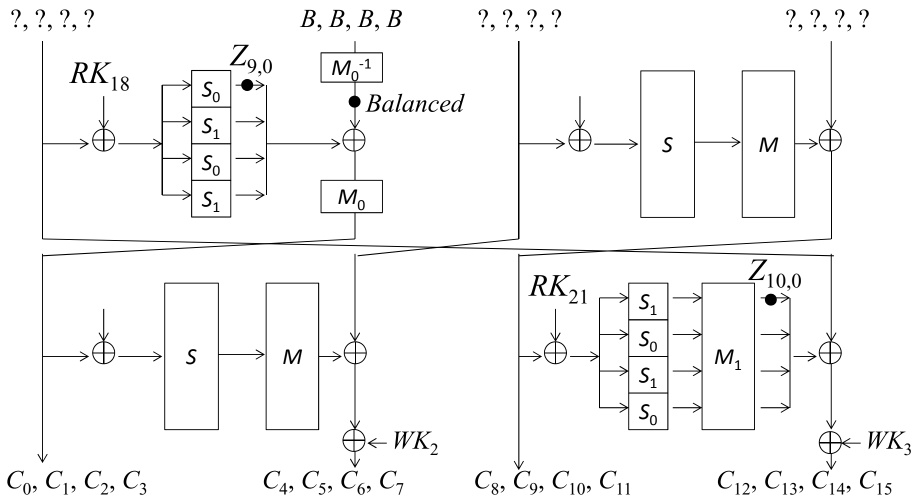

**Fig. 7.** Key recovery phase for 11-round CLEFIA

$$\bigoplus\Big[S_0\Big(S_1(C_8 \oplus RK_{21,0}) \oplus 08 \cdot S_0(C_9 \oplus RK_{21,1}) \oplus$$
$$02 \cdot S_1(C_{10} \oplus RK_{21,2}) \oplus 0a \cdot S_0(C_{11} \oplus RK_{21,3}) \oplus C_{12} \oplus RK'_{18,0}\Big)\Big]$$
$$= \bigoplus C', \tag{8}$$

where $RK'_{18,0} = WK_{3,0} \oplus RK_{18,0}$ and $C'$ is the 0-th byte of $M_0^{-1}(C_0, C_1, C_2, C_3)$, i.e., $C' = C_0 \oplus 02 \cdot C_1 \oplus 04 \cdot C_2 \oplus 06 \cdot C_3$. The simple method requires $2^{40+40} = 2^{80}$ partial decryptions, while Li *et al.* compute it only with $2^{56}$ partial decryptions by using the partial-sum technique.

For $2^{112}$ chosen-plaintexts, the attacker only picks 5-byte values $(C_8, C_9, C_{10}, C_{11}, C_{12})$ and 4-byte values $(C_0, C_1, C_2, C_3)$ that appear odd times. Then, the right-hand side of Eq. (8) can be computed with at most $2^{32}$ $M_0^{-1}$ computations. The computation for the left-hand side of Eq. (8) starts from $2^{40}$ texts of 5-byte values $(C_8, C_9, C_{10}, C_{11}, C_{12})$. For simplicity, let $t_0, t_1, \ldots, t_l$ and $r_0, r_1, \ldots, r_l$ be the ciphertext bytes and the corresponding key bytes. Then, let $x_i$ be $\bigoplus_{p=0}^{i} S(t_p \oplus r_p)$. Firstly, the attacker guesses two key bytes $r_0$ and $r_1$ and computes $x_1$. Then, the size of the data to be analyzed can be reduced to 4 bytes $(x_1, C_{10}, C_{11}, C_{12})$. Secondly, the attacker guesses a single key byte $r_2$, and computes $x_2$. Then, the size of the data can be reduced to 3 bytes $(x_2, C_{11}, C_{12})$. Similarly, the attacker guesses $r_3$ and picks 2-byte data $(x_3, C_{12})$, then guesses $r_4$ and computes the final sum. The complexity for computing $x_1$ is $2^{16} \cdot 2^{40} = 2^{56}$, for computing $x_2$ is $2^{16} \cdot 2^8 \cdot 2^{32} = 2^{56}$, and similarly the complexity of $2^{56}$ is preserved until the final sum is obtained. With the analysis for one data set, the key space becomes $2^{-8}$ times. Li *et al.* analyzed 6 data sets to uniquely determine the key. The final complexity was estimated as $2^{54}$ 11-round CLEFIA encryptions.

**Improving the Previous 11-Round Attack.** We show that the partial-sum technique in the 11-round attack by Li *et al.* can be improved without using our meet-in-the-middle technique. To compute the left-hand side of Eq. (8), Li *et al.* guessed two key bytes $r_0$ and $r_1$ to obtain $x_1$. This procedure seems to come from the original partial-sum application by Ferguson *et al.* [5], which guessed two key bytes at the first step. However, in the analysis for Feistel ciphers, the equation to compute the sum (the left-hand side of Eq. (8) for CLEFIA) has already included a term that only consists of a ciphertext byte (without a key byte). This is actually different from Eq. (1) for AES. Therefore, guessing only a single byte $r_0$ is enough to compress the data from $2^{40}$ to $2^{32}$.

In details, we firstly guess a single byte $RK_{21,0}$ and compute $S_1(C_8 \oplus RK_{21,0}) \oplus C_{12}$ for $2^{40}$ texts. Let $x'_0$ be the result of this computation. We then focus on a 4-byte tuple $x'_0, C_9, C_{10}, C_{11}$, and compress the data size to $2^{32}$ by only picking the values that appear odd times. For the second step, we guess a single byte $RK_{21,1}$ and compute $08 \cdot S_0(C_9 \oplus RK_{21,1}) \oplus x'_0$ for $2^{32}$ texts. Let $x'_1$ be the result. Then, the data size can be reduced to $2^{24}$ by focusing on 3-byte tuple $x'_1, C_{10}, C_{11}$. We continue the similar procedure until the final sum is obtained. The complexity for computing $x'_0$ is $2^8 \cdot 2^{40} = 2^{48}$, for computing $x'_1$ is $2^8 \cdot 2^8 \cdot 2^{32} = 2^{48}$, and similarly the complexity of $2^{48}$ is preserved until the final sum is obtained.

In summary, the attack complexity can be reduced by a factor of $2^8$, and the total complexity is reduced to approximately $2^{46}$ 11-round CLEFIA encryptions.

**Previous 12-Round Attack.** Based on our understandings, we explain the 12-round attack by Li *et al.* The key recovery phase is described in Fig. 8. Several equivalent transformations are applied. An important property is that the whitening key $WK_3$ only affects the balanced state linearly. Therefore, after taking the sum of $2^{112}$ texts, the impact of $WK_3$ disappears. Hereafter, $WK_3$ is ignored. The equation that the attacker computes can be written as follows:

$$\begin{aligned}\bigoplus\Big[&S_0\Big(S_1(b_0 \oplus RK'_{21,0}) \oplus 08 \cdot S_0(b_1 \oplus RK'_{21,1}) \oplus 02 \cdot S_1(b_2 \oplus RK'_{21,2}) \oplus \\ &\quad 0a \cdot S_0(b_3 \oplus RK'_{21,3}) \oplus C_8 \oplus RK_{18,0}\Big) \oplus \Big(y_1 \cdot S_1(C_8 \oplus RK_{23,0}) \oplus \\ &\quad y_2 \cdot S_0(C_9 \oplus RK_{23,1}) \oplus y_3 \cdot S_1(C_{10} \oplus RK_{23,2}) \oplus y_4 \cdot S_0(C_{11} \oplus RK_{23,3})\Big)\Big] \\ = \bigoplus C',\end{aligned} \tag{9}$$

where $(b_0, b_1, b_2, b_3)$ is a 4-byte word for $(X_{10,8}, X_{10,9}, X_{10,10}, X_{10,11})$, $y_0, y_1, y_2, y_3$ are coefficients derived from $M_1$ and $M_0^{-1}$, $C'$ is the 0-th byte of $M_0^{-1}(C_{12}, C_{13}, C_{14}, C_{15})$, and $RK'_{21}$ is $RK_{21} \oplus WK_2$. The computation of $\bigoplus C'$ is done with at most $2^{32}$ computations. The attacker firstly guesses 4 key bytes $RK_{22,0}, \ldots, RK_{22,3}$ and computes $(b_0, b_1, b_2, b_3)$ for all texts. Then, the left-hand side of Eq. (9) is computed with the partial-sum technique. We omit the detailed procedure. Li *et al.* concluded that the complexity to analyze one set is $2^{120}$ partial decryptions.

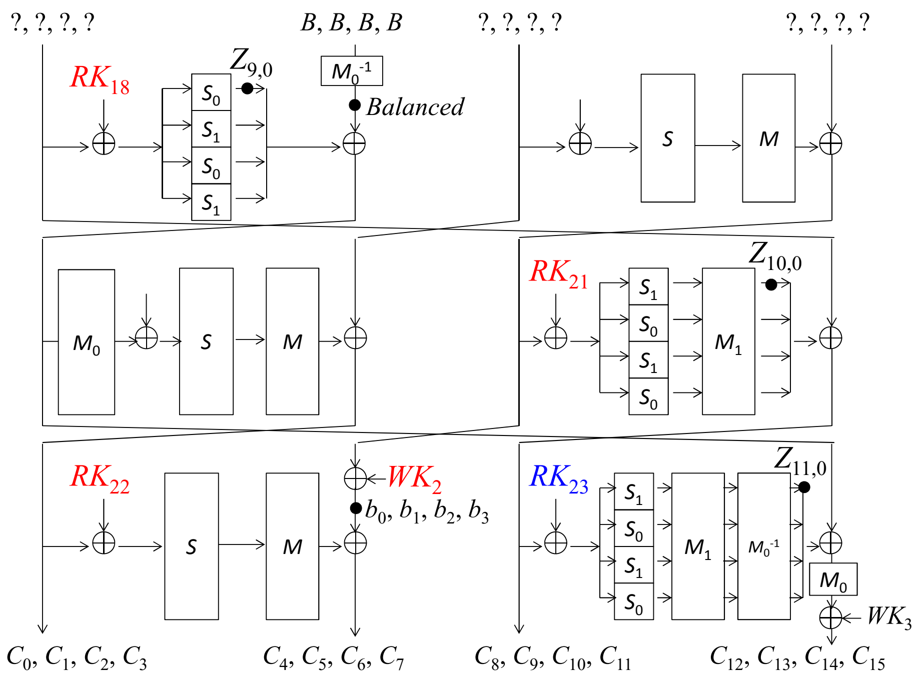

**Fig. 8.** Key recovery phase for 12-round CLEFIA

**Improved 12-Round Attack.** The attack by Li *et al.* can be improved by introducing the meet-in-the-middle approach. Eq. (9) is transformed as follows;

$$\begin{aligned}
&\bigoplus S_0\Big(S_1(b_0 \oplus RK'_{21,0}) \oplus 08 \cdot S_0(b_1 \oplus RK'_{21,1}) \oplus 02 \cdot S_1(b_2 \oplus RK'_{21,2}) \oplus \\
&\qquad 0a \cdot S_0(b_3 \oplus RK'_{21,3}) \oplus C_8 \oplus RK_{18,0}\Big) \\
&= \bigoplus\Big(y_1 \cdot S_1(C_8 \oplus RK_{23,0}) \oplus y_2 \cdot S_0(C_9 \oplus RK_{23,1}) \oplus y_3 \cdot S_1(C_{10} \oplus RK_{23,2}) \oplus \\
&\qquad y_4 \cdot S_0(C_{11} \oplus RK_{23,3})\Big) \oplus \bigoplus C'. \qquad (10)
\end{aligned}$$

The attack procedure for each set of $2^{112}$ texts is as follows.

1. With processing $2^{112}$ plaintexts, we count how many times each value of 9-byte tuple $(C_0, C_1, \ldots, C_8)$, each value of 4-byte tuple $(C_8, C_9, C_{10}, C_{11})$, and each value of 4-byte tuple $(C_{12}, C_{13}, C_{14}, C_{15})$ appears.
2. We compute the second term of the right-hand side of Eq. (10), i.e., $\bigoplus C'$.
3. We compute the first term of the right-hand side of Eq. (10) for the exhaustive guess of $RK_{23}$, and compute XOR with $\bigoplus C'$. The result, which is the right-hand side of Eq. (10) for each guess of $RK_{23}$, is stored in a list $L_{X_{10,0}}$.
4. For each guess of $RK_{22}$ (in total $2^{32}$ iterations), we do as follows.

   (a) For $2^{72}$ texts $(C_0, C_1, \ldots, C_8)$, we compute $(b_0, b_1, b_2, b_3)$ and count how many times each of 5-byte tuple $(b_0, b_1, b_2, b_3, C_8)$ appears. Therefore, the data size is compressed into $2^{40}$, and the equation becomes exactly the same as the left-hand side of Eq. (10).
   (b) We compute the left-hand side of Eq. (10) by guessing five key bytes $RK'_{21}, RK_{18,0}$ with the same method as our improved 11-round attack. The result is stored in a list $L_{z_{9,0}}$.
5. Finally, we identify right-key candidates by searching for matches between two lists $L_{X_{10,0}}$ and $L_{z_{9,0}}$.

Step 2 requires at most $2^{32}$ computations. Step 3 requires at most $2^{64}$ computations with the straightforward method, which is already enough small. This can be further reduced into $2^{48}$ computations with the partial-sum technique. After Step 3, we obtain a list $L_{X_{10,0}}$ with $2^{32}$ entries. Step 4(a) requires $2^{32} \cdot 2^{72} = 2^{104}$ partial decryptions. Because our 11-round attack requires $2^{48}$ partial decryptions, Step 4(b) requires $2^{32} \cdot 2^{48} = 2^{80}$ partial decryptions. As a result of Step 5, we expect to obtain $2^{32+72-8} = 2^{96}$ matches, because the key space is reduced by a factor of $2^8$ with the analysis of a single set.

By iterating the analysis with 13 different sets, we expect to obtain a unique solution of 13 key bytes. Note that, by analyzing 4 or more sets simultaneously, the efficiency of the match becomes 4 times or more, and the number of right-key candidates becomes $2^{32+72-(4*8)} = 2^{72}$ or less. This can avoid using $2^{96}$ memory after the analysis of the first set.

In summary, the bottle-neck of the complexity is Step 4(a), which requires $2^{104}$ 0.5-round computations. This is equivalent to $2^{104}/24 \approx 2^{99.4}$ 12-round CLEFIA encryptions. After iterating the procedure for 13 sets, the complexity becomes $13 \cdot 2^{99.4} \approx 2^{103.1}$ 12-round CLEFIA computations. The bottle-neck of the memory is for counting how many times each value of 9-byte tuple $(C_0, C_1, \ldots, C_8)$ appears for $2^{112}$ ciphertexts, which requires $2^{72}$ 9-byte information. This is equivalent to $2^{75.2}$ bytes or $2^{71.2}$ CLEFIA state. The data complexity is the same as the previous work, which is $13 \cdot 2^{112} \approx 2^{115.7}$ chosen plaintexts.

## 5 Concluding Remarks

In this paper, we showed an improvement for the integral analysis against Feistel ciphers, which recovers the key by using the meet-in-the-middle approach. We focus on the independence of two computations in the partial decryption for Feistel ciphers, and it reduces the time and memory for the key-recovery phase. Our technique can be combined with the partial-sum technique. We applied our technique for several Feistel ciphers, and showed that the previous integral attacks on LBlock, HIGHT, and CLEFIA-128 could be improved. Particularly, the number of attacked rounds with integral analysis was extended for LBlock.

One possible future work is deriving the limitation of the integral attack, i.e., how many rounds can be potentially attacked by combining currently known techniques such as the meet-in-the-middle and partial-sum techniques. As was done in this paper, the integral attack seems to have more room to be improved.

Then, presenting new techniques is an interesting topic, e.g., application of the partial-sum technique for HIGHT. Because HIGHT adopts two non-commutative operations, XOR and modular addition, the application of the partial-sum is not obvious. We leave it as an open problem.

**Acknowledgments.** We would like to thank the anonymous reviewers for many helpful comments, especially for the potential application of the partial-sum technique to LBlock.

## References

1. Daemen, J., Knudsen, L.R., Rijmen, V.: The Block Cipher SQUARE. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 149–165. Springer, Heidelberg (1997)
2. Knudsen, L.R., Wagner, D.: Integral Cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 112–127. Springer, Heidelberg (2002)
3. Daemen, J., Rijmen, V.: AES Proposal: Rijndael (1998)
4. Daemen, J., Rijmen, V.: The design of Rijndeal: AES – the Advanced Encryption Standard (AES). Springer (2002)
5. Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D., Whiting, D.L.: Improved Cryptanalysis of Rijndael. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 213–230. Springer, Heidelberg (2001)
6. Lucks, S.: The Saturation Attack - A Bait for Twofish. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 1–15. Springer, Heidelberg (2002)
7. He, Y., Qing, S.: Square Attack on Reduced Camellia Cipher. In: Qing, S., Okamoto, T., Zhou, J. (eds.) ICICS 2001. LNCS, vol. 2229, pp. 238–245. Springer, Heidelberg (2001)
8. Lei, D., Chao, L., Feng, K.: New Observation on Camellia. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 51–64. Springer, Heidelberg (2006)
9. Duo, L., Li, C., Feng, K.: Square Like Attack on Camellia. In: Qing, S., Imai, H., Wang, G. (eds.) ICICS 2007. LNCS, vol. 4861, pp. 269–283. Springer, Heidelberg (2007)
10. Yeom, Y., Park, S., Kim, I.: On the Security of CAMELLIA against the Square Attack. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 89–99. Springer, Heidelberg (2002)
11. Li, Y., Wu, W., Zhang, L.: Improved Integral Attacks on Reduced-Round CLEFIA Block Cipher. In: Jung, S., Yung, M. (eds.) WISA 2011. LNCS, vol. 7115, pp. 28–39. Springer, Heidelberg (2012)
12. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-Bit Blockcipher CLEFIA (Extended Abstract). In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 181–195. Springer, Heidelberg (2007)
13. Liu, F., Ji, W., Hu, L., Ding, J., Lv, S., Pyshkin, A., Weinmann, R.-P.: Analysis of the SMS4 Block Cipher. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 158–170. Springer, Heidelberg (2007)
14. Ji, W., Hu, L.: Square Attack on Reduced-Round Zodiac Cipher. In: Chen, L., Mu, Y., Susilo, W. (eds.) ISPEC 2008. LNCS, vol. 4991, pp. 377–391. Springer, Heidelberg (2008)
15. 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)

16. 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)
17. Bogdanov, A., Rechberger, C.: A 3-Subset Meet-in-the-Middle Attack: Cryptanalysis of the Lightweight Block Cipher KTANTAN. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 229–240. Springer, Heidelberg (2011)
18. Chaum, D., Evertse, J.-H.: Cryptanalysis of DES with a Reduced Number of Rounds. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 192–211. Springer, Heidelberg (1986)
19. Diffie, W., Hellman, M.E.: Exhaustive cryptanalysis of the NBS Data Encryption Standard. Computer 6(10) (1977)
20. 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. 340–355. Springer, Heidelberg (2012)
21. Liu, Y., Gu, D., Liu, Z., Li, W.: Impossible Differential Attacks on Reduced-Round LBlock. In: Ryan, M.D., Smyth, B., Wang, G. (eds.) ISPEC 2012. LNCS, vol. 7232, pp. 97–108. Springer, Heidelberg (2012)
22. Minier, M., Naya-Plasencia, M.: A related key impossible differential attack against 22 rounds of the lightweight block cipher LBlock. Inf. Process. Lett. 112(16), 624–629 (2012)
23. Chen, J., Wang, M., Preneel, B.: Impossible Differential Cryptanalysis of the Lightweight Block Ciphers TEA, XTEA and HIGHT. In: Mitrokotsa, A., Vaudenay, S. (eds.) AFRICACRYPT 2012. LNCS, vol. 7374, pp. 117–137. Springer, Heidelberg (2012)
24. Koo, B., Hong, D., Kwon, D.: Related-Key Attack on the Full HIGHT. In: Rhee, K.-H., Nyang, D. (eds.) ICISC 2010. LNCS, vol. 6829, pp. 49–67. Springer, Heidelberg (2011)
25. Tezcan, C.: The Improbable Differential Attack: Cryptanalysis of Reduced Round CLEFIA. In: Gong, G., Gupta, K.C. (eds.) INDOCRYPT 2010. LNCS, vol. 6498, pp. 197–209. Springer, Heidelberg (2010)
26. Mala, H., Dakhilalian, M., Shakiba, M.: Impossible differential attacks on 13-round CLEFIA-128. J. Comput. Sci. Technol. 26(4), 744–750 (2011)
27. Hong, D., Sung, J., Hong, S.H., Lim, J.-I., Lee, S.-J., Koo, B.-S., Lee, C.-H., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J.-S., Chee, S.: 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)