# Improved Meet-in-the-Middle Distinguisher on Feistel Schemes

Li Lin[1,2,3(✉)], Wenling Wu[1,2,3], and Yafei Zheng[1,2,3]

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

**Abstract.** Improved meet-in-the-middle cryptanalysis with efficient tabulation technique has been shown to be a very powerful form of cryptanalysis against SPN block ciphers, especially AES. However, few results have been proposed on Balanced-Feistel-Networks (BFN) and Generalized-Feistel-Networks (GFN). This is due to the stagger of affected trail and special truncated differential trail in the precomputation phase, i.e. these two trails differ a lot from each other for BFN and GFN ciphers. In this paper, we describe an efficient and generic algorithm to search for an optimal improved meet-in-the-middle distinguisher with efficient tabulation technique on word-oriented BFN and GFN block ciphers. It is based on recursive algorithm and greedy algorithm. To demonstrate the usefulness of our approach, we show key recovery attacks on 14/16-round CLEFIA-192/256 which are the best attacks. We also give key recovery attacks on 13/15-round Camellia-192/256 (without $FL/FL^{-1}$).

**Keywords:** Block ciphers · Improved meet-in-the-middle attack · Efficient tabulation technique · Automatic search algorithm · CLEFIA · Camellia

## 1 Introduction

Meet-in-the-middle attack is first proposed by Diffie and Hellman to attack DES [8]. In recent years, it is widely researched due to its effectiveness against block cipher AES [4]. For AES, Gilbert and Minier show in [12] some collision attacks on 7-round AES. At FSE 2008, Demirci and Selçuk improve the Gilbert and Minier attacks using meet-in-the-middle technique instead of collision idea. More specifically, they show that the value of each byte of 4-round AES ciphertext can be described by a function of the $\delta$-*set*, i.e. a set of 256 plaintexts where a byte (called active byte) can take all values and the other 15 bytes are constant, parameterized by 25 [5] and 24 [6] 8-bit parameters. The last improvement is due to storing differences instead of values. This function is used to build a distinguisher in the offline phase, i.e. they build a lookup table containing all the


O. Dunkelman and L. Keliher (Eds.): SAC 2015, LNCS 9566, pp. 122–142, 2016.
DOI: 10.1007/978-3-319-31301-6_7

possible sequences constructed from a $\delta$-set. In the online phase, they identify a $\delta$-set, and then partially decrypt the $\delta$-set through some rounds and check whether it belongs to the table. At ASIACRYPT 2010, Dunkelman, Keller and Shamir develop many new ideas to solve the memory problems of the Demirci and Selçuk attacks [10]. First of all, they only store *multiset*, i.e. an unordered sequence with multiplicity, rather than the ordered sequence. The second and main idea is the differential enumeration technique which uses a special property on a truncated differential characteristic to reduce the number of parameters that describes the set of functions from 24 to 16. Furthermore, Derbez, Fouque and Jean present a significant improvement to the Dunkelman et al. attacks at EUROCRYPT 2013 [7], called efficient tabulation technique. Using this rebound-like idea, they show that many values in the precomputation table are not reached at all under the constraint of a special truncated differential trail. Actually, the size of the precomputation table is determined by 10 byte-parameters only. At FSE 2014, Li et al. give an attack on 9-round AES-192 using some relations among subkeys [17].

In [18], Lin et al. summarize former works of improved MITM distinguisher, and then define *T-$\delta$-set* which is a special *$\delta$-set* of $T$ active cells and *S-multiset* which is a *multiset* of $S$ cells. With these definitions, they get affected trail which is a function connecting a *T-$\delta$-set* to an *S-multiset* with the minimal number of active cells after $R$-round encryption. After that, they introduce a general algorithm to search for the affected trail from a *T-$\delta$-set* to an *S-multiset*, and find that building a better distinguisher is equivalent to a positive integer optimization problem.

Although new results on Substitution-Permutation Networks(SPN) block ciphers using improved meet-in-the-middle attack with efficient tabulation technique are given in many literatures [7,16–18], few results have been proposed on Balanced-Feistel-Networks (BFN) [14] and Generalized-Feistel-Networks (GFN) [23][1]. This is due to the stagger of affected trail and special truncated differential trail in the precomputation phase, i.e. these two trails differ a lot from each other for BFN and GFN ciphers. However, they are almost the same for SPN block ciphers.

**Our Contribution.** In this paper, we describe an efficient and generic algorithm to search for an optimal improved meet-in-the-middle distinguisher with efficient tabulation technique on word-oriented BFN and GFN block ciphers. It is based on recursive algorithm and greedy algorithm. Given an affected trail $\mathcal{D}$ connecting a *T-$\delta$-set* to an *S-multiset* by the algorithm of [18], our algorithm can get an optimal truncated differential trail. The algorithm is made up of two phases: table construction phase and searching phase.

The table construction phase is based on the precomputation phase proposed by Fouque et al. in [11]. In this phase, we build a 2-equipartite directed acyclic graph $G$ containing all the possible one-round transitions.

The searching phase is based on the algorithm proposed by Matsui to find the best differential characteristics for DES [22]. Our algorithm works by recursion

[1] [13] was published after the accomplishment of this paper.

and can be seen as a tree traversal in a depth-first manner. One truncated differential trail is a path in this tree, and its weight equals the product of all traversed edges. We are looking for the path with the minimum number of guessed-cells in this tree under certain transition probability. The knowledge of the previous best truncated differential trail allows pruning during the procedure. To speed up this algorithm, We also use greedy algorithm to divide the search process into 2 parts: one starts from the beginning, the other one starts from the end.

To demonstrate the usefulness of our approach, we apply our algorithm to CLEFIA-192/256 [24] and Camellia-192/256 (without $FL/FL^{-1}$)[2] [1]. For CLEFIA-256, we give a 10-round distinguisher, and then give a 16-round key recovery attack with data complexity of $2^{113}$ chosen-plaintexts, time complexity of $2^{219}$ encryptions and memory complexity of $2^{210}$ 128-bit blocks. To the best of our knowledge, this is currently the best attack with respect to the number of attacked rounds. For CLEFIA-192, we give a 9-round distinguisher, and then give a 14-round key recovery attack with data complexity of $2^{113}$ chosen-plaintexts, time complexity of $2^{155}$ encryptions and memory complexity of $2^{146}$ 128-bit blocks. To the best of our knowledge, this is currently the best attack with respect to the complexity.

We also give an 8-round distinguisher on Camellia*-192, and then give a 13-round key recovery attack with data complexity of $2^{113}$ chosen-plaintexts, time complexity of $2^{180}$ encryptions and memory complexity of $2^{130}$ 128-bit blocks. For Camellia*-256, we give a 9-round distinguisher, and then give a 15-round key recovery attack with data complexity of $2^{113}$ chosen-plaintexts, time complexity of $2^{244}$ encryptions and memory complexity of $2^{194}$ 128-bit blocks. Although Lu et al. proposed a 14-round attack on Camellia*-192 and a 16-round attack on Camellia*-256 [21], they didn't consider the whitening operations. So we think our works on Camellia* have certain significance.

We present here a summary of our attack results on CLEFIA and Camellia*, and compare them to the best attacks known for them. This summary is given in Table 1.

**Organization of the paper.** The rest of this paper is organized as follows. Section 2 provides notations and definitions used throughout this paper, and then gives a brief review of the previous improved MITM distinguishers. Section 2 also gives the general attack scheme, and discusses distinguisher on Feistel schemes with efficient tabulation technique. We provide the automatic search algorithm to search for an optimal improved meet-in-the-middle distinguisher with efficient tabulation technique on Feistel schemes in Sect. 3. Our attacks on CLEFIA-192/256 and Camellia*-192/256 are described in Sect. 4. Finally, we conclude this paper in Sect. 5.

## 2 Preliminaries

In this section, we give notations used throughout this paper, and then briefly recall the previous improved MITM distinguishers, and after that the attack

[2] We call it Camellia* in this paper.

**Table 1.** Summary of the best attacks on CLEFIA-192/256, Camellia*-192/256.

| Cipher | Attack type | Rounds | Data | Memory (Bytes) | Time (Euc) | Source |
|---|---|---|---|---|---|---|
| CLEFIA-192 | Improbable | 14 | $2^{127.0}$ CPs | $2^{127.0}$ | $2^{183.2}$ | [25] |
| | Multidim. ZC | 14 | $2^{127.5}$ KPs | $2^{115}$ | $2^{180.2}$ | [2] |
| | Improved MITM | 14 | $2^{113}$ CPs | $2^{150}$ | $2^{155}$ | Sect. 4.1 |
| CLEFIA-256 | Improbable | 15 | $2^{127.4}$ CPs | $2^{127.4}$ | $2^{247.5}$ | [25] |
| | Multidim. ZC | 15 | $2^{127.5}$ KPs | $2^{115}$ | $2^{244.2}$ | [2] |
| | Improved MITM | 16 | $2^{113}$ CPs | $2^{214}$ | $2^{219}$ | Sect. 4.1 |
| Camellia*-192 | Impossible Diff | 12 | $2^{119}$ CPs | $2^{124}$ | $2^{147.3}$ | [19] |
| | Impossible Diff | $14^{ww}$ | $2^{117}$ CPs | $2^{122.1}$ | $2^{182.2}$ | [20] |
| | HO MITM | $14^{ww}$ | $2^{118}$ CPs | $2^{166}$ | $2^{164.6}$ | [21] |
| | Improved MITM | 13 | $2^{113}$ CPs | $2^{134}$ | $2^{180}$ | Sect. 4.2.1 |
| Camellia*-256 | Impossible Diff | $15^{ww}$ | $2^{122.5}$ KPs | $2^{233}$ | $2^{236.1}$ | [3] |
| | Impossible Diff | $16^{ww}$ | $2^{123}$ KPs | $2^{129}$ | $2^{249}$ | [20] |
| | HO MITM | $16^{ww}$ | $2^{120}$ CPs | $2^{230}$ | $2^{252}$ | [21] |
| | Improved MITM | 15 | $2^{113}$ CPs | $2^{198}$ | $2^{244}$ | Sect. 4.2.2 |

KPs: Known-Plaintexts. CPs: Chosen-Plaintexts. ww: Without Whiten Operation

scheme is given. Finally, we discuss the improved meet-in-the-middle distinguisher on Feistel schemes with efficient tabulation technique.

### 2.1 Notation

The following notations will be used throughout this paper (the sizes are counted in number of cells):

- $b$: number of cells in a state.
- $k$: the size of the master key.
- $o$: the size of one branch.
- $r$: number of rounds.
- $c$: number of bits in a cell.
- $n$: number of branches that will get through $F$-function in a state.
- $|X|$: number of active cells in a state $X$.
- $x_i$: the input state of round-$i$.
- $y_i$: the state after the key addition layer of round-$i$.
- $z_i$: the state after the S-box layer of round-$i$.
- $w_i$: the state after the linear transformation layer of round-$i$.
- $s[i]$: the $i^{th}$ branch of a state.
- $s[i][j]$: the $j^{th}$ cell in the $i^{th}$ branch of a state.
- $K_i[j]$: the $j^{th}$ cell of the $i^{th}$ round key.

### 2.2 Reviews of Former Works

In this section, we present a unified view of the previously known MITM distinguishers on AES in [5,7,10] and the algorithm to search for the affected trail given by Lin et al. in [18]. Let's start with particular structures of messages captured by Definitions 1 and 2.

**Definition 1 ($\delta$-set, [4]).** *Let a $\delta$-set be a set of 256 AES-states that are all different in one state byte (active byte) and all equal in the other state bytes (inactive bytes).*

**Definition 2 (Multisets of bytes, [7]).** *A multiset generalizes the set concept by allowing elements to appear more than once. Here, a multiset of 256 bytes can take as many as $\binom{2^8+2^8-1}{2^8} \approx 2^{506.17}$ different values.*

**Proposition 1 (Differential Property of S, [7]).** *Given $\Delta_i$ and $\Delta_0$ two non-zero differences, the equation of S-box*

$$S(x) \oplus S(x \oplus \Delta_i) = \Delta_0, \tag{1}$$

*has one solution in average.*

**Demirci and Selçuk distinguisher.** Consider the set of functions

$$f : \{0,1\}^8 \longrightarrow \{0,1\}^8$$

that maps a byte of a $\delta$-set to another byte of the state after four AES rounds. A convenient way is to view $f$ as an ordered byte sequence $(f(0), \ldots, f(255))$ so that it can be represented by 256 bytes. The crucial observation made by the generalizing Gilbert and Minier attacks [12] is that this set is tiny since it can be described by 25 byte-parameters ($2^{25 \cdot 8} = 2^{200}$) compared with the set of all functions of this type which counts as may as $2^{8 \cdot 2^8} = 2^{2048}$ elements [5]. Considering the differences $(f(0) - f(0), f(1) - f(0), \ldots, f(255) - f(0))$ rather than values, the set of functions can be described by 24 parameters [6]. The 24 byte-parameters which map $x_1[0]$ to $\Delta x_5[0]$ are presented as gray cells in Fig. 1.

**Dunkelman et al. Distinguisher and Derbez et al. Distinguisher.** In [10], Dunkelman et al. introduced two new improvements to further reduce the memory complexity of [6]. The first uses *multiset* which is an unordered sequence

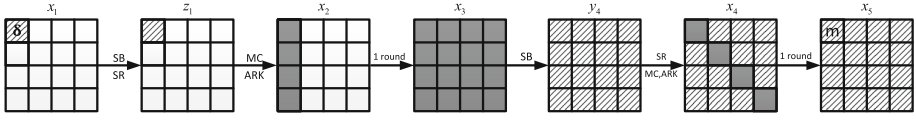


**Fig. 1.** The 4-round AES distinguisher used in [6]. The gray cells represent 24 byte-parameters, $\delta$ represents the $\delta$-set and $m$ represents the differential sequence to be stored.

with multiplicity to replace ordered sequence in the offline phase, since there is enough information so that the attack succeeds. The second improvement uses a novel idea named differential enumeration technique. The main idea of this technique is to use a special 4-round property on a truncated differential characteristic to reduce the number of parameters which describes the set of functions from 24 to 16.

In [7], Derbez et al. presented an improvement to Dunkelman et al.'s differential enumeration technique, called efficient tabulation technique (Proposition 2). Combining with the rebound-like idea, many values in the precomputation table are not reached at all under the constraint of a special truncated differential trail.

**Proposition 2 (Efficient Tabulation Technique, [7]).** *If a message of $\delta$-set belongs to a pair conforming to the 4-round truncated differential trail outlined in Fig. 2, the values of multiset are only determined by 10 byte-parameters of intermediate state $\Delta z_1[0]||x_2[0,1,2,3]||\Delta x_5[0]||z_4[0,1,2,3]$ presented as gray cells in this figure.*

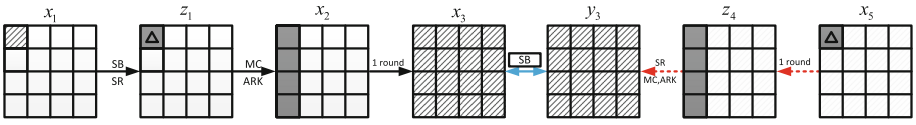


**Fig. 2.** The truncated differential trail of 4-round AES used in [5], the gray cells represent 10 byte-parameters, $\Delta$ represents difference.

The main idea of their works is that suppose one get a pair of messages conforming to this truncated differential trail, the differences $\Delta x_3$ and $\Delta y_3$ can be determined by these 10 byte-parameters. By Proposition 1, part of the 24 byte-parameters in the Demirci and Selçuk distinguisher, i.e. $x_3$, can be determined.

**Lin et al. Algorithm.** In [18], Lin et al. summarized former works of improved MITM distinguisher and gave a general model for this kind of distinguisher on SPN block ciphers. In their works, they define *$T$-$\delta$-set* which is a special *$\delta$-set* of $T$ active cells and *$S$-multiset* which is a *multiset* of $S$ cells to extend the definitions of $\delta$-set and *multiset*. With these definitions, they get **affected trail** which is a function connecting a *$T$-$\delta$-set* to an *$S$-multiset* with the minimal number of active cells[3] after $r$-round encryption. We call these active cells **guessed-cells** in this paper. As shown in Fig. 1, $x_1[0]$ is an 1-*$\delta$-set* and $\Delta x_5[0]$ is an 1-*multiset*. $(x_4[0,5,10,15]||x_3||x_2[0,1,2,3])$ is the affected trail connecting $x_1[0]$ to $x_5[0]$, i.e. the value of $\Delta x_5$ can be determined by $x_1[0]$ and these bytes. After that, they introduce a general algorithm to search for the best affected trail from a *$T$-$\delta$-set*

[3] These active cells are before the S-box layer, so we need to guess their values to get the *$S$-multiset*.

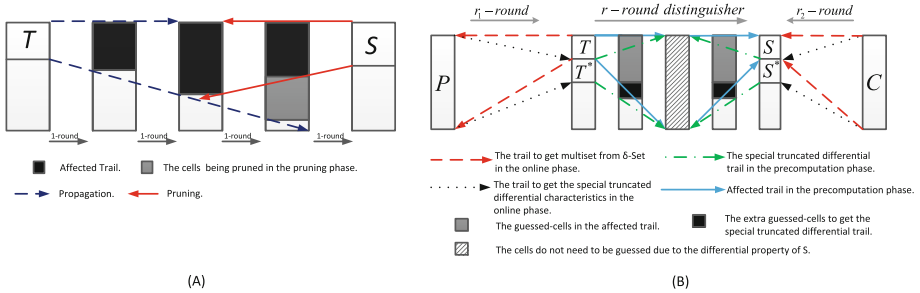


**Fig. 3.** (A) 4-round example of propagation-then-pruning algorithm; (B) General scheme of the improved meet-in-the-middle attack with efficient tabulation technique.

to an *S-multiset*, and find that building a better affected trail is equivalent to a positive integer optimization problem. In this paper, $T$ represents a *T-δ-set* and $S$ represents an *S-multiset*. With these two definitions in mind, we can choose appropriate values of $|T|$ and $|S|$ according to the success probability [18].

The search algorithm is based on the propagation-then-pruning algorithm as shown in Fig. 3(A). Suppose we have one $(T, S)$ pair, the algorithm to build an affected trail $\mathcal{D}$ is as follows:

1. **Propagation.** In the forward direction, differences of $T$ can propagate from one round to the next. Active cells before the S-box layer need to be guessed, and then $S$ can be got from this trail.
2. **Pruning.** In this trail, some guessed-cells have nothing to do with the building of $S$. These cells are pruned.

Using this algorithm, we can get an affected trail $\mathcal{D}$ for $(T, S)$.

### 2.3 Attack Scheme

In this subsection, we present our new unified view of the improved meet-in-the-middle attack, where $R$ rounds of block cipher can be split into three consecutive parts: $r_1$, $r$, and $r_2$.

The general attack scheme is shown in Fig. 3(B), here $T$ represents a *T-δ-set* and $S$ represents an *S-multiset*, $T^*$ represents input difference of the truncated differential trail and $S^*$ represents the output difference. For SPN block ciphers, $(T, S)$ and $(T^*, S^*)$ are almost the same. But for Feistel block ciphers, they are always different. The reason will be explained in Subsect. 2.4.

The general attack scheme uses two successive phases:

**Precomputation phase**

1. In the precomputation phase, we build an affected trail from $T$ to $S$ by guessing some cells.
2. Use efficient tabulation technique to build a special truncated differential trail from $T^*$ to one middle round, and then build a truncated differential trail from $S^*$ to another middle round in the reverse direction. This step needs to guess some more cells. Using Proposition 1, we can prune some guessed-cells made in step 1.
3. Build a lookup table $L$ containing all the possible sequences constructed from $T$ such that one message verifies the truncated differential trail.

**Online phase**

1. In the online phase, we need to identify a $T$-$\delta$-set containing a message $m$ verifying the desired property. This is done by using a large number of plaintexts and ciphertexts, and expecting that for each key candidate, there is one pair of plaintexts satisfying the truncated differential trail from $P \rightarrow T^*$ and $C \rightarrow S^*$. $m$ is one member of this plaintext pair. Then use $m$ to build a $T$-$\delta$-set.
2. Finally, we partially decrypt the associated $T$-$\delta$-set through the last $r_2$ rounds and check whether it belongs to $L$.

### 2.4 Feistel Ciphers with Efficient Tabulation Technique

In this paper, we focus on the application of efficient tabulation technique on word-oriented BFN and GFN. The round function $F$ is made up of 3 layers: key addition layer, S-box layer and linear transformation layer. This is true for most BFN and GFN ciphers.

The advantage of improved meet-in-the-middle attack with efficient tabulation technique on Feistel ciphers is that it can use Proposition 1 in $\lfloor \frac{b}{n \times o} \rfloor$ rounds, i.e. less cells are guessed in these rounds. We take CLEFIA [24] as an example. CLEFIA is a 4-branch type-2 GFN cipher with $b = 16$, $n = 2$ and $o = 4$.

As shown in Fig. 4(A), $\Delta y_i[0] = \Delta x_i[0]$, $\Delta z_i[0] = M_0^{-1}(\Delta x_i[1] \oplus \Delta x_{i+2}[3])$. If $\Delta x_i$ and $\Delta x_{i+2}$ are known, using Proposition 1, the values of active bytes in $y_i[0]$ can be got. Using the same method, if $\Delta x_i$ and $\Delta x_{i+2}$ are known, the values of active bytes in $y_i[2]$, $y_{i+1}[0]$ and $y_{i+1}[2]$ can be got as well. So if the special truncated differential trail is got, some cells need not to be guessed.

Although affected trail and truncated differential trail are almost the same for SPN ciphers, they differ a lot from each other for BFN and GFN ciphers. We also take CLEFIA as an example. As shown in Fig. 4(B), the guessed-cells of affected trail and truncated differential trail are totally different in the backward direction. For affected trail, if we want to get $\Delta x_{i+2}[1][0]$ from $\Delta x_i$, $\Delta x_{i+2}[1][0] = \Delta x_{i+1}[1][0] = \Delta w_i[2][0] \oplus \Delta x_i[3][0]$, only $y_i[2]$ need to be guessed. For truncated differential trail, if $\Delta x_{i+2}[1][0]$ is active and we know its value, by guessing $y_{i+1}[2][0]$, $\Delta x_{i+1}[2][0]$ and $\Delta x_{i+1}[3]$ can be got. By guessing $y_i[0]$, $\Delta x_i$ can be got. So $y_{i+1}[2][0]$ and $y_i[0]$ need to be guessed in addition. But for another

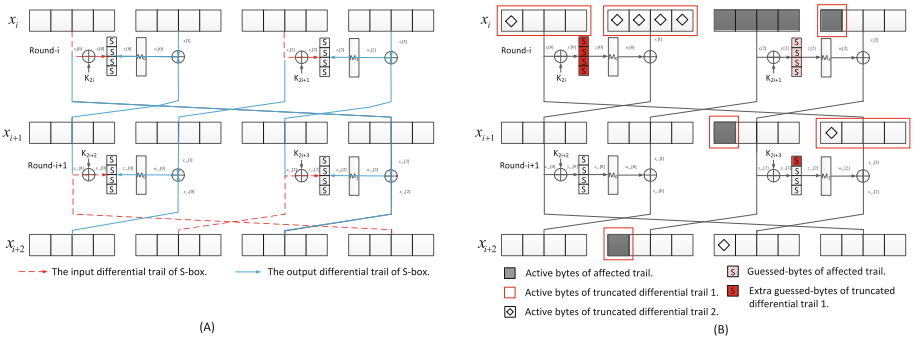


**Fig. 4.** (A) The truncated differential trail using Proposition 1; (B) The stagger of an affected trail and a truncated differential trail (Color figure online).

truncated differential trail, if $\Delta x_{i+2}[2][0]$ is active, by guessing $y_i[0][0]$, $\Delta x_i$ can be got. So only one additional cell need to be guessed.

In order to apply this technique to Feistel schemes, first of all, we should get an affected trail; and then we can use a truncated differential trail to restrain values of guessed-cells in the middle $\lfloor \frac{b}{n \times o} \rfloor$ rounds by guessing some extra cells. These two trails may differ a lot from each other, so we need to find a method to minimize the total number of guessed-cells.

To solve this problem, we present an automatic search algorithm in Sect. 3 to search for an optimal improved meet-in-the-middle distinguisher with efficient tabulation technique on BFN and GFN ciphers.

## 3 Automatic Search Algorithm Using Efficient Tabulation Technique

In this section, we present a practical algorithm to derive an optimal improved meet-in-the-middle distinguisher on Feistel schemes with efficient tabulation technique, which combines the precomputation phase of [11] and the search procedure of [22].

Suppose we get the affected trail [18] gives, our algorithm works by recursion and consists of 2 phases: table construction phase and searching phase.

### 3.1 Table Construction Phase

As shown in [11], the table construction phase builds a 2-equipartite directed acyclic graph $G$ which contains all the possible one-round transitions. This graph can be built and stored efficiently by observing its inner structure: the block cipher internal state output depends only on the block cipher internal state input. Unlike [11], since we don't consider the key schedule, the graph is small and can be stored in truncated differential characteristic[4]. A toy example of $G$ is shown in Fig. 5.

[4] The memory cost of CLEFIA is less than 20 MB.

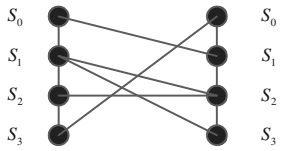


**Fig. 5.** Example of graph product to build $G$, with 4 possible internal states $s_0$, $s_1$, $s_2$ and $s_3$. There is an edge from $s_i$ to $s_j$ if and only if $s_i \rightarrow s_j$ after one round encryption.

A 2-equipartite directed acyclic graph $G^{-1}$ that contains all the possible one-round transitions in the backward direction should also be built as well.

### 3.2 Searching Phase

As the algorithm in [22], our searching phase finds an optimal $n$-round improved meet-in-the-middle distinguisher. The algorithm works by recursion and can be seen as a tree traversal in a depth-first manner, where the tree represents all the possible truncated differential trail in the cipher layered by round. The nodes represent the truncated differential characteristics and the edges the possible transitions between them, and are labeled by their numbers of guessed-cells and transition probabilities. One truncated differential trail is a path in this tree, and its weight equals the product of all traversed edges. We are looking for the path with the minimum number of guessed-cells in this tree under certain transition probability. The knowledge of the previous best truncated differential trail allows pruning during the procedure. Since we only consider truncated differential characteristic and the pruning is very efficient, the running time will not be too long.

#### 3.2.1 Trail Probability

Almost all the truncated differential trails used in the distinguishers on SPN ciphers are with probability 1. In our algorithm, we consider the truncated differential trail with probability less than 1, i.e. operations such as XOR can output inactive cells with certain probability. Suppose there is one less active cell in the backward direction (with probability $2^{-c}$), it may cause less extra cells to be guessed. Getting a trail with probability $2^{-c}$ means that the online phase should be repeated $2^c$ times. So our algorithm require an "initial value" for the minimum trail probability, which is presented as $\overline{P}$. This value can be determined by analyzing the online phase.

#### 3.2.2 Comparing Trails

Next we introduce a (quasi-)order relation for two truncated differential trail $\mathcal{T}_1$ and $\mathcal{T}_2$ as follows:

**Definition 3** ($\succ$). *$\mathcal{T}_1$ is better than $\mathcal{T}_2$ if and only if it has less guessed-cells or its probability is higher with the same number of guessed-cells, i.e.*

$$\mathcal{T}_1 \succ \mathcal{T}_2 \Leftrightarrow \begin{cases} \mathcal{G}(\mathcal{T}_1) < \mathcal{G}(\mathcal{T}_2) \\ or \\ \mathcal{G}(\mathcal{T}_1) = \mathcal{G}(\mathcal{T}_2) \ and \ \mathcal{P}(\mathcal{T}_1) > \mathcal{P}(\mathcal{T}_2) \end{cases} \tag{2}$$

Also $\mathcal{T}_1 \equiv \mathcal{T}_2 \Leftrightarrow \mathcal{G}(\mathcal{T}_1) = \mathcal{G}(\mathcal{T}_2)$ and $\mathcal{P}(\mathcal{T}_1) = \mathcal{P}(\mathcal{T}_2)$[5].

### 3.2.3 Ending Condition

Given key length $k$, probability lower bound $\overline{P}$ and the best trail $\mathcal{T}_{best}$ we found so far, we define ending condition $\mathcal{E}$ as follows:

$$\mathcal{T} \in \mathcal{E} \Leftrightarrow \begin{cases} \mathcal{P}(\mathcal{T}) \leq \overline{P} \\ \text{or } \mathcal{G}(\mathcal{T}) \geq k \\ \text{or } \mathcal{T}_{best} \succ \mathcal{T} \end{cases} \tag{3}$$

If a trail belongs to $\mathcal{E}$, we should stop the search procedure and try another trail.

### 3.2.4 Finding an Optimal Trail

Although we could test all the truncated differential trail under the probability lower bound $\overline{P}$ to find the best one, we have a more efficient way using the greedy algorithm. Since the affected trail is unique for each $(S, T)$ pair, we can find out $\lfloor \frac{b}{n \times o} \rfloor$ successive rounds which need to guess more cells than others, i.e. there are more active cells in these rounds. This means that more cells need not to be guessed using Proposition 1. If the number of these successive rounds is more than one, the beginning of these rounds can be represented as a set, called $SR$-set.

With $SR$-set in mind, the search procedure can be divided into 2 parts: one starts at the first round in the forward direction, the other starts at the last round in the backward direction. This algorithm will divide an $r$-round truncated differential trail search process into 2 parts: $r_1$ and $r_2$, where $r = r_1 + \lfloor \frac{b}{n \times o} \rfloor + r_2$.

At the beginning of this algorithm, we should decrease $\mathcal{G}(\mathcal{D})$ by the number of guessed-cells in the middle $\lfloor \frac{b}{n \times o} \rfloor$ rounds. We may meet the situation that not all the guessed-cells can be pruned in the middle $\lfloor \frac{b}{n \times o} \rfloor$ rounds, i.e. there are inactive cells in the truncated differential trail. However, since this also restrains the values of guessed-cells in the truncated differential trail, the total number of guessed-cells remains unchanged.

Although the first and last truncated differential characteristics in the trail can take any values, we put some limitations on them by some observations. Since there is little difference between an affected trail and a truncated differential trail in the first $r_1$-round, we fix the first truncated differential characteristic $T^*$ to $T$.

[5] $\mathcal{G}$: total number of guessed-cells including the affected trail. $\mathcal{P}$: trail probability.

And by the propagation of differences, we constrain values of the last truncated differential characteristics $S^*$ with $|S^*| \leq |S|$.

The framework of our algorithm for the first $r_1$ and the last $r_2$ rounds is now established by Algorithms 1 and 2 including essentially recursive calls.

The inputs of Algorithms 1 and 2 are affected trail $\mathcal{D}$, input truncated differential trail $\mathcal{T}$, best truncated differential trail $\mathcal{T}_{best}$, probability lower bound $\overline{P}$ and graph $G/G^{-1}$.

**Algorithm 1.** Search the first $r_1$-round

```
1: function Procedure Round_i^begin(D, T, T_best, P̄)
2:   Find all truncated values this trail can lead to in graph G
3:   Sort these truncated differential characteristics
4:   for all truncated differential characteristics do
5:     Add this characteristic to T
6:     if T ∉ E then
7:       if i < r_1 − 1 then
8:         Call Procedure Round_{i+1}^begin
9:       else
10:        for all truncated differences S* satisfying |S*| ≤ |S| do
11:          Call Procedure Round_0^end with S*
12:        end for
13:      end if
14:    end if
15:  end for
16: end function
```

The sorting algorithm of line 3 in Algorithms 1 and 2 is according to $\succ$.

Line 10 of Algorithm 2 means $s_{r_1} \xrightarrow{\lfloor \frac{b}{n \times o} \rfloor rounds} s_{r_2}$, where $s_{r_1}$ and $s_{r_2}$ are truncated differential characteristics of round-$r_1$ and round-$(r_1 + \lfloor \frac{b}{n \times o} \rfloor)$ in the trail, respectively. Take CLEFIA as an example, since the branch number of $M_0$ and $M_1$ is 5, if the total number of active cells before and after $M_0$ or $M_1$ is less than 5, we should stop the search procedure and try another trail.

Algorithm 3 presents the search algorithm for a $(T, S)$ pair.

We can loop through all possible$(T, S)$ pairs to find an optimal $r$-round distinguisher under $\overline{P}$, and then find $r_{max} = max\{r | \mathcal{P}(\mathcal{T}_{best}) > \overline{P}$ and $\mathcal{G}(\mathcal{T}_{best}) < k\}$.

## 4 Applications

In this section, we give our attacks on CLEFIA-192/256 and Camellia*-192/256.

### 4.1 Applications to CLEFIA-192/256

CLEFIA is a lightweight 128-bit block cipher designed by Shirai et al. in 2007 [24] and based on a 4-branch type-2 GFN. It is adopted as an international ISO/IEC 29192 standard in lightweight cryptography. We refer to [24] for a detailed description.

**Algorithm 2.** Search the last $r_2$-round

```
function Procedure Round_i^end(D, T, T_best, P̄)
  Find all truncated values this trail can lead to in graph G^{-1}
  Sort these truncated differential characteristics
  for all truncated differential characteristics do
    Add this characteristic to T
    if T ∉ E then
      if i < r_2 − 1 then
        Call Procedure Round_{i+1}^end
      else
        if Combining 2 parts of T leads to a trail and T ∉ E then
          T_best ← T
        end if
      end if
    end if
  end for
end function
```

**Algorithm 3.** Search for an optimal trail for $(T, S)$

```
function Procedure SerachingTrail(D, P̄, T, S, ⌊b/(n×o)⌋)
  Initial T_best with k and P̄
  Get the SR-set from D
  for all r_1 in SR-set do
    Decrease G(D) by the number of guessed-cells in these ⌊b/(n×o)⌋ rounds
    Call Round_0^begin with r_1 and T
  end for
return T_best
end function
```

#### 4.1.1 9/10-Round Distinguisher on CLEFIA

First, we use our search algorithm to find an optimal 10-round distinguisher on CLEFIA-256 and 9-round distinguisher on CLEFIA-192. They are shown in Figs. 6 and 7.

In the attack of CLEFIA, we apply an equivalent transformation to the 10-round and 9-round distinguishers, as shown in Figs. 6 and 7. Namely, the right linear transformations of round $i+8$ and $i+7$ are removed from these rounds, and these linear transformations are added to three different positions in order to obtain distinguishers that are computationally equivalent to the original one.

The 10-round distinguisher on CLEFIA-256 is based on the proposition below.

**Proposition 3.** *Considering to encrypt* $2^8$ *values of the (1-)δ-set after 10-round CLEFIA-256 starting from round-i, where* $x_i[1][0]$ *is the active byte, in the case of that a message of the δ-set belongs to a pair which conforms to the truncated differential trail outlined in Fig. 6, then the corresponding (1-)multiset of* $\Delta x_{i+10}[1][0]$ *only contains about* $2^{208}$ *values.*

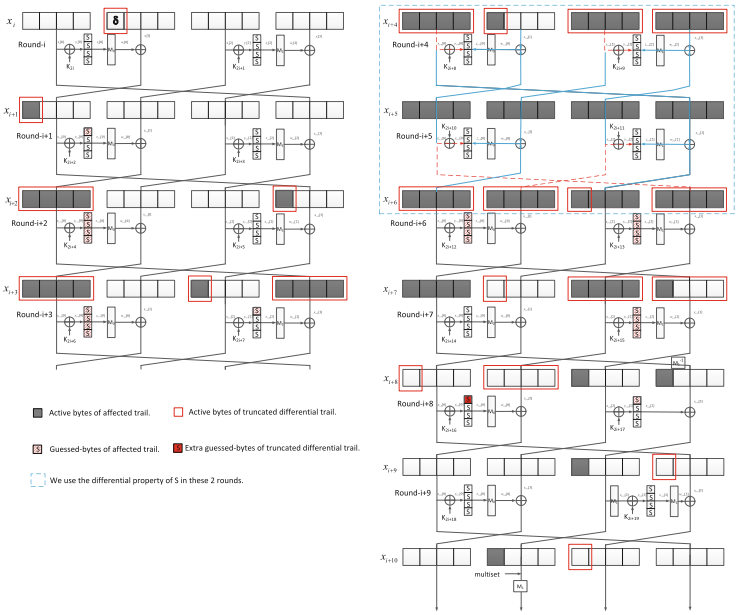


**Fig. 6.** 10-Round Distinguisher on CLEFIA-256

*Proof.* As shown in Fig. 6, we first consider the affected trail from $x_i[1][0]$ to $x_{i+10}[1][0]$. This affected trail is determined by 39 byte-parameters: $y_{i+1}[0][0]||y_{i+2}[0]||y_{i+3}[0]||y_{i+3}[2][0]||y_{i+4}[0]||y_{i+4}[2]||y_{i+5}[0]||y_{i+5}[2]||y_{i+6}[0]||y_{i+6}[2]||y_{i+7}[2]||y_{i+8}[2][0]$.

This can be easily seen from the figure.

Furthermore, if there exists a message of the (1-)$\delta$-set belongs to a pair which conforms the truncated differential trail as in Fig. 6, the 35 byte-parameters $y_{i+1}[0][0]||y_{i+2}[0]||y_{i+3}[0]||y_{i+3}[2][0]||y_{i+4}[0]||y_{i+4}[2]||y_{i+5}[0]||y_{i+5}[2]||y_{i+6}[0]||y_{i+6}[2][0]||y_{i+7}[2]$ is determined by 22 byte-parameters: $\Delta x_i[1][0]||y_{i+1}[0][0]||y_{i+2}[0]||y_{i+3}[0]||y_{i+3}[2][0]||y_{i+6}[0]||y_{i+6}[2][0]||y_{i+7}[2]||y_{i+8}[0][0]||\Delta x_{i+10}[2][0]$.

Using 11 byte-parameters $\Delta x_i[1][0]||y_{i+1}[0][0]||y_{i+2}[0]||y_{i+3}[0]||y_{i+3}[2][0]$, we can deduce $\Delta x_{i+4}$. In the backward direction, using $\Delta x_{i+10}[2][0]||y_{i+8}[0][0]||y_{i+7}[2]||y_{i+6}[0]||y_{i+6}[2][0]$,we can deduce $\Delta x_{i+6}$. By Proposition 1, this can deduce $y_{i+4}[0]$, $y_{i+4}[2]$, $y_{i+5}[0]$ and $y_{i+5}[2]$.

In conclusion, the corresponding *multiset* of byte $\Delta x_{i+10}[1][0]$ only contains about $2^{208}$ values with the truncated differential trail. □

The 9-round distinguisher on CLEFIA-192 is based on the proposition below, and we will meet the situation Sect. 3.2.4 gives.

**Proposition 4.** *Considering to encrypt* $2^8$ *values of the (1-)*$\delta$*-set after 9-round CLEFIA-192 starting from round-i, where* $x_i[1][0]$ *is the active byte, in the case*

*of that a message of the δ-set belongs to a pair which conforms to the truncated differential trail outlined in Fig. 7, and then the corresponding multiset of* $\Delta x_{i+9}[1][0]$ *only contains about* $2^{144}$ *values.*

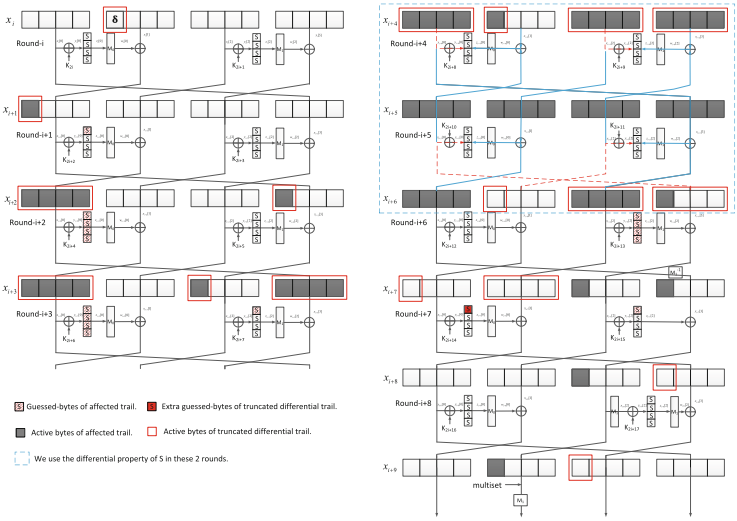


**Fig. 7.** 9-Round Distinguisher on CLEFIA-192

The proof of this proposition is the same as before and is shown in Fig. 7.

The online phase of 16/14-round attack on CLEFIA-256/192 is shown in Appendix A. For CLEFIA-256, we give a 16-round key recovery attack with data complexity of $2^{113}$ chosen-plaintexts, time complexity of $2^{219}$ encryptions and memory complexity of $2^{210}$ 128-bit blocks. For CLEFIA-192, we give a 14-round key recovery attack with data complexity of $2^{113}$ chosen-plaintexts, time complexity of $2^{155}$ encryptions and memory complexity of $2^{146}$ 128-bit blocks.

### 4.2 Applications to Camellia*-192/256

Camellia is a 128-bit block cipher designed by Aoki et al. in 2000 [1]. It is a Feistel-like construction where two key-dependent layer $FL$ and $FL^{-1}$ are applied every 6 rounds to each branch. In this paper, we analyze Camellia without $FL$ and $FL^{-1}$, and call it Camellia* here. We refer to [1] for the detailed description of Camellia.

#### 4.2.1 Attack on Camellia*-192

The 8-round distinguisher of Camellia*-192 with 16 guessed-bytes is shown in Fig. 8. We apply an equivalent transformation to the distinguisher. Namely, the linear transformation of round $i + 6$ is removed, and this linear transformation

is added to three different positions in order to obtain distinguishers that are computationally equivalent to the original one. This idea is inspired by [9,15].

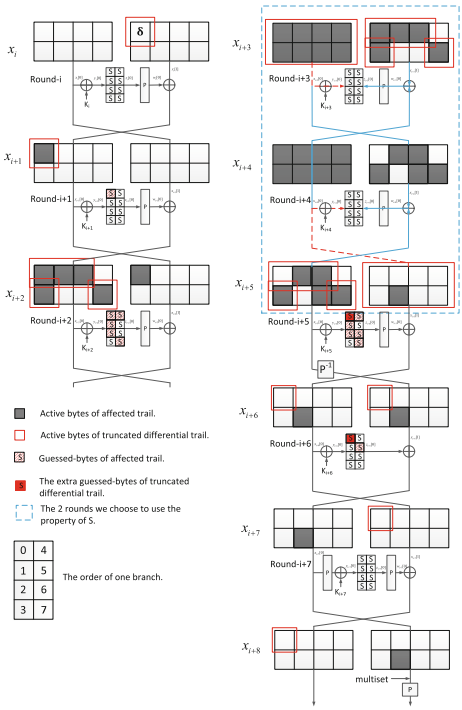


**Fig. 8.** 8-Round Distinguisher on Camellia*-192

By guessing $y_{i+1}[0][0]$, $y_{i+2}[0][0,1,2,4,7]$, $y_{i+3}[0]$, $y_{i+4}[0]$, $y_{i+5}[0][1,2,4,6,7]$ and $y_{i+6}[0][5]$, we can get *multiset* of $\Delta x_{i+8}[1][5]$. If there is a truncated differential trail from $x_i[1][0]$ to $x_{i+8}[0][0]$ as the figure shows, then $y_{i+1}[0][0]$, $y_{i+2}[0][0,1,2,4,7]$, $y_{i+3}[0]$, $y_{i+4}[0]$ and $y_{i+5}[0][1,2,4,7]$ can be determined by $\Delta x_i[1][0]$, $y_{i+1}[0][0]$, $y_{i+2}[0][0,1,2,4,7]$, $\Delta x_{i+8}[0][0]$, $y_{i+6}[0][0]$, $y_{i+5}[0][0,1,2,4,7]$.

In a word, the *multiset* of byte $\Delta x_{i+8}[1][5]$ can be determined by 16 byte-parameters.

The online phase of this attack is the same as the 12-round attack on Camellia-192 in [15]. So we can extend 2 rounds on the top and 3 rounds on the bottom to build a 13-round attack on Camellia*-192 with time complexity of $2^{180}$ encryptions, data complexity of $2^{113}$ chosen plaintexts and memory complexity of $2^{130}$ 128-bit blocks.

#### 4.2.2 Attack on Camellia*-256

For the distinguisher of Camellia*-256, we simply extend one round after round-$(i+3)$ by guessing the whole 8 byte-parameters after the key addition layer. Then we can get a 10-round distinguisher on Camellia*-256.

In the online phase, we simply extend one round after the distinguisher by guessing all the 8 byte-parameters after the key addition layer. Then we can build a 15-round attack on Camellia*-256 with time complexity of $2^{244}$ encryptions, data complexity of $2^{113}$ chosen plaintexts and memory complexity of $2^{194}$ 128-bit blocks.

## 5 Conclusion and Future Work

This paper has shown the improved meet-in-the-middle distinguisher with efficient tabulation technique on BFN and GFN block ciphers. We discuss the problem why this technique is rarely used on the attacks of BFN and GFN ciphers, and then describe an efficient and generic algorithm to search for an optimal improved meet-in-the-middle distinguisher with efficient tabulation technique on them. It is based on recursive algorithm and greedy algorithm.

To demonstrate the usefulness and versatility of our approaches, we show attacks on CLEFIA and Camellia*. Among them, we would like to stress that the presented attacks on 14/16-round CLEFIA-192/256 are the best attacks. Since our approach is generic, it is expected to be applied to other BFN and GFN ciphers. We believe that our results are useful not only for a deeper understanding of the security of Feistel schemes, but also for designing a secure block cipher.

The research community still has a lot to learn on the way to build better attacks and there are many future works possible: the algorithm combining the precomputation phase and online phase together, and the link between this kind of attack with other kinds of attacks, such as truncated differential attack and impossible differential attack.

**Acknowledgements.** We would like to thank the anonymous reviewers for providing valuable comments. The research presented in this paper is supported by the National Basic Research Program of China (No. 2013CB338002) and National Natural Science Foundation of China (No. 61272476, No.61232009 and No. 61202420).

## References

1. Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., Tokita, T.: *Camellia*: a 128-bit block cipher suitable for multiple platforms - design and analysis. In: Stinson, D.R., Tavares, S. (eds.) SAC 2000. LNCS, vol. 2012, pp. 39–56. Springer, Heidelberg (2001)
2. Bogdanov, A., Geng, H., Wang, M., Wen, L., Collard, B.: Zero-correlation linear cryptanalysis with FFT and improved attacks on ISO standards Camellia and CLEFIA. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 306–323. Springer, Heidelberg (2014)

3. Chen, J., Jia, K., Yu, H., Wang, X.: New impossible differential attacks of reduced-round Camellia-192 and Camellia-256. In: Parampalli, U., Hawkes, P. (eds.) ACISP 2011. LNCS, vol. 6812, pp. 16–33. Springer, Heidelberg (2011)
4. Daemen, J., Rijmen, V.: The Design of Rijndael: AES-the Advanced Encryption Standard. Springer, Heidelberg (2002)
5. Demirci, H., Selçuk, A.A.: A meet-in-the-middle attack on 8-round AES. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 116–126. Springer, Heidelberg (2008)
6. Demirci, H., Taşkın, I., Çoban, M., Baysal, A.: Improved meet-in-the-middle attacks on AES. In: Roy, B., Sendrier, N. (eds.) INDOCRYPT 2009. LNCS, vol. 5922, pp. 144–156. Springer, Heidelberg (2009)
7. Derbez, P., Fouque, P.-A., Jean, J.: Improved key recovery attacks on reduced-round AES in the single-key setting. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 371–387. Springer, Heidelberg (2013)
8. Diffie, W., Hellman, M.E.: Special feature exhaustive cryptanalysis of the NBS data encryption standard. Computer **10**(6), 74–84 (1977)
9. Dong, X., Li, L., Jia, K., Wang, X.: Improved attacks on reduced-round Camellia-128/192/256. In: Nyberg, K. (ed.) CT-RSA 2015. LNCS, vol. 9048, pp. 59–83. Springer, Heidelberg (2015)
10. Dunkelman, O., Keller, N., Shamir, A.: Improved single-key attacks on 8-round AES-192 and AES-256. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 158–176. Springer, Heidelberg (2010)
11. Fouque, P.-A., Jean, J., Peyrin, T.: Structural evaluation of AES and Chosen-Key Distinguisher of 9-Round AES-128. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 183–203. Springer, Heidelberg (2013)
12. Gilbert, H., Minier, M.: A collisions attack on the 7-rounds Rijndael. In: AES Candidate Conference, Citeseer (2000)
13. Guo, J., Jean, J., Nikolić, I., Sasaki, Y.: Meet-in-the-middle attacks on generic Feistel constructions. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 458–477. Springer, Heidelberg (2014)
14. Isobe, T., Shibutani, K.: All subkeys recovery attack on block ciphers: extending meet-in-the-middle approach. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 202–221. Springer, Heidelberg (2013)
15. Li, L., Jia, K.: Improved meet-in-the-middle attacks on reduced-round Camellia-192/256. IACR Cryptology ePrint Archive 2014, 292 (2014)
16. Li, L., Jia, K., Wang, X.: Improved meet-in-the-middle attacks on AES-192 and PRINCE. IACR Cryptology ePrint Archive 2013, 573 (2013)
17. Li, L., Jia, K., Wang, X.: Improved single-key attacks on 9-round AES-192/256. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 127–146. Springer, Heidelberg (2015)
18. Lin, L., Wu, W., Wang, Y., Zhang, L.: General model of the single-key meet-in-the-middle distinguisher on the word-oriented block cipher. In: Lee, H.-S., Han, D.-G. (eds.) ICISC 2013. LNCS, vol. 8565, pp. 203–223. Springer, Heidelberg (2014)
19. Jiqiang, L.: Cryptanalysis of block ciphers. Ph.D. thesis. University of London, UK (2008)
20. Jiqiang, L., Wei, Y., Fouque, P.-A., Kim, J.: Cryptanalysis of reduced versions of the Camellia block cipher. IET Inf. Secur. **6**(3), 228–238 (2012)
21. Jiqiang, L., Wei, Y., Kim, J., Pasalic, E.: The higher-order meet-in-the-middle attack and its application to the Camellia block cipher. Theoret. Comput. Sci. **527**, 102–122 (2014)

22. Matsui, M.: On correlation between the order of S-Boxes and the strength of DES. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 366–375. Springer, Heidelberg (1995)
23. Nyberg, K.: Generalized Feistel networks. In: Kim, K., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 91–104. Springer, Heidelberg (1996)
24. 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)
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)

## Appendix A: 16/14-Round Attack on CLEIFA-256/192

Based on the 10-round distinguisher, we extend 3 rounds on the top and 3 rounds on the bottom to present a 16-round improved meet-in-the-middle attack on CLEFIA-256, and based on the 9-round distinguisher, we extend 3 rounds on the top and 2 rounds on the bottom to present a 14-round improved meet-in-the-middle attack on CLEFIA-192. The procedure of the attack on CLEFIA-256 is shown in Fig. 9.

The following proposition is important in finding the special truncated differential trail.

**Proposition 5.** *If* $M_i(\Delta s_0) = (a_0, 0, 0, 0)$ *and* $M_i(\Delta s_1) = (a_1, 0, 0, 0)$*, then* $M_i(\Delta s_0 \oplus \Delta s_1) = (a_2, 0, 0, 0)$*, where* $a_0$*,* $a_1$*,* $a_2$ *are any bytes*[6].

This is because of the linearity of $M_i$. The set of $2^8$ differences that results in $(a, 0, 0, 0)$ after the linear transformation layer is called $\alpha$-set, and it is marked by red triangle in Fig. 9.

The detailed attack is shown below:

1. **Precomputation phase.** In the precomputation phase of the attack, we build the lookup table $L$ that contains the multiset of size $2^{208}$ for difference $\Delta x_{13}[1][0]$ by following the method of Proposition 4.
2. **Online Phase.**
   (a) **Detecting the Right Pair:**
      i. We prepare a structure of $2^{80}$ plaintexts where $\Delta P[0][0]$, $\Delta P[2]$ and $\Delta P[3]$ take all the $2^{72}$ values and $\Delta P[1]$ takes all the values of the $\alpha$-set. Hence, we can generate $2^{80} \times (2^{80} - 1)/2 \approx 2^{159}$ pairs satisfying the plaintext differences. Choose $2^{33}$ structures and get the corresponding ciphertexts. Among the $2^{159+33} = 2^{192}$ corresponding ciphertext pairs, we expect $2^{192} \times 2^{-48} = 2^{144}$ pairs to verify the truncated difference pattern where $\Delta C[0][0]$, $\Delta C[2]$, $\Delta C[3]$ have differences, and $\Delta C[1]$ has difference in $\alpha$-set. Store the $2^{144}$ remaining pairs in a hash table. This step require $2^{113}$ plaintext and ciphertext pairs.

[6] $M_i$ denotes the linear transformation layer.

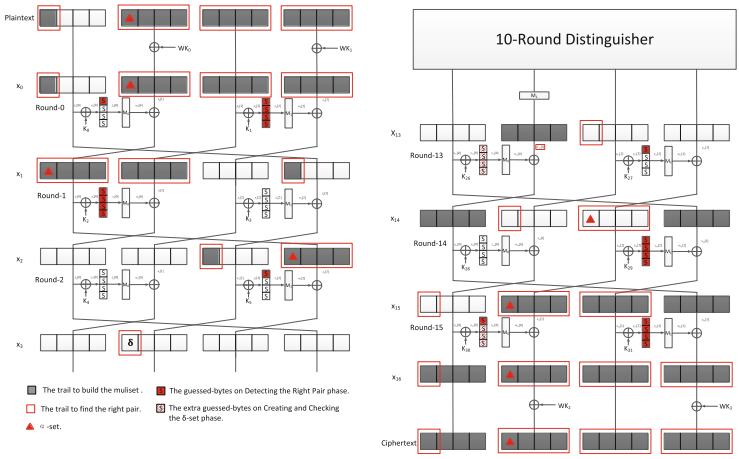


**Fig. 9.** 16-Round Attack on CLEFIA-256

ii. Guess the values of $K_0[0]$, $K_1$, $y_1[0]$ and $y_2[2][0]$, using the guessed values to encrypt the remaining pairs to $x_3$. We choose the pairs that have difference only in byte $x_3[1][0]$, there are $2^{144-72} = 2^{72}$ pairs left.

iii. Guess the values of $K_{30}[0]$, $K_{31}$, $y_{14}[2]$ and $y_{13}[2][0]$, using the guessed values to decrypt the remaining pairs to $x_{13}$. We choose the pairs that have differences only in byte $x_{13}[2][0]$, there are $2^{72-72} = 1$ pair left.

(b) **Creating and Checking the Multiset:**

i. For each guess of the 20 bytes made in Phase (a), decrypt the $2^8$ possible differences in $\Delta x_3$ to $\Delta x_0$ using the knowledge of $K_0[0]$, $K_1$, $y_1[0]$ and $y_2[2][0]$. Then XOR it with one plaintext $P_0$ of the pair.

ii. Using $P_0$ as the standard plaintext, denote the other 255 plaintexts as $P_1$ to $P_{255}$, and the corresponding ciphertexts as $C_0$ to $C_{255}$. Partially decrypt the ciphertexts to $x_{13}$ (here $x'_{13}[1] = M_1(x_{13}[1])$), and then get the *multiset* of $\Delta x_{13}[1][0]$ by guessing $K_{30}[1,2,3]$, $y_{13}[0]$, and using the knowledge of $K_{30}[0]$, $K_{31}$, $y_{14}[2]$.

iii. Checking whether the *multiset* exists in $L$. If not, discard the key guess. The probability for a wrong guess to pass this test is smaller than $2^{208}2^{-506.17} = 2^{-306.17}$.

(c) **Searching the Rest of Key:** For each remaining key guess, find the remaining key bytes by exhaustive search.

**Complexity.** The look up table of the $2^{208}$ possible values requires about $2^{210}$ 128-bit blocks to be stored [7]. To construct the table, we have to perform $2^{208}$ partial encryptions on 256 messages, which we estimate to be equivalent to $2^{208+8-4} = 2^{212}$ encryptions.

We take another look at online phase to evaluate the complexity. For each of the $2^{144}$ remaining pairs after the first part of Phase (a), since $\Delta y_0[2] = \Delta P[2]$

and $\Delta z_0[2] = M_1^{-1}(\Delta P[3] \oplus \Delta x_1[2]) = M_1^{-1}(\Delta P[3])$, by Proposition 1, we can get $K_1$. By guessing $y_2[2][0]$, we can deduce $\Delta x_2[3]$, and then get $y_1[0]$ and $K_0[0]$ for the same reason as before. Since $\Delta y_{15}[2] = \Delta C[2]$ and $\Delta z_{15}[2] = M_1^{-1}(\Delta C[3])$, by Proposition 1, we can get $K_{31}$. By guessing $y_{14}[2][0]$, we can deduce $\Delta x_{14}[2]$, and then get $y_{14}[2]$ and $K_{30}[0]$ for the same reason as before. Therefore, there are $2^{16}$ of 20 byte-parameters for one pair. After that, we should guess $K_{30}[1, 2, 3]$ and $y_{13}[0]$ in addition. In conclusion, for each of the $2^{144}$ found pairs, we perform $2^{72}$ partial encryptions/decryptions of a $\delta$-set. We evaluate the time complexity of this part to $2^{144+72+8-5} = 2^{219}$ encryptions.

Hence, the data complexity is $2^{113}$ chosen-plaintexts, the time complexity is $2^{219}$ encryptions and the memory complexity is $2^{210}$ 128-bit blocks.

The 14-round attack on CLEFIA-192 is almost the same as the former attack. The data complexity is $2^{113}$ chosen-plaintexts, the time complexity is $2^{155}$ encryptions and the memory complexity is $2^{146}$ 128-bit blocks.