Improved Impossible Differential Cryptanalysis of Reduced-Round Camellia

Wenling Wu, Lei Zhang, and Wentao Zhang

State Key Laboratory of Information Security, Institute of Software,
Chinese Academy of Sciences, Beijing 100190, P.R. China
{wwl,zhanglei1015,zhangwt}@is.iscas.ac.cn

Abstract. The block cipher Camellia has now been adopted as an international standard by ISO/IEC, and it has also been selected to be Japanese CRYPTREC e-government recommended cipher and in the NESSIE block cipher portfolio. Most recently, Wu et al constructed some 8-round impossible differentials of Camellia, and presented an attack on 12-round Camellia-192/256 in [5]. Later in [6], Lu et al improved the above attack by using the same 8-round impossible differential and some new observations on the diffusion transformation of Camellia. Considering that all these previously known impossible differential attacks on Camellia have not taken the key scheduling algorithm into account, in this paper we exploit the relations between the round subkeys of Camellia, together with some novel techniques in the key recovery process to improve the impossible differential attack on Camellia up to 12-round Camellia-128 and 16-round Camellia-256. The data complexities of the two attacks are 2^{65} and 2^{89} respectively, and the time complexities of the two attacks are less than $2^{111.5}$ and $2^{222.1}$ respectively. The presented results are better than any previously published cryptanalytic results on Camellia without the FL/FL^{-1} functions and whitening layers.

Keywords: Block cipher, Camellia, Impossible differential, Cryptanalysis, Round subkey.

1 Introduction

The block cipher Camellia [1], with the same interface specification as the Advanced Encryption Standard(AES), supports 128-bit block size and 128-, 192- and 256-bit key sizes, which can usually be denoted as Camellia-128, Camellia-192 and Camellia-256 respectively. Camellia was jointly developed by NTT and Mitsubishi Electric Corporation, and it was first published at SAC 2000. Then it was submitted to some cryptographic evaluation projects such as the European NESSIE Project and the Japanese CRYPTREC Evaluation, and Camellia was selected to be CRYPTREC e-government recommended cipher in 2002 and in the NESSIE block cipher portfolio in 2003. Furthermore, it was adopted as a new international standard for 128-bit block cipher by ISO/IEC in 2005. As Camellia has become one of the most worldwide used block ciphers, in the last few years

R. Avanzi, L. Keliher, and F. Sica (Eds.): SAC 2008, LNCS 5381, pp. 442–456, 2009.

cryptanalysts had evaluated the security of Camellia against various cryptanalytic techniques, including truncated differential cryptanalysis [2,3], higher order differential cryptanalysis [4], impossible differential cryptanalysis [3,5,6], Square attack/Integral attack [7-11], collision attack [12,13], linear cryptanalysis [14,15] and so on.

Impossible differential cryptanalysis [16] was first proposed by Biham et al in 1999, and was applied to the Skipjack cipher reduced from 32 to 31 rounds. Unlike traditional differential cryptanalysis which exploits differentials with the highest possible probability, impossible differential cryptanalysis uses differentials which hold with probability 0, which can also be called impossible differentials. An impossible differential can usually be built in a miss-in-the-middle manner. Recently, impossible differential cryptanalysis had received worldwide attention, and its application to the security analysis of AES and CLEFIA both got very good results [17-23].

The initial analysis of Camellia against impossible differential cryptanalysis was given by M.Sugita et al [3] in 2001, they constructed a nontrivial 7-round impossible differential for Camellia. In 2007, by exploiting some properties of the linear diffusion function, Wu et al [5] presented some 8-round impossible differentials for Camellia, and based on it they mounted an impossible differential attack on 12-round Camellia-192/256. Then in [6], Lu et al exploited the same 8-round impossible differential together with the early abort technique and improved the impossible differential cryptanalysis of Camellia. However, all of these impossible differential attacks on Camellia have not taken the key scheduling algorithm into account. Thus in this paper, we first present some observations of the relations between round subkeys, and then by taking advantage of these relations and some novel techniques in the key recovery process, we improve the impossible differential attack on Camellia up to 12-round Camellia-128 and 16-round Camellia-256. As far as we know, these are the best published cryptanalytic results on Camellia without the FL/FL^{-1} functions and whitening layers, and we summarize our results together with the previously known results on Camellia in Table 1.

The cryptanalytic results of [6] in Table 1 come from an early version, not the published version, so we mark them with "†". This is because there are some mistakes in the published version. In Step 3 of the 14-round attack for camellia-256 in [6], the authors wrote: "Finally, for every remaining pair of plaintexts we can get the first bytes of their intermediate values just after Round 2." Byte 1,3,4,6,7,8 of K_2 should be known for calculating the first byte just after round 2. However, only byte 1,2,3,5,8 are guessed in the attack, whereas byte 4,6,7 are unknown. There are similar mistakes in the other attacks in [6].

This paper is organized as follows. In Section 2, we give a brief description of Camellia. In Section 3, we describe the 8-round impossible differential and some properties of Camellia which are used in our attacks. Then in Section 4, we present our impossible differential attacks on 12-round Camellia-128 and 16-round Camellia-256 respectively. Finally, in Section 5 we summarize this paper.

Table 1. Summary of known cryptanalytic results on Camellia

Cipher	# of Rounds	FL/FL^{-1}	Attack Type	Data Complexity	Time Complexity	Source
Camellia-128	8	×	Truncated DC	$2^{83.6}$	$2^{55.6}$	[2]
	9	×	Collision Attack	$2^{113.6}$	2^{121}	[12]
	9	×	Square Attack	2^{66}	$2^{84.8}$	[11]
	11	×	Impossible DC	2^{120}	$2^{83.4}$	[6]†
	12	×	Impossible DC	2^{65}	$2^{111.5}$	Sec. 4.1
Camellia-192/256	11	√	Higher Order DC	2^{93}	2^{256}	[4]
	12	×	Linear Attack	2^{119}	2^{247}	[14]
	12	×	Impossible DC	2^{120}	2^{181}	[5]
	12	×	Square Attack	2^{66}	$2^{249.6}$	[11]
	13	×	Impossible DC	2^{120}	$2^{211.7}$	[6]†
	16	×	Impossible DC	2^{89}	$2^{222.1}$	Sec. 4.2

2 Description of Camellia

The overall structure of Camellia is a variant of Feistel structure, with the FL/FL^{-1} functions inserted at every 6 rounds. Before the first round and after the last round, there are pre- and post- whitening layers which employ bitwise exclusive-OR operations with 128-bit whitening subkeys respectively. In this paper, we will only consider Camellia without FL/FL^{-1} functions and whitening layers, namely the simplified variant of Camellia.

Let L_{r-1} and R_{r-1} be the left and the right halves of the r-th round input, and K_r be the r-th round subkey respectively. Then the r-th round of Camellia can be expressed as follows.

$$\begin{aligned} L_r &= R_{r-1} \oplus F(L_{r-1} \oplus K_r), \\ R_r &= L_{r-1}. \end{aligned}$$

Here the round function of Camellia is $F = P \circ S$, and the transformations S and P are defined as follows.

$$\begin{aligned} S : (F_2^8)^8 &\longrightarrow (F_2^8)^8 \\ & x_1 \,|\, x_2 \,|\, x_3 \,|\, x_4 \,|\, x_5 \,|\, x_6 \,|\, x_7 \,|\, x_8 \longrightarrow y_1 \,|\, y_2 \,|\, y_3 \,|\, y_4 \,|\, y_5 \,|\, y_6 \,|\, y_7 \,|\, y_8 \\ & y_1 = s_1(x_1), \quad y_2 = s_2(x_2), \quad y_3 = s_3(x_3), \quad y_4 = s_4(x_4), \\ & y_5 = s_2(x_5), \quad y_6 = s_3(x_6), \quad y_7 = s_4(x_7), \quad y_8 = s_1(x_8). \end{aligned}$$

where s_1, s_2, s_3 and s_4 are four 8×8 S-boxes.

$$\begin{aligned} P : (F_2^8)^8 &\longrightarrow (F_2^8)^8 \\ & y_1 \,|\, y_2 \,|\, y_3 \,|\, y_4 \,|\, y_5 \,|\, y_6 \,|\, y_7 \,|\, y_8 \longrightarrow z_1 \,|\, z_2 \,|\, z_3 \,|\, z_4 \,|\, z_5 \,|\, z_6 \,|\, z_7 \,|\, z_8 \\ & z_1 = y_1 \oplus y_3 \oplus y_4 \oplus y_6 \oplus y_7 \oplus y_8, \qquad z_5 = y_1 \oplus y_2 \oplus y_6 \oplus y_7 \oplus y_8, \\ & z_2 = y_1 \oplus y_2 \oplus y_4 \oplus y_5 \oplus y_7 \oplus y_8, \qquad z_6 = y_2 \oplus y_3 \oplus y_5 \oplus y_7 \oplus y_8, \\ & z_3 = y_1 \oplus y_2 \oplus y_3 \oplus y_5 \oplus y_6 \oplus y_8, \qquad z_7 = y_3 \oplus y_4 \oplus y_5 \oplus y_6 \oplus y_8, \\ & z_4 = y_2 \oplus y_3 \oplus y_4 \oplus y_5 \oplus y_6 \oplus y_7, \qquad z_8 = y_1 \oplus y_4 \oplus y_5 \oplus y_6 \oplus y_7. \end{aligned}$$

Key Scheduling Algorithm of Camellia. First of all, two 128-bit variables K_L and K_R are generated from the master key K. For Camellia-128, the 128-bit key K is used as K_L, and K_R is 0. For Camellia-192, the left 128 bits of the key K is used as K_L, and concatenation of the right 64-bit of K and the complement of the right 64-bit of K is used as K_R. For Camellia-256, the left 128-bit of the key K is used as K_L and the right 128-bit of K is used as K_R. Then two 128-bit variables K_A and K_B are generated from K_L and K_R, but note that K_B is used only if the length of the master key is 192 or 256 bits. Finally, the 64-bit round subkeys K_r are generated by rotating (K_L, K_R, K_A, K_B) and then taking the left half or the right half of them. More details are shown in [1], and in the following we only present some observations which are useful for our later attacks.

For Camellia-128, the round subkeys $K_r (1 \leq r \leq 18)$ are generated by rotating (K_L, K_A), and we can get the following expressions:

$$K_1 = (K_A \lll 0)_{L\,(64)}, \qquad K_2 = (K_A \lll 0)_{R\,(64)},$$
$$K_{11} = (K_A \lll 60)_{L\,(64)}, \qquad K_{12} = (K_A \lll 60)_{R\,(64)}.$$

For Camellia-192/256, the round subkeys $K_r (1 \leq r \leq 24)$ are generated by rotating (K_L, K_R, K_A, K_B), and we can get the following expressions:

$$K_1 = (K_B \lll 0)_{L\,(64)}, \qquad K_2 = (K_B \lll 0)_{R\,(64)},$$
$$K_3 = (K_R \lll 15)_{L\,(64)}, \qquad K_4 = (K_R \lll 15)_{R\,(64)},$$
$$K_{13} = (K_R \lll 60)_{L\,(64)}, \qquad K_{14} = (K_R \lll 60)_{R\,(64)},$$
$$K_{15} = (K_B \lll 60)_{L\,(64)}, \qquad K_{16} = (K_B \lll 60)_{R\,(64)}.$$

3 Preliminaries

3.1 Notations

Camellia is a byte-oriented block cipher, in which the 128-bit intermediate variables are represented as 16 bytes and the 64-bit round subkeys are represented as 8 bytes. The subkey of the r-th round is represented as $K_r = (k_{r,1}, k_{r,2}, k_{r,3}, k_{r,4}, k_{r,5}, k_{r,6}, k_{r,7}, k_{r,8})$. Furthermore, $k_{r,1}[i \sim j](i, j = 1, 2, \ldots, 8, i \leq j)$ denotes the i-th to the j-th bits of $k_{r,1}$.

For a pair of plaintexts (L_0, R_0) and (L_0^*, R_0^*), we denote the plaintext difference as $(\Delta L_0, \Delta R_0)$, where $\Delta L_0 = L_0 \oplus L_0^*$, $\Delta R_0 = R_0 \oplus R_0^*$. $(\Delta L_r, \Delta R_r)$ denotes the output difference of the r-th round. ΔL_r and ΔR_r can be represented as 8 bytes, such as $\Delta L_r = (a, 0, 0, 0, 0, 0, 0, 0)$ and $\Delta R_r = (?, ?, ?, 0, ?, 0, 0, ?)$, where 0 denotes a zero byte difference, a denotes a nonzero byte difference and the question mark ? denotes an unknown byte difference(two bytes marked with ? may be different).

3.2 The 8-Round Impossible Differential of Camellia

In [5] Wu et al presented an impossible differential attack on 12-round Camellia-192/256, which was based on the following 8-round impossible differential.

$$(0, 0, 0, 0, 0, 0, 0, 0, a, 0, 0, 0, 0, 0, 0, 0) \not\rightarrow (h, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0).$$

where a and h are arbitrary nonzero bytes. Refer to [5] for more details, and the 8-round impossible differential is also illustrated in Fig. 1. In this paper, we will exploit this 8-round impossible differential and improve the impossible differential cryptanalysis of Camellia up to 12-round Camellia-128 and 16-round Camellia-256.

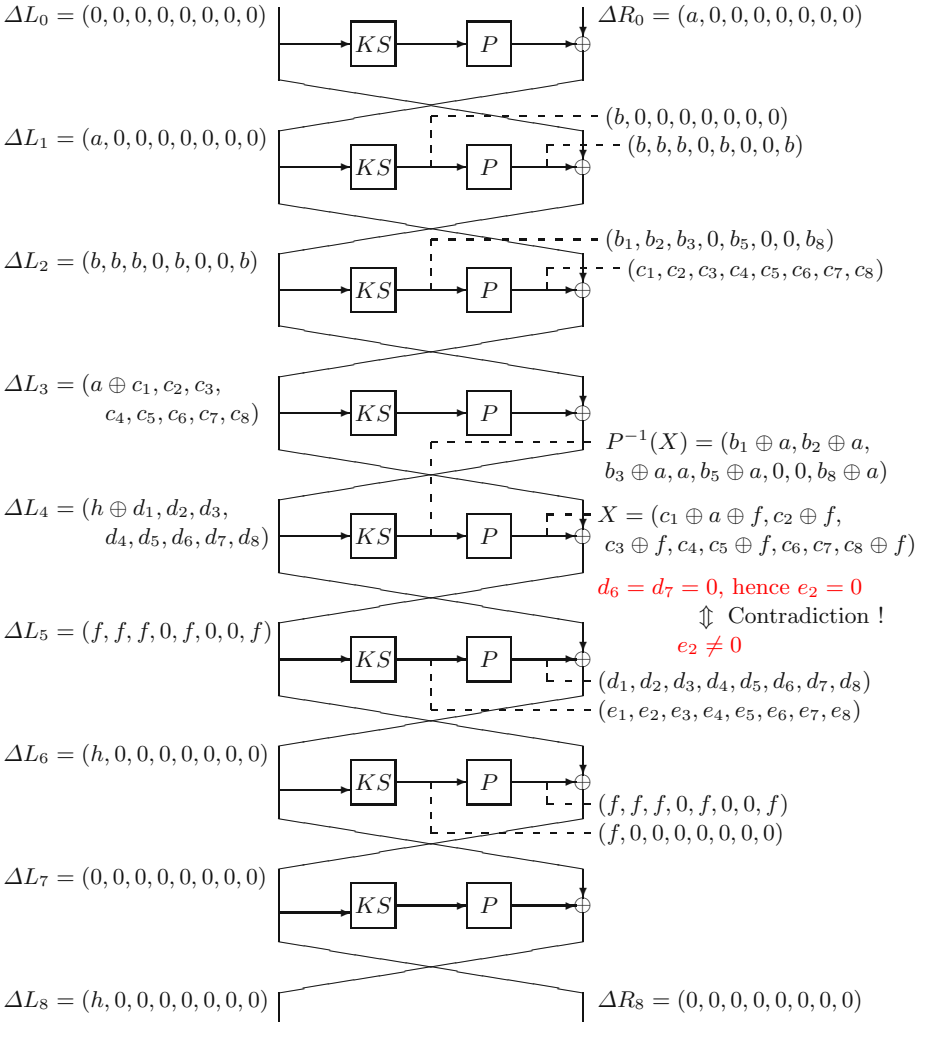

Fig. 1. 8-Round Impossible Differential of Camellia

3.3 Some Properties of Camellia

In this subsection, we exploit some properties of the key scheduling algorithm of Camellia-128 and Camellia-192/256, and present the following observations of the relations between round subkeys.

For Camellia-128, according to its key scheduling algorithm, we know that:

$$K_1 = (K_A \lll 0)_L, \quad K_{11} = (K_A \lll 60)_L,$$
$$K_2 = (K_A \lll 0)_R, \quad K_{12} = (K_A \lll 60)_R.$$

Therefore, the five bytes of K_1 and K_{12} at positions $(1, 2, 3, 5, 8)$ can be expressed as follows, respectively.

$$k_{1,1} = K_A[1 \sim 8], \qquad k_{12,1} = K_A[125 \sim 128, 1 \sim 4],$$
$$k_{1,2} = K_A[9 \sim 16], \qquad k_{12,2} = K_A[5 \sim 12],$$
$$k_{1,3} = K_A[17 \sim 24], \qquad k_{12,3} = K_A[13 \sim 20],$$
$$k_{1,5} = K_A[33 \sim 40], \qquad k_{12,5} = K_A[29 \sim 36],$$
$$k_{1,8} = K_A[57 \sim 64], \qquad k_{12,8} = K_A[53 \sim 60].$$

Furthermore, the first bytes of K_2 and K_{11} are expressed as follows, respectively.

$$k_{2,1} = K_A[65 \sim 72], \qquad k_{11,1} = K_A[61 \sim 68].$$

According to the above expressions, we can obtain the following property of Camellia-128:

Property 1. *For the round subkeys of Camellia-128:*

1) $(k_{1,1}, k_{1,2}, k_{1,3}, k_{1,5}, k_{1,8})$ *and* $(k_{12,1}, k_{12,2}, k_{12,3}, k_{12,5}, k_{12,8})$ *have 28 common bits.*
2) If $(k_{1,1}, k_{1,2}, k_{1,3}, k_{1,5}, k_{1,8})$ *and* $(k_{12,1}, k_{12,2}, k_{12,3}, k_{12,5}, k_{12,8})$ *are known, there remains only 16 unknown bits of* K_1*, namely* $(k_{1,4}[1 \sim 4], k_{1,6}, k_{1,7}[1 \sim 4])$.
3) If K_1 *and* $(k_{12,1}, k_{12,2}, k_{12,3}, k_{12,5}, k_{12,8})$ *are known, the value of* K_{12} *is determined.*
4) $k_{2,1}[1 \sim 4] = k_{11,1}[5 \sim 8], \quad k_{11,1}[1 \sim 4] = k_{1,8}[5 \sim 8]$.

For Camellia-192/256, according to the key scheduling algorithm, we notice that the round subkeys of Rounds 1, 2, 15 and 16 are all determined by the intermediate variable K_B, and the expressions are as follows.

$$K_1 = (K_B \lll 0)_L, \quad K_{15} = (K_B \lll 60)_L,$$
$$K_2 = (K_B \lll 0)_R, \quad K_{16} = (K_B \lll 60)_R.$$

Similarly, we can obtain the following properties of Camellia-192/256.

Property 2. *For the round subkeys of Camellia-192/256:*

1) If the value of K_1 *is known, then there remains only 4 unknown bits of* K_{16}.
2) If the value of K_1 *and* K_{16} *are known, there remains 60 unknown bits of* K_2.
3) If the value of K_1 *and* K_2 *are known, then the value of* K_{15} *is determined.*

Furthermore, according to the key scheduling algorithm of Camellia-192/256, the round subkeys of Rounds 3, 4, 13 and 14 are all determined by the intermediate variable K_R, and the expressions are as follows.

$$K_3 = (K_R \lll 15)_L, \quad K_{13} = (K_R \lll 60)_L,$$
$$K_4 = (K_R \lll 15)_R, \quad K_{14} = (K_R \lll 60)_R.$$

Based on these expressions, we can get the following relations between subkey bytes.

$$\begin{aligned}
k_{14,1} &= k_{4,6}[6 \sim 8] \,||\, k_{4,7}[1 \sim 5],\\
k_{14,2} &= k_{4,7}[6 \sim 8] \,||\, k_{4,8}[1 \sim 5],\\
k_{14,3} &= k_{4,8}[6 \sim 8] \,||\, k_{3,1}[1 \sim 5],\\
k_{14,5} &= k_{3,2}[6 \sim 8] \,||\, k_{3,3}[1 \sim 5],\\
k_{14,8} &= k_{3,5}[6 \sim 8] \,||\, k_{3,6}[1 \sim 5],\\
k_{13,1} &= k_{3,6}[6 \sim 8] \,||\, k_{3,7}[1 \sim 5].
\end{aligned}$$

Therefore, we can obtain another property of Camellia-192/256.

Property 3. *For the round subkeys of Camellia-192/256:*

1) $(k_{3,1}, k_{3,2}, k_{3,3}, k_{3,5}, k_{3,8})$ *and* $(k_{14,1}, k_{14,2}, k_{14,3}, k_{14,5}, k_{14,8})$ *have 16 common bits.*
2) If $(k_{3,1}, k_{3,2}, k_{3,3}, k_{3,5}, k_{3,8})$ *and* $(k_{14,1}, k_{14,2}, k_{14,3}, k_{14,5}, k_{14,8})$ *are known, there remains only 19 unknown bits of* K_3*, namely* $(k_{3,4}, k_{3,6}[6 \sim 8], k_{3,7})$.
3) If K_3 *and* $(k_{14,1}, k_{14,2}, k_{14,3}, k_{14,5}, k_{14,8})$ *are known, the value of* K_{14} *is determined.*
4) $k_{13,1} = k_{3,6}[6 \sim 8] \,||\, k_{3,7}[1 \sim 5]$.

Finally, according to the analysis in [6], the linear diffusion function P of Camellia satisfies the following property.

Property 4. [6] *For* $X, X^* \in (F_2^8)^8$*, if there exists an* h *such that* $P^{-1}(X \oplus X^* \oplus (h, 0, 0, 0, 0, 0, 0, 0))$ *has the form of* $(?, ?, ?, 0, ?, 0, 0, ?)$*, then there is only one possible value of* h.

4 Impossible Differential Cryptanalysis of Reduced-Round Camellia

4.1 Impossible Differential Attack on 12-Round Camellia-128

We set the 8-round impossible differential at Rounds 3 to 10, and present an impossible differential attack on 12-round Camellia-128, which is illustrated in Fig. 2. The first step of the attack is data collection, and we only choose the pairs whose output differences of Round 2 satisfy the above impossible differential distinguisher. According to the round function of Camellia, we can know that the required plaintext difference must have the following form:

$$\begin{aligned}
\Delta L_0 &= (u, u, u, 0, u, 0, 0, u),\\
\Delta R_0 &= P(?, ?, ?, 0, ?, 0, 0, ?) \oplus (?, 0, 0, 0, 0, 0, 0, 0).
\end{aligned}$$

Then by constructing appropriate plaintext structures, we can obtain plaintext pairs with the required difference, and this technique helps us reduce the data complexity.

The second step of the attack is data filtering. Based on certain property of ciphertext difference, we can filter out part of the wrong pairs and this may help

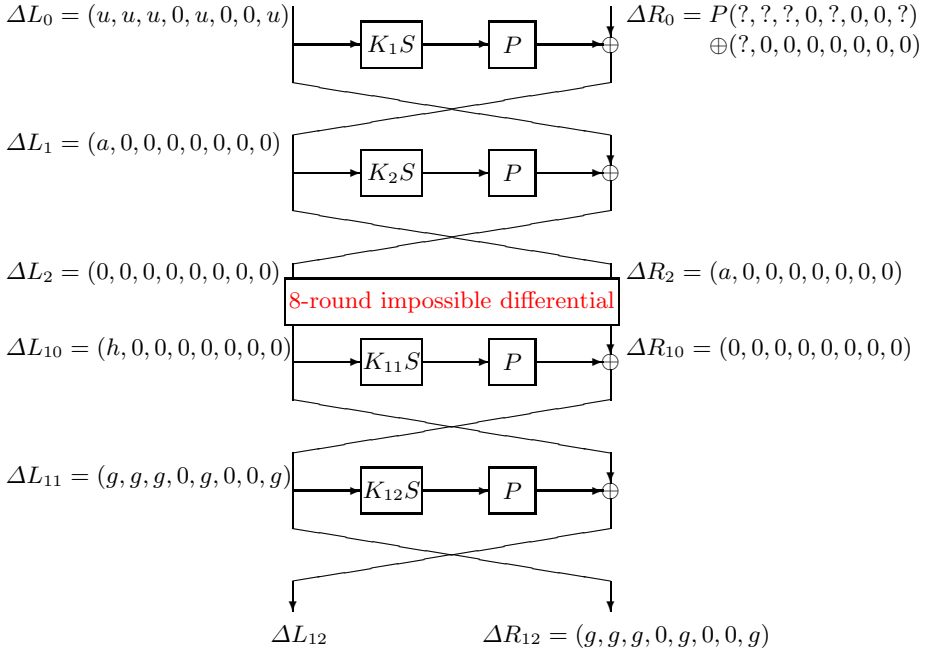

Fig. 2. Impossible Differential Attack on 12-Round Camellia-128

us reduce the time complexity of the following computation. Note that all the useful ciphertext pairs must satisfy the following condition:

$$\Delta L_{10} = (h, 0, 0, 0, 0, 0, 0, 0), \quad \Delta R_{10} = (0, 0, 0, 0, 0, 0, 0, 0).$$

where h denotes a nonzero byte, namely h has 255 possible values. Moreover, for every S-box of Camellia, when the input difference of S-box is nonzero, there are at most 2^7 possible output differences. Therefore, there are at most 255×2^7 possible output differences $(\Delta L_{11}, \Delta R_{11})$ after Round 11. Considering that there are 5 nonzero bytes of ΔL_{11}, namely bytes at positions (1,2,3,5,8), then there are at most $255 \times 2^7 \times (2^7)^5 \approx 2^{50}$ possible output differences $(\Delta L_{12}, \Delta R_{12})$ after Round 12. Therefore, in the data filtering step, the probability that a random pair remains after the test is about $2^{-78} = 2^{50} \times 2^{-128}$.

The third step of the attack is key recovery. According to Property 4, we can compute the output differences of S-boxes used in Round 1 and Round 12. Then by utilizing the difference distribution tables of S-boxes, we can recover 5 bytes $(k_{1,1}, k_{1,2}, k_{1,3}, k_{1,5}, k_{1,8})$ of K_1 and 5 bytes $(k_{12,1}, k_{12,2}, k_{12,3}, k_{12,5}, k_{12,8})$ of K_{12}. Furthermore, the corresponding pairs must be discarded if the relations between the subkeys contradict with Property 1-1. Lastly, we recover the correct key by discarding all the wrong subkeys using the impossible differential. In this step, we employ the divide-and-conquer technique when decrypting with the subkey guesses value and this also helps us reduce the time complexity of the attack.

In the following, we describe the attack procedure in detail.

1. Data Collection: Choose 2^m structures and each structure is as follows:

$$L_0 = (x, x, x, a_4, x, a_6, a_7, x),$$
$$R_0 = P(y_1, y_2, y_3, b_4, y_5, b_6, b_7, y_8) \oplus (y, c_2, c_3, c_4, c_5, c_6, c_7, c_8).$$

   where (a_i, b_j, c_l) are fixed constants, and the 7 bytes (x, y_1, y_2, y_3, y_5, y_8, y) take all possible values. Therefore, each structure contains 2^{56} plaintexts, which can generate about $2^{56} \times 2^{56}/2 = 2^{111}$ plaintext pairs. Hence 2^m structures can generate about 2^{111+m} plaintext pairs.
2. Data Filtering: According to the above analysis of the ciphertext differences, there are 2^{50} possible ciphertext differences. Therefore, after this test the expected number of remaining pairs is about $2^{111+m} \times 2^{50} \times 2^{-128} = 2^{33+m}$.
3. For each remaining pair $(L_0||R_0, L_{12}||R_{12})$ and $(L_0^*||R_0^*, L_{12}^*||R_{12}^*)$, do as follows:
   (a) Compute $P^{-1}(L_{12} \oplus L_{12}^* \oplus (h, 0, ..., 0))$ for all the 255 possible values of h. According to Property 4, we can obtain only one value of h such that it has the form $(?, ?, ?, 0, ?, 0, 0, ?)$. Similarly, we can compute the only one value of a such that $P^{-1}(R_0 \oplus R_0^* \oplus (a, 0, ..., 0))$ has the form $(?, ?, ?, 0, ?, 0, 0, ?)$.
   (b) Using the obtained input and output differences of the S-box in Round 1 and Round 12, together with the value of L_0 and R_{12}, we can calculate subkey bytes $(k_{1,1}, k_{1,2}, k_{1,3}, k_{1,5}, k_{1,8})$ and $(k_{12,1}, k_{12,2}, k_{12,3}, k_{12,5}, k_{12,8})$ by searching the difference distribution tables of S-boxes. Check if the deduced subkey bytes satisfy the 28-bit condition suggested by Property 1-1, and if this is not the case, discard the pair and return to Step 3 to try another pair. After this test, there remains about $2^{33+m} \times 2^{-28} = 2^{5+m}$ pairs.
   (c) For every guess of the 16 unknown bits $(k_{1,4}[1 \sim 4], k_{1,6}, k_{1,7}[1 \sim 4])$, do as follows. Note that according to Property 1-3, we can determine the value of K_{12} now.
      i. For every remaining pairs, encrypt the first round to get (L_1, L_1^*) using K_1, and decrypt the last round to get (R_{11}, R_{11}^*) using K_{12}.
      ii. Utilizing the difference distribution tables of S-boxes, we can calculate the value of $k_{2,1}$ using (L_1, L_1^*), a and $P^{-1}(L_0 \oplus L_0^*)$; Similarly we can calculate the value of $k_{11,1}$ using (R_{11}, R_{11}^*), h and $P^{-1}(R_{12} \oplus R_{12}^*)$.
      iii. Check if the subkey bytes satisfy the following equation suggested by Property 1-4.

$$k_{11,1} = k_{1,8}[5 \sim 8]||k_{2,1}[1 \sim 4].$$

      If there exists a plaintext pair that passes this test, then discard the 76-bit subkey guess value $(k_{1,1}, k_{1,2}, k_{1,3}, k_{1,5}, k_{1,8}, k_{12,1}, k_{12,2}, k_{12,3}, k_{12,5}, k_{12,8}, k_{1,4}[1 \sim 4], k_{1,6}, k_{1,7}[1 \sim 4], k_{2,1}, k_{11,1})$, as this is an impossible differential and the subkey guess satisfying it must be wrong.

Furthermore, the probability that a subkey guess may remain after this test is about $1 - 2^{-8}$. We choose $m = 9$, hence the number of remaining wrong subkey is about $2^{76}(1-2^{-8})^{2^{5+m}} \approx 2^{76} \times e^{-2^6} < 1$.

The data complexity of the attack is about $2^{56} \times 2^9 = 2^{65}$ CP, and the time complexity of the attack is estimated as follows. The time complexity of Step 1 is 2^{65} encryptions, and the memory spaces needed to store the plaintexts and ciphertexts are about 2^{66} blocks, where one block means 128 bits. In Step 2, choosing the qualified pairs requires about $2^{65} \times 2^{50} = 2^{115}$ MA (Memory Access), and the memory spaces needed to store the possible ciphertext differences are about 2^{50} blocks. In Step 3, the time complexity of Step (a) is about $2^{42} \times 2/12 < 2^{40}$ encryptions; the time complexity of Step (b) is less than 2^{39} encryptions, since the calculation of key using difference distribution table of S-box is only about one F computation; and the time complexity of Step (c) is about $2^{14} \times 2^{16} \times 2/12 < 2^{28}$ encryptions.

As a rule, one MA is equivalent to about one-round encryption of Camellia. Therefore, the total data complexity of the attack is 2^{65} CP, and the time complexity of the attack is less than $2^{111.5}$ encryptions, and the memory complexity of the attack is about 2^{66} blocks.

4.2 Impossible Differential Attack on 16-Round Camellia-256

We set the 8-round impossible differential at Rounds 5 to 12, and present an impossible differential attack on 16-round Camellia-256, which is illustrated in Fig. 3. The first step of the attack is data collection, and we also exploit the plaintext structure to reduce data complexity.

The second step of the attack is data filtering. In this step we try to filter out part of the wrong pairs whose plaintext and ciphertext differences can not satisfy the impossible differential, so as to reduce the computation workload for later analysis. According to the 8-round impossible differential, the output differences of a useful pair after Round 4 and Round 12 must be as follows, respectively.

$$\Delta L_4 = (0,0,0,0,0,0,0,0), \quad \Delta R_4 = (a,0,0,0,0,0,0,0),$$

$$\Delta L_{12} = (h,0,0,0,0,0,0,0), \quad \Delta R_{12} = (0,0,0,0,0,0,0,0).$$

where a and h are nonzero bytes. Therefore, for a useful pair, the left half of output difference after Round 1 must have the form $P(?,?,?,0,?,0,0,?) \oplus (?,0,0,0,0,0,0,0)$, and the left half of the output difference after Round 2 must have the form $(u,u,u,0,u,0,0,u)$. Similarly, the right half of the input difference before Round 16 must have the form $P(?,?,?,0,?,0,0,?) \oplus (?,0,0,0,0,0,0,0)$, and the right half of the input difference before Round 15 must have the form $(u,u,u,0,u,0,0,u)$. Hereafter, we denote the set of differences with the form $P(?,?,?,0,?,0,0,?) \oplus (?,0,0,0,0,0,0,0)$ as Π_1, and the set of differences with the form $(u,u,u,0,u,0,0,u)$ as Π_2. Obviously, there are 2^{48} elements in the set Π_1 and 2^8 elements in the set Π_2, namely $\#\{\Pi_1\} = 2^{48}$ and $\#\{\Pi_2\} = 2^8$. Therefore, the probability that a random plaintext pair is a useful pair for our analysis is about $2^{-144} = \frac{2^{48}}{2^{64}} \times \frac{2^{48}}{2^{64}} \times \frac{2^8}{2^{64}} \times \frac{2^8}{2^{64}}$.

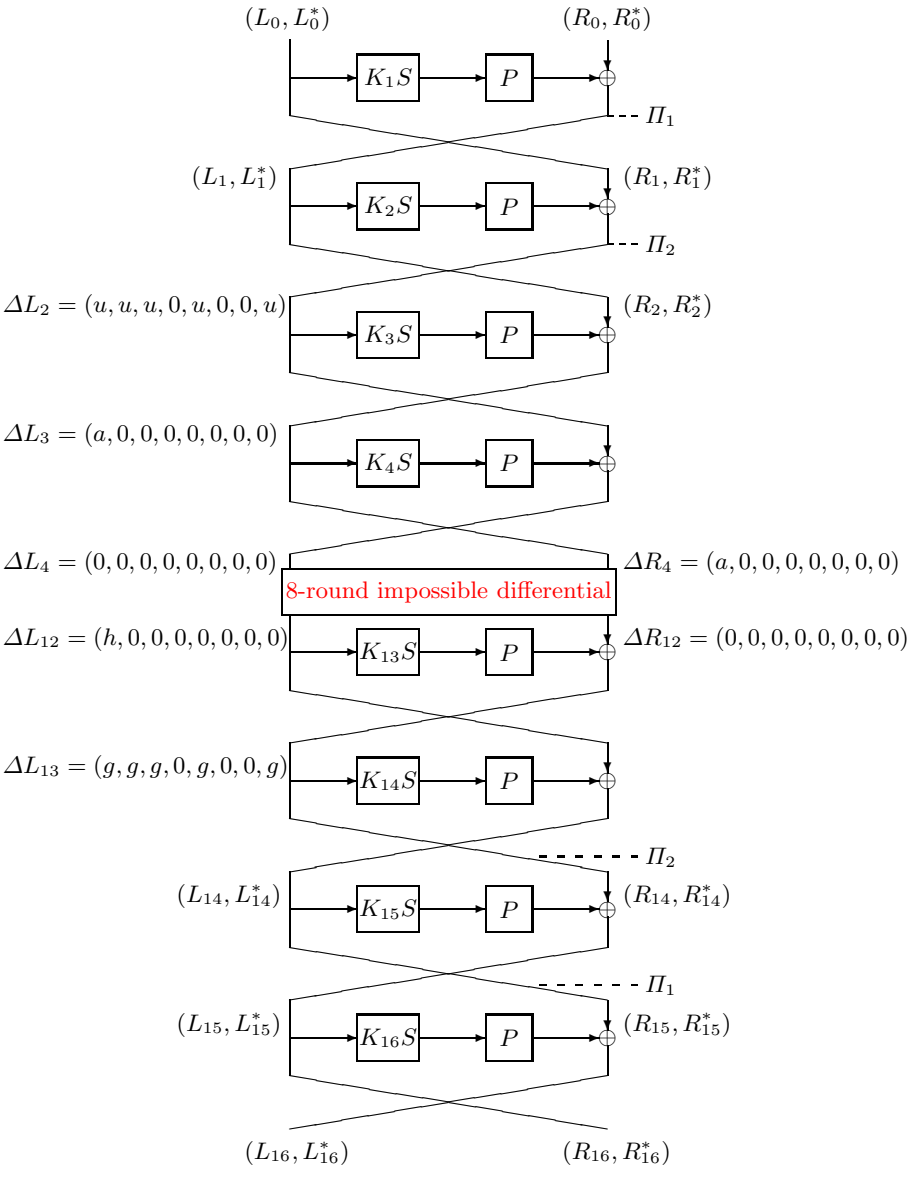

Fig. 3. Impossible Differential Attack on 16-Round Camellia-256

The third step of the attack is subkey guessing and sieving, and the divide-and-conquer technique is also used to reduce the time complexity. First of all, we need to guess part of the round subkeys to encrypt and decrypt the first and last two rounds, respectively. Then based on Property 4, we can compute the output differences of S-boxes used in Round 3 and Round 14. Utilizing the difference

distribution tables of the S-boxes, we can calculate 5 bytes $(k_{3,1}, k_{3,2}, k_{3,3}, k_{3,5}, k_{3,8})$ of K_3 and 5 bytes $(k_{14,1}, k_{14,2}, k_{14,3}, k_{14,5}, k_{14,8})$ of K_{14} , respectively. Then we use Property 2 and Property 3 to filter out the wrong pairs. Lastly, using the remaining pairs we can discard all the wrong subkey guesses based on the impossible differential, and thereby recover the correct key.

In the following, we describe the attack procedure in detail.

1. Data Collection: Choose 2^{89} plaintexts as follows:

$$L_0 = (x_1, ..., x_8), \qquad R_0 = (y_1, ..., y_8)$$

   where $x_i (1 \leq i \leq 8)$ and y_1 all take 64 arbitrary values chosen from F_2^8, and $y_j (1 < j \leq 8)$ all take 32 arbitrary values chosen from F_2^8. This way we can get about $2^{89} \cdot 2^{89}/2 \approx 2^{177}$ plaintext pairs, and these pairs need to be stored for later analysis which requires about 2^{178} blocks memory.
2. For every guess of K_1, do the followings:
   (a) Encrypt the first round for each of the 2^{89} plaintexts, and check if the output difference $F(L_0, K_1) \oplus F(L_0^*, K_1) \oplus R_0 \oplus R_0^*$ satisfies the form of $P(?, ?, ?, 0, ?, 0, 0, ?) \oplus (?, 0, 0, 0, 0, 0, 0, 0)$. If this is not the case, discard the corresponding plaintext pair. After this test, there remains about $2^{177} \times \frac{2^{48}}{2^{64}} = 2^{161}$ pairs.
   (b) According to Property 2-1, K_{16} has 60 common bits with K_1, and thus there are only 2^4 possible values of K_{16}. For every possible value of K_{16}, decrypt the last round for each of the 2^{89} plaintexts. Check if the input difference $F(R_{16}, K_{16}) \oplus F(R_{16}^*, K_{16}) \oplus L_{16} \oplus L_{16}^*$ of Round 16 satisfies the form of $P(?, ?, ?, 0, ?, 0, 0, ?) \oplus (?, 0, 0, 0, 0, 0, 0, 0)$. If this is not the case, discard the corresponding pair. After this test, there remains about $2^{161} \times \frac{2^{48}}{2^{64}} = 2^{145}$ pairs.
   (c) For every guess of K_2, encrypt the second round for the 2^{89} plaintexts. Note that there are only 60 unknown bits of K_2 after guessing the values of K_1 and K_{16} according to Property 2-2. Check if $F(L_1, K_2) \oplus F(L_1^*, K_2) \oplus R_1 \oplus R_1^*$ has the form of $(u, u, u, 0, u, 0, 0, u)$. If this is not the case, discard the pair. After this test there remains about $2^{145} \times \frac{2^8}{2^{64}} = 2^{89}$ pairs.
   (d) Decrypt Round 15 for each of the 2^{89} pairs using K_{15} which can be deduced by K_1 and K_2. Check if the difference $F(R_{15}, K_{15}) \oplus F(R_{15}^*, K_{15}) \oplus L_{15} \oplus L_{15}^*$ satisfies the form $(u, u, u, 0, u, 0, 0, u)$. Discard the unsatisfied pairs, and after this step there remains about $2^{89} \times \frac{2^8}{2^{64}} = 2^{33}$ pairs.
3. For each of the 2^{128} possible candidates of (K_1, K_2, K_{15}, K_{16}), and for each of the 2^{33} remained pairs $(L_0||R_0, L_{16}||R_{16})$ and $(L_0^*||R_0^*, L_{16}^*||R_{16}^*)$, do as follows:
   (a) Encrypt the first two rounds and decrypt the last two rounds to get $(L_2||R_2, L_{14}||R_{14})$ and $(L_2^*||R_2^*, L_{14}^*||R_{14}^*)$.
   (b) Compute $P^{-1}(L_{14} \oplus L_{14}^* \oplus (h, 0, ..., 0))$ for all the 255 possible values of h. According to Property 4, we can obtain only one value of h such that it has the form $(?, ?, ?, 0, ?, 0, 0, ?)$. Similarly, we can compute the

only one value of a such that $P^{-1}(R_2 \oplus R_2^* \oplus (a, 0, ..., 0))$ has the form $(?, ?, ?, 0, ?, 0, 0, ?)$.

(c) Using the obtained input and output differences of the S-box in Round 2 and Round 14, together with the value of L_2 and R_{14}, we can calculate subkey bytes $(k_{3,1}, k_{3,2}, k_{3,3}, k_{3,5}, k_{3,8})$ and $(k_{14,1}, k_{14,2}, k_{14,3}, k_{14,5}, k_{14,8})$ by searching the difference distribution tables of S-boxes. Check if the deduced subkey bytes satisfy the 16-bit condition suggested by Property 3-1, and if this is not the case, discard the pair and return to Step 3 to try another pair. After this test, there remains about $2^{33} \times 2^{-16} = 2^{17}$ pairs.

(d) For each possible value of the 19 unknown bits $(k_{3,4}, k_{3,6}[6 \sim 8], k_{3,7})$ of K_3, do as follows. Note that according to Property 3-3, we can know the value of K_{14} now.

i. For every remaining pairs, encrypt Round 3 using K_3 and decrypt Round 14 using K_{14} to get (L_3, L_{12}) and (L_3^*, L_{12}^*).

ii. Utilizing the difference distribution tables of S-boxes, calculate the value of $k_{4,1}$ using (L_3, L_3^*), a and $P^{-1}(L_2 \oplus L_2^*)$; calculate the value of $k_{13,1}$ using (L_{12}, L_{12}^*), h and $P^{-1}(L_{13} \oplus L_{13}^*)$.

iii. Check if the subkey bytes satisfy the following equation suggested by Property 3-4.

$$k_{13,1} = k_{3,6}[6 \sim 8] \,||\, k_{3,7}[1 \sim 5].$$

If there exists a plaintext pair that passes this test, then discard the $221 (= 128 + 64 + 19)$ bits subkey guess(K_B, K_3, K_{14}), as this is an impossible differential and the subkey guess satisfies it must be wrong. Furthermore, the probability that a subkey guess may remain is about $1 - 2^{-8}$, and the number of remaining wrong subkey is about $2^{221}(1 - 2^{-8})^{2^{17}} \approx 2^{221} \times e^{-2^9} < 1$.

The data complexity of the attack is about 2^{89} CP, and the time complexity of the attack can be estimated as follows. Step 2(a) has a time complexity of about $2^{64} \times 2^{89} \times 2^{-4} = 2^{149}$, and needs about $2^{48} \times 64$ bits memory spaces to store the elements in the set Π_1, and then requires about $2^{64} \times 2^{89} \times 2^{48} = 2^{201} MA$(Memory Access) for testing the qualified pairs. Step 2(b) has a time complexity of about $2^{64} \times 2^4 \times 2^{89} \times 2^{-4} = 2^{153}$, and requires about $2^{64} \times 2^4 \times 2^{89} \times 2^{48} = 2^{205} MA$(Memory Access) for testing the qualified pairs. Step 2(c) has a time complexity of about $2^{64} \times 2^{64} \times 2^{89} \times 2^{-4} = 2^{213}$, and needs about $2^{64} \times 2^{64} \times 2^{89} \times 2^8 = 2^{225} MA$ for testing the qualified pairs. Step 2(d) has a time complexity of about $2^{64} \times 2^{64} \times 2^{89} \times 2^{-4} = 2^{213}$ encryptions and $2^{225} MA$. In Step 3, the time complexity of Step 3(a) is about $2^{128} \times 2^{33} \times 2/4 = 2^{160}$; the time complexity of Step 3(b) is about $2^{128} \times 2^{33} \times 2/16 = 2^{158}$; the time complexity of Step 3(c) is less than 2^{158} encryptions, since the time to calculate subkey using difference distribution table of S-box is only about one F computation; and the time complexity of Step 3(d) is about $2^{128} \times 2^{19} \times 2^{17} \times 2/16 < 2^{151}$ encryptions.

Therefore, the total data complexity of the attack is 2^{89} CP, and the time complexity of the attack is less than $2^{222.1}$ encryptions, and the memory complexity of the attack is about 2^{178} blocks.

5 Conclusion

In [5] Wu et al constructed some 8-round impossible differentials of Camellia, and based on it they successfully attacked Camellia reduced up to 12 rounds using impossible differential cryptanalysis. Then in [6] Lu et al observed some new properties of the linear diffusion function P, and by using the same 8-round impossible differential they improved the impossible differential cryptanalysis of Camellia. However, all of these impossible differential attacks on Camellia have not taken the key scheduling algorithm into account. In this paper, we present some observations of the relations between round subkeys of Camellia, and by taking advantage of these relations and some novel techniques (such as differential cryptanalysis, divide-and-conquer etc.), we improve the impossible differential attack on Camellia up to 12-round Camellia-128 and 16-round Camellia-256. These results are better than any previously published cryptanalytic results on Camellia without the FL/FL^{-1} functions and whitening layers. Note that our method used in this paper does not apply to Camellia-192 effectively, since the relations between round subkeys of Camellia-192 are difficult to exploit in the attack process.

Acknowledgments. This work is supported by the National High-Tech Research and Development Program of China (No.2007AA01Z470), the National Natural Science Foundation of China (No.90604036), and the National Grand Fundamental Research 973 Program of China (No.2004CB318004). Moreover, the authors are very grateful to the anonymous reviewers for their comments and editorial suggestions.

References

1. Aoki, K., Ichikawa, T., Kanda, M., et al.: Specification of Camellia–a 128-bit Block Cipher. In: Selected Areas in Cryptography-SAC 2000. LNCS, vol. 2012, pp. 183–191. Springer, Heidelberg (2001)
2. Lee, S., Hong, S.H., Lee, S.-J., Lim, J.-I., Yoon, S.H.: Truncated differential cryptanalysis of camellia. In: Kim, K.-c. (ed.) ICISC 2001. LNCS, vol. 2288, pp. 32–38. Springer, Heidelberg (2002)
3. Sugita, M., Kobara, K., Imai, H.: Security of reduced version of the block cipher camellia against truncated and impossible differential cryptanalysis. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 193–207. Springer, Heidelberg (2001)
4. Hatano, Y., Sekine, H., Kaneko, T.: Higher order differential attack of (II). In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 129–146. Springer, Heidelberg (2003)
5. Wu, W., Zhang, W., Feng, D.: Impossible Differential Cryptanalysis of Reduced-Round ARIA and Camellia. Journal of Computer Science and Technology 22(3), 449–456 (2007)
6. Lu, J., Kim, J.-S., Keller, N., Dunkelman, O.: Improving the efficiency of impossible differential cryptanalysis of reduced camellia and MISTY1. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 370–386. Springer, Heidelberg (2008)

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. Yeom, Y., Park, S., Kim, I.: On the Security of Camellia against the Square Attack. In: FSE 2002. LNCS, vol. 2356, pp. 89–99. Springer, Heidelberg (2002)
9. Yeom, Y., Park, S., Kim, I.: A Study of Integral Type Cryptanalysis on Camellia. In: The 2003 Symposium on Cryptography and Information Security-SCIS 2003, Hamamatsu, Japan, pp. 26–29 (2003)
10. 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)
11. 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)
12. Wenling, W., Dengguo, F., Hua, C.: Collision attack and pseudorandomness of reduced-round camellia. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 252–266. Springer, Heidelberg (2004)
13. Jie, G., Zhongya, Z.: Improved collision attack on reduced round camellia. In: Pointcheval, D., Mu, Y., Chen, K. (eds.) CANS 2006. LNCS, vol. 4301, pp. 182–190. Springer, Heidelberg (2006)
14. Shirai, T.: Differential, Linear, Boomerang and Rectangle Cryptanalysis of Reduced-Round Camellia. In: Proceedings of the Third NESSIE Workshop, Munich, Germany, November 6-7 (2002), https://www.cosic.esat.kuleuven.be/nessie/
15. Wu, W., Feng, D.: Differential-Linear Cryptanalysis of Camellia. In: Progress on Cryptography, pp. 173–180. Kluwer Academic Publishers, Dordrecht (2004)
16. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. In: EUROCRYPT 1999. LNCS, vol. 2595, pp. 12–23. Springer, Heidelberg (1999)
17. Cheon, J.H., Kim, M., Kim, K., Lee, J.-Y., Kang, S.: Improved impossible differential cryptanalysis of rijndael and crypton. In: Kim, K.-c. (ed.) ICISC 2001. LNCS, vol. 2288, pp. 39–49. Springer, Heidelberg (2002)
18. Phan, R.C.-W.: Impossible Differential Cryptanalysis of 7-round AES. Information Processing Letters 91(1), 33–38 (2004)
19. Jakimoski, G., Desmedt, Y.: Related-Key Differential Cryptanalysis of 192-bit key AES Variants. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 208–221. Springer, Heidelberg (2004)
20. Zhang, W., Wu, W., Zhang, L., Feng, D.: Improved related-key impossible differential attacks on reduced-round AES-192. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 15–27. Springer, Heidelberg (2007)
21. Biham, E., Dunkelman, O., Keller, N.: Related-key impossible differential attacks on 8-round AES-192. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 21–33. Springer, Heidelberg (2006)
22. Zhang, W., Wu, W., Feng, D.: New results on impossible differential cryptanalysis of reduced AES. In: Nam, K.-H., Rhee, G. (eds.) ICISC 2007. LNCS, vol. 4817, pp. 239–250. Springer, Heidelberg (2007)
23. Tsunoo, Y., Tsujihara, E., Saito, T., Suzaki, T., Kubo, H.: Impossible Differential Cryptanalysis of CLEFIA. In: Fast Software Encryption-FSE 2008. Springer, Heidelberg (2008)