Cryptanalysis of Reduced Versions of the Camellia Block Cipher

Jiqiang Lu[1,⋆], Yongzhuang Wei[2,3,⋆⋆],
Jongsung Kim[4,⋆⋆⋆], and Pierre-Alain Fouque[1]

[1] Département d'Informatique, École Normale Supérieure,
45 Rue d'Ulm, Paris 75005, France
lvjiqiang@hotmail.com, Pierre-Alain.Fouque@ens.fr
[2] Guilin University of Electronic Technology,
Guilin City, Guangxi Province 541004, China
[3] State Key Lab of Information Security, Institute of Software,
Chinese Academy of Sciences, Beijing 100190, China
walker_wei@msn.com
[4] Division of e-Business, Kyungnam University,
449 Wolyoung-dong, Masan, Kyungnam, Korea
jongsung.k@gmail.com

Abstract. The Camellia block cipher has a 128-bit block length, a user key of 128, 192 or 256 bits long, and a total of 18 rounds for a 128-bit key and 24 rounds for a 192 or 256-bit key. It is a Japanese CRYPTREC-recommended e-government cipher, an European NESSIE selected cipher and an ISO international standard. In this paper, we describe a flaw in the approach used to choose plaintexts or ciphertexts in certain previously published square-like cryptanalytic results for Camellia and give possible approaches to correct them. Finally, by taking advantage of the early abort technique and a few observations on the key schedule, we present impossible differential attacks on 10-round Camellia with the FL/FL^{-1} functions under 128 key bits, 11-round Camellia with the FL/FL^{-1} functions under 192 key bits, 14-round Camellia without the FL/FL^{-1} functions under 192 key bits and 16-round Camellia without the FL/FL^{-1} functions under 256 key bits. These are better than any previously published cryptanalytic results for the respective versions of Camellia in terms of the numbers of attacked rounds.

Key words: Block cipher, Camellia, Square attack, Impossible differential cryptanalysis, The early abort technique.

⋆ This author was supported by the French ANR project SAPHIR II.
⋆⋆ This author was partially supported by the Natural Science Foundation of China (No. 60873259), the National Science Foundation of Guangxi (No. 2011GXNSFB018071), and the National Basic Research 973 Program of China (No. 2007CB311201).
⋆⋆⋆ This author was supported by the Basic Science Research Program through the National Research Foundation of Korea funded by the Ministry of Education, Science and Technology (No. 2011-0014876).

1 Introduction

The Camellia [1] block cipher was published in 2000; it has a 128-bit Feistel block length, a user key of 128, 192 or 256 bits long, and a total of 16 rounds if a 128-bit key is used and 24 rounds if a 192/256-bit key is used. For simplicity, we denote by Camellia-128/192/256 the three versions of Camellia that use 128, 192 and 256 key bits, respectively. Camellia became a CRYPTREC e-government recommended cipher [6] in 2002, a NESSIE selected block cipher [20] in 2003, and was adopted as an ISO international standard [11] in 2005.

So far, in terms of the numbers of attacked rounds, the best previously published cryptanalytic results on Camellia with the FL/FL^{-1} functions are the square [7] attack on 9-round Camellia-128 [8], the impossible differential [2, 12] attack on 10-round Camellia-192 [5], and the higher-order differential [13] and impossible differential attacks on 11-round Camellia-256 [5,10]; and the best previously published cryptanalytic results on Camellia without the FL/FL^{-1} functions are the impossible differential attacks on 12-round Camellia-128/192 [14,19] and 15-round Camellia-256 [5]. Besides, Biryukov and Nikolic [4] analysed a reduced Camellia-128 with a modified key schedule.

In this paper, we are particularly concerned with some previously published square and collision attacks on Camellia, as follows. In 2002, Yeom et al. [23] presented square attacks on up to 9 rounds of Camellia-256 with the FL/FL^{-1} functions, and in 2003 they improved their square attacks to break up to 10 rounds of Camellia-256 with the FL/FL^{-1} functions [24]. In 2005, Duo et al. [8] gave square/collision attacks on up to 9 rounds of Camellia-128 with/without the FL/FL^{-1} functions and up to 11 rounds of Camellia-256 without the FL/FL^{-1} functions; and in 2007 they presented improved versions to attack 9 rounds of Camellia-128 with/without the FL/FL^{-1} functions and up to 12 rounds of Camellia-256 without the FL/FL^{-1} functions [9]. However, we find a common flaw in these attacks; the flaw is from the approach used to choose plaintexts/ciphertexts. Possible approaches to correct them are given. This flaw also exists in Hatano et al.'s higher-order differential attacks on up to 11 rounds of Camellia-256 with/without the FL/FL^{-1} functions [10], but they can be easily corrected with the same complexity. It is worthy to note that Hatano et al.'s attack on 11-round Camellia-256 with FL/FL^{-1} functions is still one of the best previously published cryptanalytic results for Camellia-256.

Finally, by taking advantage of the early abort technique [16] and a few observations on the key schedule, we present impossible differential attacks on 10-round Camellia-128 with FL/FL^{-1} functions, 11-round Camellia-192 with FL/FL^{-1} functions, 14-round Camellia-192 without FL/FL^{-1} functions and 16-round Camellia-256 without FL/FL^{-1} functions, basing them on either the 6-round impossible differentials of Camellia with FL/FL^{-1} functions due to Chen et al. [5] or the 8-round impossible differentials of Camellia without FL/FL^{-1} functions due to Wu et al. [22]. Table 1 summarises the best previously published and our main cryptanalytic results on Camellia, where CP and KP refer to the numbers of chosen plaintexts and known plaintexts, respectively, Enc. refers to the required number of encryption operations of the relevant reduced version of

Table 1. The best previously published and our main cryptanalytic results on Camellia

Key Length	FL/FL^{-1}	Rounds	Attack Type	Data	Time	Source
128 bits	yes	9	Square	2^{48}CP	2^{122}Enc.	[8]
		10	Impossible differential	2^{118}CP	2^{118}Enc.	Sect. 4.2
	no	12	Impossible differential	$2^{116.3}$CP	$2^{116.6}$Enc.	[19]
192 bits	yes	10†	Impossible differential	2^{121}CP	$2^{175.3}$Enc.	[5]
		10	Impossible differential	2^{121}CP	2^{144}Enc.	[5]
		11	Impossible differential	2^{118}CP	$2^{163.1}$Enc.	Sect. 4.3
	no	12‡	Impossible differential	2^{119}CP	$2^{147.3}$Enc.	[14]
		14	Impossible differential	$2^{122.5}$KP	$2^{187.7}$Enc.	Sect. 4.4
		14	Impossible differential	2^{117}CP	$2^{182.2}$Enc.	Sect. 4.4
256 bits	yes	11‡	High-order differential	2^{93}CP	$2^{255.6}$Enc.	[10]
		11†	Impossible differential	2^{121}CP	$2^{206.8}$Enc.	[5]
	no	15	Impossible differential	$2^{122.5}$KP	$2^{236.1}$Enc.	[5]
		16	Impossible differential	2^{123}KP	2^{249}Enc.	Sect. 4.5

†: Include whitening operations; ‡: Can include whitening operations by making use of an equivalent structure of Camellia.

Camellia-128/192/256, "yes" means "with FL/FL^{-1} functions", and "no" means "without FL/FL^{-1} functions".[1]

The remainder of the paper is organised as follows. In the next section, we describe the notation and the Camellia block cipher. In Section 3 we address the flaw in the concerned cryptanalytic results for Camellia. In Section 4, we present our impossible differential attacks on Camellia. Section 5 concludes this paper.

2 Preliminaries

In this section we give the notation used in this paper and briefly describe the Camellia block cipher.

2.1 Notation

The bits of a value are numbered from left to right, starting with 1. We use the following notation throughout this paper.

- $\oplus$ bitwise logical exclusive OR (XOR) of two bit strings of the same length
- $\lll$ left rotation of a bit string
- $||$ bit string concatenation
- $\circ$ functional composition. When composing functions X and Y, X $\circ$ Y denotes

[1] This paper was part of a manuscript in which we used three different techniques to cryptanalyse Camellia. Because of page constraint we split the manuscript into three parts; see [17, 18] for the other two parts.

the function obtained by first applying X and then applying Y
$\overline{X}$ bitwise logical complement of a bit string X
$\star$ an arbitrary 8-bit value, where two values represented by the $\star$ symbol may be different

2.2 The Camellia Block Cipher

The Camellia [1] cipher has a Feistel structure with key-dependent logical functions $\mathbf{FL}/\mathbf{FL}^{-1}$ inserted after every six rounds, plus additional whitening operations. The round function $\mathbf{F}$ comprises a XOR operation with a round subkey, then a non-linear substitution operation $\mathbf{S}$ constructed by applying eight 8×8-bit S-boxes $\mathrm{S}_1, \mathrm{S}_2, \mathrm{S}_3, \mathrm{S}_4, \mathrm{S}_5, \mathrm{S}_6, \mathrm{S}_7$ and S_8 in parallel to the input, and finally a linear permutation $\mathbf{P}$ equivalent to pre-multiplication by a 8×8 byte matrix P. The matrix P and its reverse P^{-1} are given below. As we do not use explicitly the $\mathbf{FL}/\mathbf{FL}^{-1}$ functions in this paper, we omit the description of the two functions.

$$\mathrm{P} = \begin{pmatrix} 1&0&1&1&0&1&1&1 \\ 1&1&0&1&1&0&1&1 \\ 1&1&1&0&1&1&0&1 \\ 0&1&1&1&1&1&1&0 \\ 1&1&0&0&0&1&1&1 \\ 0&1&1&0&1&0&1&1 \\ 0&0&1&1&1&1&0&1 \\ 1&0&0&1&1&1&1&0 \end{pmatrix}, \ \mathrm{P}^{-1} = \begin{pmatrix} 0&1&1&1&0&1&1&1 \\ 1&0&1&1&1&0&1&1 \\ 1&1&0&1&1&1&0&1 \\ 1&1&1&0&1&1&1&0 \\ 1&1&0&0&1&0&1&1 \\ 0&1&1&0&1&1&0&1 \\ 0&0&1&1&1&1&1&0 \\ 1&0&0&1&0&1&1&1 \end{pmatrix}.$$

Camellia takes a 128-bit plaintext P as input, represented as 16 bytes, and has a total of N_r rounds, where N_r is 18 for Camellia-128 and 24 for Camellia-192/256. Its encryption procedure is as follows, where L_0, R_0, L_i, R_i, $\widehat{L}_i$ and $\widehat{R}_i$ are 64-bit variables, $KW_1, KW_2, KW_3, KW_4, KI_{\frac{i}{3}-1}, KI_{\frac{i}{3}}$ and K_i are 64-bit subkeys.

1. $L_0||R_0 = P \oplus (KW_1||KW_2)$
2. For $i = 1$ to N_r:
   if $i = 6$ or 12 (or 18 for Camellia-192/256),
   $\widehat{L}_i = \mathbf{F}(L_{i-1}, K_i) \oplus R_{i-1}$, $\widehat{R}_i = L_{i-1}$;
   $L_i = \mathbf{FL}(\widehat{L}_i, KI_{\frac{i}{3}-1})$, $R_i = \mathbf{FL}^{-1}(\widehat{R}_i, KI_{\frac{i}{3}})$;
   else
   $L_i = \mathbf{F}(L_{i-1}, K_i) \oplus R_{i-1}$, $R_i = L_{i-1}$;
3. Ciphertext $C = (R_{N_r} \oplus KW_3)||(L_{N_r} \oplus KW_4)$,

We refer to the ith iteration of Step 2 in the above description as Round i, and write $K_{i,j}$ for the j-th byte of K_i, $(1 \leqslant j \leqslant 8)$.

We now describe the key schedule of Camellia. All the subkeys KW_j, KI_l, K_i, $(1 \leqslant j \leqslant 4, 1 \leqslant l \leqslant \frac{N_r-6}{3}, 1 \leqslant i \leqslant N_r)$, are derived from a user key K by the following algorithm, where K is 128 bits long for Camellia-128, 192 bits long for Camellia-192 and 256 bits long for Camellia-256.

1. Generate two 128-bit strings K_L and K_R from K in the following way: For Camellia-128, K_L is the 128-bit key K, and K_R is zero; for Camellia-192, K_L is the left 128 bits of K, and K_R is the concatenation of the right 64 bits of K and the complement of the right 64 bits of K; and for Camellia-256, K_L is the left 128 bits of K, and K_R is the right 128 bits of K.

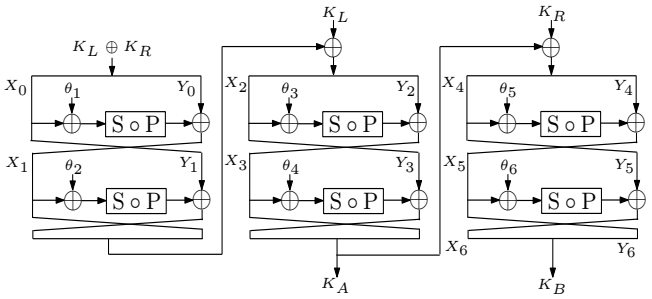

Fig. 1. Generation of K_A and K_B

2. Generate two 128-bit strings K_A and K_B from (K_L, K_R) as follows, where $X_0, Y_0, X_i', Y_i', X_i, Y_i$ are 64-bit variables, and θ_i are public constants. It is shown diagrammatically in Fig. 1.
   (a) $X_0||Y_0 = K_L \oplus K_R$
   (b) For $i = 1$ to 6:
   if $i = 2$,
   $X_i' = \mathbf{F}(X_{i-1}, \theta_i) \oplus Y_{i-1}$, $Y_i' = X_{i-1}$, $X_i||Y_i = K_L \oplus (X_i'||Y_i')$;
   else if $i = 4$,
   $X_i' = \mathbf{F}(X_{i-1}, \theta_i) \oplus Y_{i-1}$, $Y_i' = X_{i-1}$, $X_i||Y_i = K_R \oplus (X_i'||Y_i')$;
   else
   $X_i = \mathbf{F}(X_{i-1}, \theta_i) \oplus Y_{i-1}$, $Y_i = X_{i-1}$;
   (c) $K_A = (X_4'||Y_4'), K_B = (X_6||Y_6)$.
3. The subkeys are as follows.[2]
   - For Camellia-128: $K_8 = (K_L \lll 45)[65 \sim 128], K_9 = (K_A \lll 45)[1 \sim 64], K_{16} = (K_A \lll 94)[65 \sim 128], K_{17} = (K_L \lll 111)[1 \sim 64], \cdots$.
   - For Camellia-192/256: $K_1 = (K_B \lll 0)[1 \sim 64], K_2 = (K_B \lll 0)[65 \sim 128], K_3 = (K_R \lll 15)[1 \sim 64], K_4 = (K_R \lll 15)[65 \sim 128], K_5 = (K_A \lll 15)[1 \sim 64], K_8 = (K_B \lll 30)[65 \sim 128], K_{11} = (K_A \lll 45)[1 \sim 64], K_{12} = (K_B \lll 45)[65 \sim 128], K_{13} = (K_R \lll 60)[1 \sim 64], K_{14} = (K_R \lll 60)[65 \sim 128], K_{15} = (K_B \lll 60)[1 \sim 64], K_{16} = (K_B \lll 60)[65 \sim 128], K_{22} = (K_A \lll 94)[65 \sim 128], K_{23} = (K_L \lll 111)[1 \sim 64] \cdots$.

3 On Certain Cryptanalytic Results for Camellia

In this section we describe the flaw in Yeom et al.'s square attacks [23, 24] and Duo et al.'s square and collision attacks [8, 9], and give possible approaches to correct them. Note that here we focus on only the main results from these papers, and the flaw also exists in some results for a smaller number of rounds.

[2] Here we give only the subkeys concerned in this paper, $(K_L \lll 45)[65 \sim 128]$ represents bits $(65, 66, \cdots, 128)$ of $(K_l \lll 45)$, and so on.

3.1 On Yeom et al.'s Square Attacks

The Flaw. Let's consider Yeom et al.'s 9-round Camellia-256 attack described in [23], which can be summarised as follows:

1. Construct a 4-round square distinguisher for a Λ-set, where a Λ-set was defined to be a set of 256 (16-byte) values with the 9-th byte position taking all the possible values in $\{0,1\}^8$ and the other 15 bytes fixed.
2. The attacked 9 rounds involves 2 rounds immediately before the 4-round square distinguisher and 3 rounds immediately after the 4-round square distinguisher.
3. For every guess of the 6 required unknown subkey bytes in the 2 rounds immediately before the 4-round square distinguisher, obtain 256 plaintexts that can produce a Λ-set at the output of Round 2. The 6 required unknown subkey bytes are $K_{1,1}, K_{1,2}, K_{1,3}, K_{1,5}, K_{1,8}, K_{2,1}$ in this attack.
4. For every guess of the required unknown subkey bytes in the 3 immediately after the 4-round square distinguisher, partially decrypt the set of 256 plaintexts to check whether they meet the output property of the 4-round square distinguisher. Finally, the correct key guess can be identified with a sufficient number of Λ-sets.

In Step 3, Yeom et al. used the following approach to obtain 256 plaintexts that can produce a Λ-set after Round 2: Denote a guess for $(K_{1,1}, K_{1,2}, K_{1,3}, K_{1,5}, K_{1,8}, K_{2,1})$ by $(K^*_{1,1}, K^*_{1,2}, K^*_{1,3}, K^*_{1,5}, K^*_{1,8}, K^*_{2,1})$, then below are the 256 plaintexts $(P_L^{(x)}, P_R^{(x)})$ that can produce a Λ-set at the output of Round 2, where $x = 0, 1, \cdots, 255$ and $\beta_4, \beta_6, \beta_7$ are arbitrary constants, (Note that there can exist an arbitrary constant in some places, but we omit them to make things clearer, for they do not change much).

$$P_L^{(x)} = \begin{pmatrix} \mathrm{S}_1(x \oplus K^*_{2,1}) \\ \mathrm{S}_1(x \oplus K^*_{2,1}) \\ \mathrm{S}_1(x \oplus K^*_{2,1}) \\ \beta_4 \\ \mathrm{S}_1(x \oplus K^*_{2,1}) \\ \beta_6 \\ \beta_7 \\ \mathrm{S}_1(x \oplus K^*_{2,1}) \end{pmatrix}^{\mathtt{T}}, \; P_R^{(x)} = \mathbf{P} \begin{pmatrix} \mathrm{S}_1(\mathrm{S}_1(x \oplus K^*_{2,1}) \oplus K^*_{1,1}) \\ \mathrm{S}_2(\mathrm{S}_1(x \oplus K^*_{2,1}) \oplus K^*_{1,2}) \\ \mathrm{S}_3(\mathrm{S}_1(x \oplus K^*_{2,1}) \oplus K^*_{1,3}) \\ 0 \\ \mathrm{S}_5(\mathrm{S}_1(x \oplus K^*_{2,1}) \oplus K^*_{1,5}) \\ 0 \\ 0 \\ \mathrm{S}_8(\mathrm{S}_1(x \oplus K^*_{2,1}) \oplus K^*_{1,8}) \end{pmatrix}^{\mathtt{T}} \oplus \begin{pmatrix} x \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}^{\mathtt{T}},$$

However, we find that there exists a flaw, and the 256 plaintexts $(P_L^{(x)}, P_R^{(x)})$ do not necessarily produce a Λ-set at the end of Round 2 when the guess $(K^*_{1,1}, K^*_{1,2}, K^*_{1,3}, K^*_{1,5}, K^*_{1,8}, K^*_{2,1})$ is correct. We explain this below.

1. The output $(L_1^{(x)}, R_1^{(x)})$ of Round 1 is as follows, where $c_1 = \mathrm{S}_4(\beta_4 \oplus K_{1,4}), c_2 = \mathrm{S}_6(\beta_6 \oplus K_{1,6}), c_3 = \mathrm{S}_7(\beta_7 \oplus K_{1,7})$, and c denotes a constant (two values represented by c may be different).

$$L_1^{(x)} = \mathbf{P} \begin{pmatrix} 0 \\ 0 \\ 0 \\ c_1 \\ 0 \\ c_2 \\ c_3 \\ 0 \end{pmatrix}^{\mathtt{T}} \oplus \begin{pmatrix} x \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}^{\mathtt{T}} = \begin{pmatrix} x \oplus c_1 \oplus c_2 \oplus c_3 \\ c \\ c \\ c \\ c \\ c \\ c \\ c \end{pmatrix}^{\mathtt{T}}, \; R_1^{(x)} = \begin{pmatrix} \mathrm{S}_1(x \oplus K^*_{2,1}) \\ \mathrm{S}_1(x \oplus K^*_{2,1}) \\ \mathrm{S}_1(x \oplus K^*_{2,1}) \\ \beta_4 \\ \mathrm{S}_1(x \oplus K^*_{2,1}) \\ \beta_6 \\ \beta_7 \\ \mathrm{S}_1(x \oplus K^*_{2,1}) \end{pmatrix}^{\mathtt{T}}.$$

2. The output $(L_2^{(x)}, R_2^{(x)})$ of Round 2 is as follows, where $c_4, \cdots, c_{11}$ are constants.

$$L_2^{(x)} = \begin{pmatrix} \mathrm{S}_1(x \oplus c_1 \oplus c_2 \oplus c_3 \oplus K_{2,1}) \oplus c_4 \oplus \mathrm{S}_1(x \oplus K_{2,1}^*) \\ \mathrm{S}_1(x \oplus c_1 \oplus c_2 \oplus c_3 \oplus K_{2,1}) \oplus c_5 \oplus \mathrm{S}_1(x \oplus K_{2,1}^*) \\ \mathrm{S}_1(x \oplus c_1 \oplus c_2 \oplus c_3 \oplus K_{2,1}) \oplus c_6 \oplus \mathrm{S}_1(x \oplus K_{2,1}^*) \\ c_7 \oplus \beta_4 \\ \mathrm{S}_1(x \oplus c_1 \oplus c_2 \oplus c_3 \oplus K_{2,1}) \oplus c_8 \oplus \mathrm{S}_1(x \oplus K_{2,1}^*) \\ c_9 \oplus \beta_6 \\ c_{10} \oplus \beta_7 \\ \mathrm{S}_1(x \oplus c_1 \oplus c_2 \oplus c_3 \oplus K_{2,1}) \oplus c_{11} \oplus \mathrm{S}_1(x \oplus K_{2,1}^*) \end{pmatrix}^{\mathtt{T}}, \; R_2^{(x)} = L_1^{(x)}.$$

Observe that $R_2^{(x)}$ meets the right-half property of a Λ-set, (though there is an unknown constant $c_1 \oplus c_2 \oplus c_3$ associated with x, but it does not matter). The left-half property of a Λ-set requires that $L_2^{(x)}$ should be constant for the 256 plaintexts; however, obviously $L_2^{(x)}$ does not necessarily meet this property, because $\mathrm{S}_1(x \oplus c_1 \oplus c_2 \oplus c_3 \oplus K_{2,1}) \oplus \mathrm{S}_1(x \oplus K_{2,1}^*)$ does not necessarily equal a constant under $K_{2,1}^* = K_{2,1}$. On the other hand, the guessed value $K_{2,1}^*$ may happen to equal $K_{2,1} \oplus c_1 \oplus c_2 \oplus c_3$, and at this moment we can have a Λ-set at the end of Round 2 when the required subkey guesses of K_1 are correct, but a successful attack will recover the value of $K_{2,1} \oplus c_1 \oplus c_2 \oplus c_3$, instead of $K_{2,1}$, provided that the same $\beta_4, \beta_6, \beta_7$ are used for all chosen Λ-sets. So we cannot exploit the fact that there are some overlapping bits between $K_{2,1}$ and other required subkeys in the subsequent procedure. However, Yeom et al. exploited the fact that there are some overlapping bits between $K_{2,1}$ and other required subkeys, and they mentioned in [23] that "One byte of the 8th round key is guessed in the second round" for their 9-round Camellia-256 attack, so this is not correct.

Therefore, there exists a flaw in the 9-round Camellia-256 attack. Likewise, a similar flaw exists in Yeom et al.'s square attacks on 6, 7, 8 and 10-round Camellia-256 with FL/FL^{-1} functions given in [23, 24].

Correct Approaches. Following the above discussions, we observe that a correct approach for obtaining the 256 plaintexts is to guess for $K_{2,1} \oplus c_1 \oplus c_2 \oplus c_3$ (in this particular example, but similar for others), not for $K_{2,1}$, that is, $K_{2,1}^*$ should denote a guess for $K_{2,1} \oplus c_1 \oplus c_2 \oplus c_3$, (Note that certain constants, namely $\beta_4, \beta_6, \beta_7$ in this particular example, should be fixed for all Λ-sets used, so that c_1, c_2, c_3 are the same for these Λ-sets). Now, if $K_{2,1}^* = K_{2,1} \oplus c_1 \oplus c_2 \oplus c_3$, then we will definitely get a Λ-set at the end of Round 2 (when the subkey guess of the required bytes of K_1 is correct). Of course, a straightforward correcting approach is to guess the whole 8 bytes of K_1, plus $K_{2,1}$. Generally speaking, the first approach implies that the resulting corrected attack will have a time complexity no smaller than that of the uncorrected attack, since the information for $K_{2,1}$ becomes unavailable and those required subkey bits from the rounds after the square distinguisher that overlap part or all of $K_{2,1}$ need to be guessed; while the second approach implies that the resulting corrected attack will have a greater data complexity, for there are more subkeys to guess, but it does not necessarily result in a corrected attack with a larger time complexity than the

corrected attack using the first approach, due to the fact some required subkey bits from the rounds after the square distinguisher may have been guessed when generating the plaintexts.

If we follow the first approach mentioned above to correct Yeom et al.'s 9-round Camellia-256 attack, we have to guess for the "One byte of the 8th round key". Roughly, the resulting corrected 9-round Camellia-256 attack has a time complexity of 2^8 times of the time complexity of $2^{202.2}$ for the uncorrected 9-round Camellia-256 attack, that is $2^{210.2}$. Similarly, the resulting corrected 10-round Camellia-256 attack has a time complexity of 2^8 times of the time complexity of $2^{246.3}$ for the uncorrected 10-round Camellia-256 attack from [24], i.e., $2^{254.3}$. (Yeom et al. wrote "Guess additional 59 bits of 7 and 8 round subkeys" in Step 3(a) of the attack in [24], however, the correct number seems to be 58.)

Fortunately, the corrected attacks still break the same numbers of rounds of Camellia-256 faster than exhaustive key search. However, it is not so lucky for Duo et al.'s certain attack(s) [8, 9].

3.2 On Duo et al.'s Square and Collision Attacks

Following Yeom et al.'s and Wu et al.'s work [21, 23], in 2005 and 2007 Duo et al. [8, 9] presented new square/collision attacks on Camellia.

In those attacks with 2 (or more) rounds appended before a square/collision distinguisher, Duo et al. used an approach similar to Yeom et al.'s to obtain the plaintexts, so a similar flaw exists in these attacks; and furthermore, since they exploited in some attacks the fact that there are some overlapping bits between K_2 (or respectively the corresponding round subkey) and other required subkeys, these attacks' complexities might be incorrect and should be reevaluated.

In particular, a notable result from [9] is a square attack on 12-round Camellia-256 without FL/FL^{-1} functions under a chosen-ciphertext attack scenario, where a distinguisher was used in the decryption direction. Duo et al. appended 2 rounds before the distinguisher and 5 rounds after the distinguisher, and guessed $(K_{12,1}, K_{12,2}, K_{12,3}, K_{12,5}, K_{12,8}, K_{11,1})$ to choose required ciphertexts. They used the fact that 46 bits of $(K_3, K_{4,1}, K_{4,4}, K_{4,5}, K_{4,6}, K_{4,7})$ can be known from $(K_1, K_2, K_{11,1}, K_{12,1}, K_{12,2}, K_{12,3}, K_{12,5}, K_{12,8})$; however, we now know that this is a flaw, and we should guess for a secret constant in connection with $K_{11,1}$. As a consequence, we should guess 8 more bits of K_3, but nevertheless their attack has a time complexity of $2^{248.6}$ 12-round Camellia-256 encryptions (it was claimed to be $2^{249.6}$ in [9], but we find that the correct value should be $2^{248.6}$, for only 42 bits of K_3 are unknown, rather than 43), the resulting corrected attack would have a time complexity of about $2^8 \times 2^{248.6} = 2^{256.6}$ 12-round Camellia-256 encryptions, slower than exhaustive key search, so it is invalid and cannot break the 12-round Camellia-256 without further improvement.

3.3 Notes

In 2002, Hatano et al. [10] presented a higher-order differential attack on 11-round Camellia-256 with/without FL/FL^{-1} functions, where a 16-th order dif-

ferential distinguisher was used. The distinguisher took as input a set of 2^{16} 16-byte values with the 9 and 10-th byte positions taking all the possible values in $\{0,1\}^8$ each and the other 14 bytes fixed. The attacked 11 rounds involved 2 rounds immediately before the distinguisher, and they used an approach similar to Yeom et al.'s, which they called the "-2R elimination technique", to obtain 2^{16} plaintexts for every guess of the 9 required unknown subkey bytes $(K_{1,1}, K_{1,2}, \cdots, K_{1,6}, K_{1,8}, K_{2,1}, K_{2,2})$ from the 2 rounds immediately before the distinguisher. Likewise, we can learn: Formally, $(K_{2,1}, K_{2,2})$ should be $(K_{2,1} \oplus c_1, K_{2,2} \oplus c_2)$, where c_1, c_2 are some unknown constants; similar for all the other attacks using the -2R elimination technique. Hatano et al. did not use the fact that there are some overlapping bits among the subkey bytes required to guess in their attacks, hence when we interpret $(K_{2,1}, K_{2,2})$ as $(K_{2,1} \oplus c_1, K_{2,2} \oplus c_2)$, the resulting attacks' complexities remain invariant. Note that in Hatano et al.'s attacks, the guessed $(K^*_{2,1}, K^*_{2,2})$ sometimes happens to equal $(K_{2,1} \oplus c_1, K_{2,2} \oplus c_2)$, and at this moment when all the other subkey guesses are correct, we can have a set of satisfying 2^{16} values at the end of Round 2, but a successful attack will recover the value for $(K_{2,1} \oplus c_1, K_{2,2} \oplus c_2)$, not $(K_{2,1}, K_{2,2})$, and it requires an additional constraint on plaintexts to keep the same (c_1, c_2).

In 2004, Wu et al. [21] presented a collision attack on 9-round Camellia-128 without FL/FL^{-1} functions, where they used a 4-round collision distinguisher. The distinguisher took as input a pair of 16-byte values with the 9-th byte position taking different values in $\{0,1\}^8$ and the other 15 bytes fixed. In Step 1 of the attack, Wu et al. suggested choosing 3 plaintexts for every guess of $(K_1, K_{2,1}, K_{2,2}, K_{2,3}, K_{2,5}, K_{2,8}, K_{3,1})$ (such that their intermediate values at the output of Round 3 could be used by the distinguisher). Wu et al. did not give how to choose these plaintexts; here we would like to suggest guessing for $K_{3,1} \oplus c_1$ to avoid any confusion, where c_1 is some constant (as discussed above). Since the key bits involved in $K_{3,1}$ do not appear in the required subkey bytes from the rounds following the distinguisher, the attack complexity remains unchanged. (Note that in Wu et al.'s attack, when the guessed $K^*_{3,1}$ happens to equal $K_{3,1} \oplus c_1$ and all the other subkey guesses are correct, we can have 3 satisfying values at the output of Round 3, but a successful attack will recover the value for $K_{3,1} \oplus c_1$, not $K_{3,1}$, and it requires an additional constraint on plaintexts to keep the same c_1.) This problem does not exist in the other attacks described in [21], for those attacks involve only one round before the distinguishers used.

4 Impossible Differential Attacks on Reduced Camellia

Impossible differential cryptanalysis was proposed independently by Knudsen [12] in 1998 and Biham et al. [2] in 1999, as a special case of differential cryptanalysis [3]; it is based on the use of one or more so-called impossible differentials (i.e., differentials with a zero probability), written $\alpha \nrightarrow \beta$, and it usually treats a block cipher $\mathbf{E} : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n$ as a cascade of three sub-ciphers $\mathbf{E} = \mathbf{E}^a \circ \mathbf{E}^b \circ \mathbf{E}^c$, where $\mathbf{E}^b$ denotes the rounds for which $\alpha \nrightarrow \beta$ holds, $\mathbf{E}^a$ denotes a few rounds before $\mathbf{E}^b$, and $\mathbf{E}^c$ denotes a few rounds after $\mathbf{E}^b$. Given a

guess for the subkeys used in $\mathbf{E}^a$ and $\mathbf{E}^c$, if a plaintext pair produces a difference of α just after $\mathbf{E}^a$, and its corresponding ciphertext pair produces a difference of β just before $\mathbf{E}^c$, then this guess for the subkeys must be incorrect. Thus, given a sufficient number of matching plaintext/ciphertext pairs, we can find the correct subkey by discarding all the wrong guesses. There are mainly two approaches to identify the wrong subkey guesses, as follows.

- The general approach is to guess all the unknown bits of the relevant round subkey necessary to partially encrypt (respectively decrypt) a candidate plaintext (respectively ciphertext) pair through the round, and finally the attacker can check whether the pair could produce an expected difference just after (respectively before) the round. In 2008, Lu et al. [16] introduced the early abort technique for impossible differential cryptanalysis, and its main idea is to partially determine whether a candidate pair is useful by guessing only a small fraction of the unknown required subkey bits of a relevant round at a time, instead of guessing all of them at once. Since some invalid pairs can be discarded before the next guess for a different fraction of the required round subkey bits, we can reduce the computational workload for an attack, and even more importantly, we may break more rounds of a cipher.
- Different from the above approach, the other approach, as used in [3] for differential cryptanalysis, is to precompute a table that contains all possible input pairs for the round function as well as their output differences under every possible value of the required round subkey bits; finally, given a candidate pair the attacker can find out the subkey(s) under which the candidate pair produces the expected output difference, by looking up the precomputation table. By contrast, this approach has in general a relatively smaller time complexity, but it requires the attacker to store the precomputation table and store the possible values for the required round subkeys in a list. It is possible in some circumstances, as employed in [15], to improve this approach by using the idea from the early abort technique.

In this section, by taking advantage of the early abort technique and a few observations on the key schedule of Camellia, we use Chen et al.'s 6-round impossible differentials [5] to break 10-round Camellia-128 with FL/FL^{-1} functions and 11-round Camellia-192 with FL/FL^{-1} functions, and finally use Wu et al.'s 8-round impossible differentials [22] to break 14-round Camellia-192 without FL/FL^{-1} functions and 16-round Camellia-256 without FL/FL^{-1} functions.

4.1 Impossible Differentials of Camellia

In 2007, Wu et al. [22] gave the following 8-round impossible differentials for Camellia without the FL/FL^{-1} functions: $(0,0,0,0,0,0,0,0,a,0,0,0,0,0,0,0) \nrightarrow (0,h,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$, where a and h are any two nonzero bytes. See [22] for more detail of the 8-round impossible differentials.

Most recently, Chen et al. [5] described the following 6-round impossible differentials for Camellia with the FL/FL^{-1} functions in the exact middle (i.e.,

between the third and fourth rounds): $(0,0,0,0,0,0,0,0,0,a,0,0,0,0,0,0) \nrightarrow (0,h,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$, where a and h are any two nonzero bytes. See [5] for a proof of the 6-round impossible differentials.

4.2 Attacking 10-Round Camellia-128 with FL/FL^{-1} Functions

First, similarly to Property 1-2 described in [16], we give the following property for Camellia, and it allows us to use the early abort technique in a couple of rounds. Its proof is similar to that for Property 1-2 in [16].

Property 1 *Suppose $(L_{i-1}||R_{i-1})$ and $(L'_{i-1}||R'_{i-1})$ are a pair of inputs of the i-th round of Camellia, and $(L_i||R_i)$ and $(L'_i||R'_i)$ are the pair of the corresponding outputs of the i-th round. Then the following results hold, where g is an 8-bit nonzero value and x is an 8-bit variable.*

1. *If $(L_{i-1}||R_{i-1}) \oplus (L'_{i-1}||R'_{i-1}) = (0,g,g,g,g,g,0,0,\star,\star,\star,\star,\star,\star,\star,\star)$ and $(L_i||R_i) \oplus (L'_i||R'_i) = (0,x,0,0,0,0,0,0,g,g,g,g,g,g,0,0)$, then there is one and only one value of x such that $\mathbf{P}^{-1}(R_{i-1} \oplus R'_{i-1} \oplus (0,x,0,0,0,0,0,0))$ has the form $(0,\star,\star,\star,\star,\star,0,0)$.*
2. *If $(L_{i-1}||R_{i-1}) \oplus (L'_{i-1}||R'_{i-1}) = (0,g,g,g,g,g,0,0,0,x,0,0,0,0,0,0)$ and $(L_i||R_i) \oplus (L'_i||R'_i) = (\star,\star,\star,\star,\star,\star,\star,\star,0,g,g,g,g,g,0,0)$, then there is one and only one value of x such that $\mathbf{P}^{-1}(L_i \oplus L'_i \oplus (0,x,0,0,0,0,0,0))$ has the form $(0,\star,\star,\star,\star,\star,0,0)$.*

Next we have the following property for Camellia-128.

Property 2 *For Camellia-128, if K_L is represented as a bit string $K_L = (k_1,k_2,\cdots,k_{128})$ and K_A is represented as a bit string $K_A = (k'_1,k'_2,\cdots,k'_{128})$, then $K_8 = (k_{110},\cdots,k_{128},k_1,\cdots,k_{45}), K_{9,2} = (k'_{54},\cdots,k'_{61}), K_{16,2} = (k'_{39},\cdots,k'_{46}), K_{17} = (k_{112},\cdots,k_{128},k_1,\cdots,k_{47})$.*

As a result, we can now use Chen et al.'s 6-round impossible differentials to break 10-round Camellia-128 with $\mathbf{FL}/\mathbf{FL}^{-1}$ functions; we attack Rounds 8 to 17 and use the 6-round impossible differentials from Rounds 10 to 15. Fig. 2-(a) illustrates the attack, where $m = 7$ (excluding the dashed round). An impossible differential attack is usually conducted in the order of checking ciphertext pairs first and finally plaintext pairs, or in the reverse order, however, we noted that Lu et al. [16] mentioned that it might be improved by using an optimised order (e.g., checking ciphertext pairs first, then plaintext pairs, next ciphertext pairs, and so on), and such an example was given in Chapter 10 of [14].

1. Choose 2^{30} structures $\mathcal{S}_i$, $(1 \leqslant i \leqslant 2^{30})$, where a structure $\mathcal{S}_i$ is defined to be a set of 2^{88} plaintexts $P^{(i,j)} = (L_7^{(i,j)}, R_7^{(i,j)})$ with $L_7^{(i,j)} = (\alpha_1^{(i)}, x_1^{(i,j)}, x_2^{(i,j)}, x_3^{(i,j)}, x_4^{(i,j)}, x_5^{(i,j)}, \alpha_2^{(i)}, \alpha_3^{(i)})$ and $R_7^{(i,j)} = \mathbf{P}(y_1^{(i,j)}, y_2^{(i,j)}, y_3^{(i,j)}, y_4^{(i,j)}, y_5^{(i,j)}, y_6^{(i,j)}, \beta_1^{(i)}, \beta_2^{(i)})$, here $\alpha_1^{(i)}, \alpha_2^{(i)}, \alpha_3^{(i)}, \beta_1^{(i)}, \beta_2^{(i)}$ are randomly chosen 8-bit constants, and $x_1^{(i,j)}, \cdots, x_5^{(i,j)}, y_1^{(i,j)}, \cdots, y_6^{(i,j)}$ take all the possible values in $\{0,1\}^8$, $(j = 1,2,\cdots,2^{88})$. In a chosen-plaintext attack scenario, obtain the ciphertexts for the 2^{30} structures of 2^{88} plaintexts, and we denote by $C^{(i,j)} = (L_{17}^{(i,j)}, R_{17}^{(i,j)})$ the ciphertext for plaintext $P^{(i,j)}$.

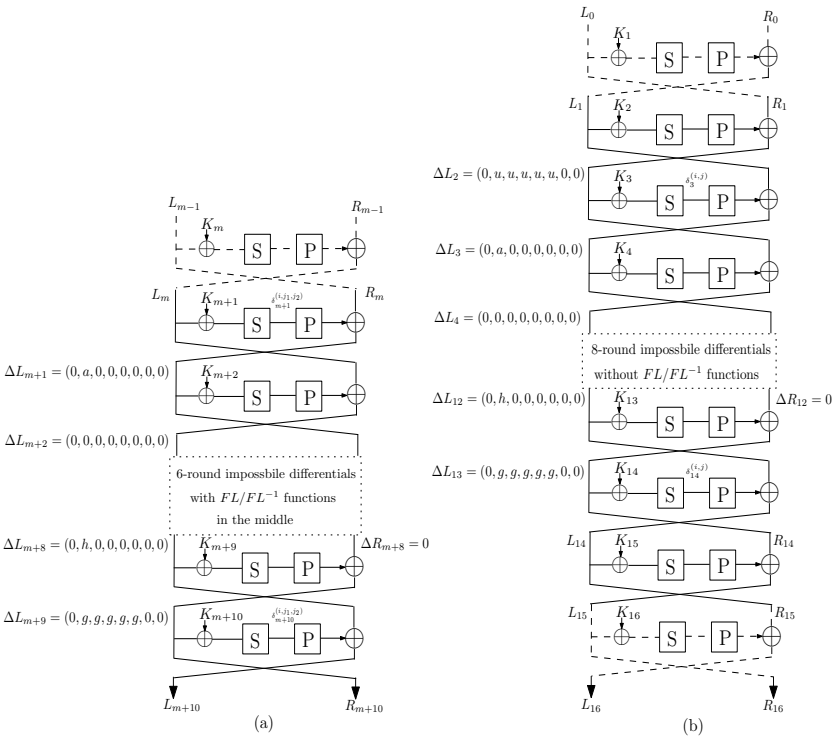

Fig. 2. Impossible differential attacks on reduced Camellia

2. For each structure $\mathcal{S}_i$, insert the plaintext-ciphertext pairs $(P^{(i,j)}, C^{(i,j)})$ into a hash table $\mathcal{L}_i$ indexed by the following thirteen 8-bit values: the XOR of the 2-nd and 3-rd bytes of $L_7^{(i,j)}$, the XOR of the 2-nd and 4-th bytes of $L_7^{(i,j)}$, the XOR of the 2-nd and 5-th bytes of $L_7^{(i,j)}$, the XOR of the 2-nd and 6-th bytes of $L_7^{(i,j)}$, the 7-th and 8-th bytes of $\mathbf{P}^{-1}(L_{17}^{(i,j)})$, the 1-st, 7-th and 8-th bytes of $R_{17}^{(i,j)}$, the XOR of the 2-nd and 3-rd bytes of $R_{17}^{(i,j)}$, the XOR of the 2-nd and 4-th bytes of $R_{17}^{(i,j)}$, the XOR of the 2-nd and 5-th bytes of $R_{17}^{(i,j)}$, and the XOR of the 2-nd and 6-th bytes of $R_{17}^{(i,j)}$.
3. We apply the early abort technique in Round 8. First, guess $(K_{8,2}, K_{8,3}, K_{8,4}, K_{8,5}, K_{8,6})$. For each quartet $((X_7^{(i,j_1)}, X_{17}^{(i,j_1)}), (X_7^{(i,j_2)}, X_{17}^{(i,j_2)}))$ from the same entry of $\mathcal{L}_i$ $(j_1, j_2 = 1, 2, \cdots, 2^{88}, j_1 \neq j_2)$, by Property 1-1 there is only one value of a such that $\mathbf{P}^{-1}(R_7^{(i,j_1)} \oplus R_7^{(i,j_2)} \oplus (0, a, 0, 0, 0, 0, 0, 0))$ has the form $(0, \star, \star, \star, \star, \star, 0, 0)$; we denote by $\delta_8^{(i,j_1,j_2)}$ the value $\mathbf{P}^{-1}(R_7^{(i,j_1)} \oplus R_7^{(i,j_2)} \oplus (0, a, 0, 0, 0, 0, 0, 0))$ with the form $(0, \star, \star, \star, \star, \star, 0, 0)$. Finally, for $l = 2, 3, \cdots, 6$, decrypt every quartet through the l-th S-box in Round 8 and check whether the obtained values have a difference equal to the l-th byte

of $\delta_8^{(i,j_1,j_2)}$. If the quartet does not meet the condition in any one of the five iterations, then discard the quartet immediately and try another quartet; keep the quartet only if it meets all the five conditions.

4. Compute $(K_{17,2}, K_{17,3}, K_{17,4}, K_{17,5}, K_{17,6})$ from the guessed $(K_{8,2}, K_{8,3}, K_{8,4}, K_{8,5}, K_{8,6})$ by Property 2. Then, apply the early abort technique in Round 17: For every remaining quartet $((X_7^{(i,j_1)}, X_{17}^{(i,j_1)}), (X_7^{(i,j_2)}, X_{17}^{(i,j_2)}))$, by Property 1-2 there is only one value of h such that $\mathbf{P}^{-1}(L_{17}^{(i,j_1)} \oplus L_{17}^{(i,j_2)} \oplus (0, h, 0, 0, 0, 0, 0, 0))$ has the form $(0, \star, \star, \star, \star, \star, 0, 0)$; we denote by $\delta_{17}^{(i,j_1,j_2)}$ the value $\mathbf{P}^{-1}(L_{17}^{(i,j_1)} \oplus L_{17}^{(i,j_2)} \oplus (0, h, 0, 0, 0, 0, 0, 0))$ with the form $(0, \star, \star, \star, \star, \star, 0, 0)$, and repeat a process similar to the one in Round 8.
5. By Property 2, the round subkeys K_8 and K_{17} only involve 68 bits of K_L; and we have guessed 40 bits of them. Subsequently, we guess the unknown 28 bits of (K_8, K_{17}), and partially encrypt/decrypt every remaining quartet $((X_7^{(i,j_1)}, X_{17}^{(i,j_1)}), (X_7^{(i,j_2)}, X_{17}^{(i,j_2)}))$ to get its corresponding values just before Rounds 9 and 17; we denote them by $((X_8^{(i,j_1)}, X_{16}^{(i,j_1)}), (X_8^{(i,j_2)}, X_{16}^{(i,j_2)}))$.
6. Guess $K_{9,2}$, and keep only the quartets $((X_8^{(i,j_1)}, X_{16}^{(i,j_1)}), (X_8^{(i,j_2)}, X_{16}^{(i,j_2)}))$ which produce a difference equal to $u^{(i,j_1,j_2)}$ (i.e., the 2-nd byte of $L_7^{(i,j_1)} \oplus L_7^{(i,j_2)}$) after the 2-nd S-box in Round 9.
7. Guess $K_{16,2}$, and check whether there is a remaining quartet $((X_8^{(i,j_1)}, X_{16}^{(i,j_1)}), (X_8^{(i,j_2)}, X_{16}^{(i,j_2)}))$ which produces the difference $g^{(i,j_1,j_2)}$ (i.e., the 2-nd byte of $R_{17}^{(i,j_1)} \oplus R_{17}^{(i,j_2)}$) after the 2-nd S-box in Round 16. If so, discard the 84-bit subkey guess $(K_8, K_{9,2}, K_{16,2}, K_{17})$, and repeat Steps 3–7 under a different subkey guess; otherwise, execute Step 2 with another structure of plaintexts. If no quartet meets the condition, then the guessed value for $(K_8, K_{9,2}, K_{16,2}, K_{17})$ is very likely to be correct, and we can find the 128-bit user key by a key exhaustion.

The attack requires 2^{118} chosen plaintexts. Any two values $(X_7^{(i,j_1)}, X_{17}^{(i,j_1)})$ and $(X_7^{(i,j_2)}, X_{17}^{(i,j_2)})$ from the same entry of $\mathcal{L}_i$ meet the following four conditions: (I) $\mathbf{P}^{-1}(R_7^{(i,j_1)} \oplus R_7^{(i,j_2)}) = (\star, \star, \star, \star, \star, \star, 0, 0)$; (II) $L_7^{(i,j_1)} \oplus L_7^{(i,j_2)} = (0, u^{(i,j_1,j_2)}, u^{(i,j_1,j_2)}, u^{(i,j_1,j_2)}, u^{(i,j_1,j_2)}, u^{(i,j_1,j_2)}, 0, 0)$, where $u^{(i,j_1,j_2)}$ is an indeterminate 8-bit value; (III) $\mathbf{P}^{-1}(L_{17}^{(i,j_1)} \oplus L_{17}^{(i,j_2)}) = (\star, \star, \star, \star, \star, \star, 0, 0)$; and (IV) $R_{17}^{(i,j_1)} \oplus R_{17}^{(i,j_2)} = (0, g^{(i,j_1,j_2)}, g^{(i,j_1,j_2)}, g^{(i,j_1,j_2)}, g^{(i,j_1,j_2)}, g^{(i,j_1,j_2)}, 0, 0)$, where $g^{(i,j_1,j_2)}$ is an indeterminate 8-bit value. Observe that the first condition and part of the second condition have been guaranteed by the pattern of the plaintexts. Hence, it is expected that there are $2^{30} \times \binom{2^{88}}{2} \times 2^{-8\times 13} \approx 2^{101}$ quartets $((X_7^{(i,j_1)}, X_{17}^{(i,j_1)}), (X_7^{(i,j_2)}, X_{17}^{(i,j_2)}))$ meeting the above four conditions. It takes $2^{30} \times 2^{88} = 2^{118}$ memory accesses to obtain all these quartets.

There is an 8-bit filtering condition in every iteration of Steps 3–4 and each of Steps 6–7, and there is no filtering condition in Step 5. Thus it is expected that there remain about $2^{101-8\times 5} = 2^{61}$ quartets $((X_7^{(i,j_1)}, X_{17}^{(i,j_1)}), (X_7^{(i,j_2)}, X_{17}^{(i,j_2)}))$ after Step 3 for every guess of $(K_{8,2}, K_{8,3}, K_{8,4}, K_{8,5}, K_{8,6})$, about $2^{61-8\times 5} = 2^{21}$ quartets $((X_7^{(i,j_1)}, X_{17}^{(i,j_1)}), (X_7^{(i,j_2)}, X_{17}^{(i,j_2)}))$ after Step 4 for every guess of

$(K_{8,2}, K_{8,3}, K_{8,4}, K_{8,5}, K_{8,6}, K_{17,2}, K_{17,3}, K_{17,4}, K_{17,5}, K_{17,6})$, and about $2^{21-8} = 2^{13}$ quartets $((X_8^{(i,j_1)}, X_{16}^{(i,j_1)}), (X_8^{(i,j_2)}, X_{16}^{(i,j_2)}))$ after Step 6 for every guess of $(K_8, K_{9,2}, K_{17})$. In Step 7, a wrong subkey guess for $(K_8, K_{9,2}, K_{16,2}, K_{17})$ will remain with a probability of $1-2^{-8}$ after testing a quartet $((X_8^{(i,j_1)}, X_{16}^{(i,j_1)}), (X_8^{(i,j_2)}, X_{16}^{(i,j_2)}))$, and thus it is expected that there remain $2^{84} \times (1-2^{-8})^{2^{13}} \approx 2^{37.84}$ possible values for $(K_8, K_{9,2}, K_{16,2}, K_{17})$. Since we obtain 68 bits of K_L, a key exhaustion will take $2^{37.84} \times 2^{60} = 2^{97.84}$ 10-round Camellia-128 encryptions.

Step 3 has a time complexity of $\sum_{i=0}^{4}(2 \times 2^{101-8\times i} \times 2^{8+8\times i} \times \frac{1}{10} \times \frac{1}{8}) = 2^{106}$ 10-round Camellia-128 encryptions. Step 4 has a time complexity of $\sum_{i=0}^{4}(2 \times 2^{61-8\times i} \times 2^{40} \times \frac{1}{10} \times \frac{1}{8}) \approx 2^{95.7}$ 10-round Camellia-128 encryptions. Step 5 has a time complexity of $2 \times 2 \times 2^{21} \times 2^{40+28} \times \frac{1}{10} \times \frac{3}{8} \approx 2^{86.3}$ 10-round Camellia-128 encryptions. Step 6 has a time complexity of $2 \times 2^{21} \times 2^{68+8} \times \frac{1}{10} \times \frac{1}{8} \approx 2^{70.7}$ 10-round Camellia-128 encryptions. Step 7 has a time complexity of about $2 \times 2^{84} \times [1 + (1-2^{-8}) + \cdots + (1-2^{-8})^{2^{13}}] \times \frac{1}{10} \times \frac{1}{8} \approx 2^{86.7}$ 10-round Camellia-128 encryptions. Therefore, the attack has a total time complexity of approximately 2^{118} 10-round Camellia-128 encryptions. The memory of the attack is dominated by the hash table $\mathcal{L}_i$ when making a list for all possible 2^{84} subkey values and using $\mathcal{L}_i$ repetitively for every structure, which is $2^{88} \times 16 \times 2 = 2^{93}$ bytes.

Note that we do not have to use the early abort technique in Round 17, and a straightforward process will take $2 \times 2^{61} \times 2^{40} \times \frac{1}{10} \times \frac{5}{8} \approx 2^{98}$ 10-round Camellia-128 encryptions, which is negligible compared with the time complexity of 2^{106} in Step 3. We can also perform an early abort more efficiently by considering Rounds 8 and 17 together: After guessing two consecutive subkey bytes in Round 8, we can deduce one subkey byte in Round 17 by Property 2, thus we can subsequently check the corresponding S-box in Round 17 (without guessing the relevant subkey byte) and discard some invalid quartets before guessing another subkey byte in Round 8; by this way Steps 3 and 4 will have smaller time complexity. We can similarly break Rounds 2 to 11 with about the same complexity as the above 10-round Camellia-128 attack.

4.3 Attacking 11-Round Camellia-192 with FL/FL^{-1} Functions

In [5] Chen et al. suggested an attack on 10-round Camellia-192 with $\mathbf{FL}/\mathbf{FL}^{-1}$ functions but without whitening keys, which has a time complexity of 2^{144} encryptions. We find that 11-round Camellia-192 with $\mathbf{FL}/\mathbf{FL}^{-1}$ functions (but without whitening keys) is breakable using Chen et al.'s 6-round impossible differentials, here the attacked 11 rounds are from Rounds 13 to 23 and the 6-round impossible differentials are used from Rounds 16 to 21. Fig. 2-(a) illustrates the attack, where $m = 13$ (including the dashed round).

We choose 2^6 structures $\mathcal{S}_i$, $(1 \leqslant i \leqslant 2^6)$, where a structure $\mathcal{S}_i$ is defined to be a set of 2^{112} plaintexts $P^{(i,j)} = (L_{12}^{(i,j)}, R_{12}^{(i,j)})$ with $L_{12}^{(i,j)} = \mathbf{P}(x_1^{(i,j)}, x_2^{(i,j)}, x_3^{(i,j)}, x_4^{(i,j)}, x_5^{(i,j)}, x_6^{(i,j)}, \alpha_1^{(i)}, \alpha_2^{(i)})$ and $R_7^{(i,j)} = (y_1^{(i,j)}, y_2^{(i,j)}, y_3^{(i,j)}, y_4^{(i,j)}, y_5^{(i,j)}, y_6^{(i,j)}, y_7^{(i,j)}, y_8^{(i,j)})$, here $\alpha_1^{(i)}, \alpha_2^{(i)}$ are randomly chosen 8-bit constants, and $x_1^{(i,j)}, \cdots, x_6^{(i,j)}, y_1^{(i,j)}, \cdots, y_8^{(i,j)}$ take all the possible values in $\{0,1\}^8$, $(j = 1, 2, \cdots, 2^{112})$.

We denote by $C^{(i,j)} = (L_{23}^{(i,j)}, R_{23}^{(i,j)})$ the ciphertext for plaintext $P^{(i,j)}$. Then, we store the plaintext-ciphertext pairs $(P^{(i,j)}, C^{(i,j)})$ into a hash table $\mathcal{L}_i$ indexed by the following nine 8-bit values: the 7-th and 8-th bytes of $\mathbf{P}^{-1}(L_{23}^{(i,j)})$, the 1-st, 7-th and 8-th bytes of $R_{23}^{(i,j)}$, the XOR of the 2-nd and 3-rd bytes of $R_{23}^{(i,j)}$, the XOR of the 2-nd and 4-th bytes of $R_{23}^{(i,j)}$, the XOR of the 2-nd and 5-th bytes of $R_{23}^{(i,j)}$, and the XOR of the 2-nd and 6-th bytes of $R_{23}^{(i,j)}$. Any two values $(X_{12}^{(i,j_1)}, X_{23}^{(i,j_1)})$ and $(X_{12}^{(i,j_2)}, X_{23}^{(i,j_2)})$ from the same entry of $\mathcal{L}_i$ meet the following two conditions $(j_1, j_2 = 1, 2, \cdots, 2^{112}, j_1 \neq j_2)$: (I) $\mathbf{P}^{-1}(L_{23}^{(i,j_1)} \oplus L_{23}^{(i,j_2)}) = (\star, \star, \star, \star, \star, \star, 0, 0)$; and (II) $R_{23}^{(i,j_1)} \oplus R_{23}^{(i,j_2)} = (0, g^{(i,j_1,j_2)}, g^{(i,j_1,j_2)}, g^{(i,j_1,j_2)}, g^{(i,j_1,j_2)}, g^{(i,j_1,j_2)}, 0, 0)$, where $g^{(i,j_1,j_2)}$ is an indeterminate 8-bit value. Hence, it is expected that there are satisfying $2^6 \times \binom{2^{112}}{2} \times 2^{-8\times 9} \approx 2^{157}$ quartets $((X_{12}^{(i,j_1)}, X_{23}^{(i,j_1)}), (X_{12}^{(i,j_2)}, X_{23}^{(i,j_2)}))$. It takes $2^6 \times 2^{112} = 2^{118}$ memory accesses to obtain these satisfying quartets.

We apply the early abort technique in Round 13 in a way used in Chen et al.'s 11-round Camellia-256 attack presented in [5], that is for every quartet $((X_{12}^{(i,j_1)}, X_{23}^{(i,j_1)}), (X_{12}^{(i,j_2)}, X_{23}^{(i,j_2)}))$ we compute $\delta_{13}^{(i,j_1,j_2)} = \mathbf{P}^{-1}(R_{12}^{(i,j_1)} \oplus R_{12}^{(i,j_2)})$ and then check the 1-st, 3-rd, 4-th, $\cdots$, and 8-th S-boxes one by one. This is because $\mathbf{P}^{-1}(R_{12}^{(i,j_1)} \oplus R_{12}^{(i,j_2)} \oplus (0, u, u, u, u, u, 0, 0)) = \mathbf{P}^{-1}(R_{12}^{(i,j_1)} \oplus R_{12}^{(i,j_2)}) \oplus (0, u, 0, 0, 0, 0, 0, 0)$ for an 8-bit value u. This is an 8-bit filtering condition in every check, and thus there remain about $2^{157-8\times 7} \approx 2^{101}$ quartets $((X_{12}^{(i,j_1)}, X_{23}^{(i,j_1)}), (X_{12}^{(i,j_2)}, X_{23}^{(i,j_2)}))$ for every guess of $(K_{13,1}, K_{13,3}, K_{13,4}, K_{13,5}, K_{13,6}, K_{13,7}, K_{13,8})$. Finally, we guess $K_{13,2}$ to get the value just after Round 13. This process has a time complexity of $\sum_{i=0}^{6}(2\times 2^{157-8\times i} \times 2^{8+8\times i} \times \frac{1}{11} \times \frac{1}{8}) + 2 \times 2^{101} \times 2^{64} \times \frac{1}{11} \times \frac{1}{8} \approx 2^{162.54}$ 11-round Camellia-192 encryptions.

Next, we check the remaining quartets $((X_{12}^{(i,j_1)}, X_{23}^{(i,j_1)}), (X_{12}^{(i,j_2)}, X_{23}^{(i,j_2)}))$ as in the above 10-round Camellia-128 attack. We check Rounds 14, 15, 23 and 22 sequentially; this allows us to make better use of the fact that K_{14} is known from the above guessed K_{13}. Finally, observe that 128 bits of the user key can be known from (K_{13}, K_{23}). A detailed analysis shows that the attack has a total time complexity of approximately $2^{162.54} + 2 \times 2^{101} \times 2^{64} \times \frac{1}{11} \times \frac{1}{8} + 2 \times 2^{144} \times [1 + (1 - 2^{-8}) + \cdots + (1 - 2^{-8})^{2^{13}}] \times \frac{1}{11} \times \frac{1}{8} \approx 2^{163.1}$ 11-round Camellia-192 encryptions. The memory complexity is dominated by a list of the 2^{144} possible subkey values, that is $2^{144} \times \frac{1}{8} = 2^{141}$ bytes.

Note that we can also apply the approach using a precomputation table and the early abort technique, which results in a 11-round Camellia-192 attack with a slightly smaller time complexity.

4.4 Attacking 14-Round Camellia-192 without FL/FL^{-1} Functions

A simple analysis on the key schedule reveals the following property.

Property 3 *For Camellia-192, the round subkeys K_2 and K_{15} involve only 68 bits of K_B; and if K_R is represented as a bit string $K_R = (k_1, k_2, \cdots, k_{64}, \overline{k_1}, \overline{k_2}, \cdots, \overline{k_{64}})$, then $K_3 = (k_{16}, \cdots, k_{23})||(k_{24}, \cdots, k_{31})||(k_{32}, \cdots, k_{39})||(k_{40}, \cdots, k_{47})||(k_{48}, \cdots, k_{55})||(k_{56}, \cdots, k_{63})||(k_{64}, \overline{k_1}, \cdots, \overline{k_7})||(\overline{k_8}, \cdots, \overline{k_{15}})$, $K_{4,2} = (\overline{k_{24}}, \cdots, \overline{k_{31}})$,*

$K_{13,2} = (\overline{k_5}, \cdots, \overline{k_{12}}), K_{14} = (\overline{k_{61}}, \cdots, \overline{k_{64}}, k_1, \cdots, k_4)||(k_5, \cdots, k_{12})||(k_{13}, \cdots, k_{20})$
$||(k_{21}, \cdots, k_{28})||(k_{29}, \cdots, k_{36})||(k_{37}, \cdots, k_{44})||(k_{45}, \cdots, k_{52})||(k_{53}, \cdots, k_{60})$.

By the structure of the key schedule of Camellia-192/256, given a value for (K_B, K_R), at the first glance it seems that recovering the corresponding K_L would need to try all the 2^{128} possible values of K_L. However, we find that it can be done in a more efficient way, as follows.

Property 4 *For Camellia-192/256, if a value for (K_B, K_R) is given, then the corresponding value for (K_L, K_A) can be obtained with a computational complexity of approximately 6 one-round Camellia computations.*

Proof. We follow the description on the key schedule of Camellia given in Section 2.2, that is $K_B = (X_6||Y_6)$. Thus, we have

$$Y_2 = \mathbf{F}(K_L[1-64] \oplus K_R[1-64], \theta_1) \oplus K_R[65 \sim 128], \quad (1)$$
$$X_2 = \mathbf{F}(Y_2 \oplus K_L[65-128], \theta_2) \oplus K_R[1 \sim 64]. \quad (2)$$

Given the value for (K_B, K_R), we can obtain $(X_2||Y_2)$ by reversing the last four iterations of Step 2(b) in the key schedule; this takes a time complexity of approximately 4 one-round Camellia computations. Subsequently, after having known $(X_2||Y_2)$ we can obtain $K_L[1 \sim 64]$ by reversing Eq. (1), that is $K_L[1 \sim 64] = \mathbf{F}^{-1}(Y_2 \oplus K_R[65 \sim 128], \theta_1) \oplus K_R[1 \sim 64]$, and obtain $K_L[65 \sim 128]$ by reversing Eq. (2), which is $K_L[65 \sim 128] = \mathbf{F}^{-1}(X_2 \oplus K_R[1 \sim 64], \theta_2) \oplus Y_2$; it takes approximately 2 one-round Camellia computations. Therefore, it takes approximately 6 one-round Camellia computations to obtain the corresponding K_L for a given value for (K_B, K_R). Since (X_4, Y_4) has been known during the reverse operations, the corresponding value for K_A can be readily known as $K_A = K_R \oplus (X_4, Y_4)$. The result follows. □

Subsequently, we can use Wu et al.'s 8-round impossible differentials to break 14-round Camellia-192 (without FL/FL^{-1} functions); the attacked 14 rounds are from Rounds 1 to 15, where we use the 8-round impossible differentials from Rounds 5 to 12. Different from the 10-round Camellia-128 and 11-round Camellia-192 attacks described in Sections 4.2 and 4.3, this 14-round Camellia-192 attack can work in a known-plaintext attack scenario. Fig. 2-(b) illustrates the attack (excluding the dashed rounds).

We obtain $2^{122.5}$ plaintext-ciphertext pairs $(P^{(i)}, C^{(i)}) = (L_1^{(i)}||R_1^{(i)}, L_{15}^{(i)}||R_{15}^{(i)})$ in a known-plaintext attack scenario, where $i = 1, 2, \cdots, 2^{122.5}$. Then, perform the following process.

1. Guess the 68 bits of K_B that are involved in (K_2, K_{15}), and compute K_2, K_{15}. For every plaintext-ciphertext pair $(P^{(i)}, C^{(i)})$: partially encrypt $P^{(i)}$ with K_2 to get the corresponding value just after Round 2, and we denote it by $X_2^{(i)} = (L_2^{(i)}, R_2^{(i)})$; and partially decrypt $C^{(i)}$ with K_{15} to get the corresponding value just after Round 14, and we denote it by $X_{14}^{(i)} = (L_{14}^{(i)}, R_{14}^{(i)})$.
2. Store the values $(X_2^{(i)}, X_{14}^{(i)})$ into a hash table $\mathcal{L}$ indexed by the following eighteen 8-bit values: the 1-st, 7-th and 8-th bytes of $L_2^{(i)}$, the XOR of the

2-nd and 3-rd bytes of $L_2^{(i)}$, the XOR of the 2-nd and 4-th bytes of $L_2^{(i)}$, the XOR of the 2-nd and 5-th bytes of $L_2^{(i)}$, the XOR of the 2-nd and 6-th bytes of $L_2^{(i)}$, the 7-th and 8-th bytes of $\mathbf{P}^{-1}(R_2^{(i)})$, the 7-th and 8-th bytes of $\mathbf{P}^{-1}(L_{14}^{(i)})$, the 1-st, 7-th and 8-th bytes of $R_{14}^{(i)}$, the XOR of the 2-nd and 3-rd bytes of $R_{14}^{(i)}$, the XOR of the 2-nd and 4-th bytes of $R_{14}^{(i)}$, the XOR of the 2-nd and 5-th bytes of $R_{14}^{(i)}$, and the XOR of the 2-nd and 6-th bytes of $R_{14}^{(i)}$.

The partial encryptions/decryptions in this process are equivalent to approximately $2^{68} \times 2^{122.5} \times \frac{2}{14} \approx 2^{187.7}$ 14-round Camellia-192 encryptions. The $2^{122.5}$ plaintexts $P^{(i)}$ yield a total of $\binom{2^{122.5}}{2} \approx 2^{244}$ plaintext pairs $(P^{(i)}, P^{(j)})$, $(i, j = 1, 2, \cdots, 2^{122.5}, j \neq i)$. Any two values $(X_2^{(i)}, X_{14}^{(i)})$ and $(X_2^{(j)}, X_{14}^{(j)})$ from the same entry of $\mathcal{L}$ meet the following four conditions: (I) $L_2^{(i)} \oplus L_2^{(j)} = (0, u^{(i,j)}, u^{(i,j)}, u^{(i,j)}, u^{(i,j)}, u^{(i,j)}, 0, 0)$, where $u^{(i,j)}$ is an indeterminate 8-bit value; (II) $\mathbf{P}^{-1}(R_2^{(i)} \oplus R_2^{(j)}) = (\star, \star, \star, \star, \star, \star, 0, 0)$; (III) $\mathbf{P}^{-1}(L_{14}^{(i)} \oplus L_{14}^{(j)}) = (\star, \star, \star, \star, \star, \star, 0, 0)$; and (IV) $R_{14}^{(i)} \oplus R_{14}^{(j)} = (0, g^{(i,j)}, g^{(i,j)}, g^{(i,j)}, g^{(i,j)}, g^{(i,j)}, 0, 0)$, where $g^{(i,j)}$ is an indeterminate 8-bit value. Hence, it is expected that for a value of the 68 bits of K_B there are $2^{244} \times 2^{-8\times 18} \approx 2^{100}$ quartets $((X_2^{(i)}, X_{14}^{(i)}), (X_2^{(j)}, X_{14}^{(j)}))$ meeting the above four conditions. For all the possible values of the 68 bits of K_B, it takes a total of $2^{68} \times 2^{122.5} = 2^{190.5}$ memory accesses to obtain these quartets; this is negligible compared with the above $2^{187.7}$ 14-round Camellia-192 encryptions.

Next, we check the quartets $((X_2^{(i)}, X_{14}^{(i)}), (X_2^{(j)}, X_{14}^{(j)}))$ as in the 10-round Camellia-128 attack given in Section 4.2. We check Rounds 3, 14, 13 and 4 sequentially; this allows us to take advantage of the fact that K_{14} is known once K_3 is guessed. By Property 3 we need to guess a total of 132 subkey bits. At the last step, since K_R (i.e., K_3) and 68 bits of K_B are known, we need to guess the remaining 60 unknown bits of K_B, then recover the corresponding value for (K_L, K_A), and finally check whether the value for (K_L, K_R, K_A, K_B) satisfies two known plaintext-ciphertext pairs. It is expected that there remain $2^{132} \times (1-2^{-8})^{2^{100-8\times 11}} \approx 2^{108.96}$ possible values for the required 132-bit subkey, and thus by Property 4, it takes $2^{108.96} \times 2^{60} \times \frac{6}{14} \approx 2^{167.74}$ 14-round Camellia-192 encryptions to obtain the possible values for (K_L, K_A), so it requires $2^{168.96} + 2^{168.96} \times 2^{-128} \approx 2^{168.96}$ trials to find the correct user key.

The time complexity of the attack is dominated by the time complexity of $2^{187.7}$ 14-round Camellia-192 encryptions for the partial encryptions and decryptions in Rounds 2 and 15. The required memory of the attack is dominated by the storage of the $2^{122.5}$ plaintext-ciphertext pairs $(P^{(i)}, C^{(i)})$ and the table $\mathcal{L}$, which is $2 \times 2 \times 2^{122.5} \times 16 = 2^{128.5}$ bytes of memory.

However, in a chosen-plaintext attack scenario, we can obtain a 14-round Camellia-192 attack with lower complexity. We choose 2^5 structures $\mathcal{S}_i$, $(1 \leqslant i \leqslant 2^5)$, where a structure $\mathcal{S}_i$ is defined to be a set of 2^{112} plaintexts $P^{(i,j)} = (L_1^{(i,j)}, R_1^{(i,j)})$ with $L_1^{(i,j)} = \mathbf{P}(x_1^{(i,j)}, x_2^{(i,j)}, x_3^{(i,j)}, x_4^{(i,j)}, x_5^{(i,j)}, x_6^{(i,j)}, \alpha_1^{(i)}, \alpha_2^{(i)})$ and $R_1^{(i,j)} = (y_1^{(i,j)}, y_2^{(i,j)}, y_3^{(i,j)}, y_4^{(i,j)}, y_5^{(i,j)}, y_6^{(i,j)}, y_7^{(i,j)}, y_8^{(i,j)})$, here $\alpha_1^{(i)}, \alpha_2^{(i)}$ are ran-

domly chosen 8-bit constants, and $x_1^{(i,j)}, x_2^{(i,j)}, \cdots, x_6^{(i,j)}, y_1^{(i,j)}, y_2^{(i,j)}, \cdots, y_8^{(i,j)}$ take all the possible values in $\{0,1\}^8$, $(j = 1, 2, \cdots, 2^{112})$. Then, we can perform the process performed at the beginning of the above 14-round Camellia-192 attack, here we store the plaintext-ciphertext pairs in the structure $\mathcal{S}_i$ to a hash table $\mathcal{L}_i$. It takes $2^{68} \times 2^{117} \times \frac{2}{14} \approx 2^{182.2}$ 14-round Camellia-192 encryptions. A structure $\mathcal{S}_i$ yields a total of $\binom{2^{112}}{2} \approx 2^{223}$ plaintext pairs $(P^{(i,j_1)}, P^{(i,j_2)})$, $(i = 1, 2, \cdots, 2^5, j_1, j_2 = 1, 2, \cdots, 2^{112}, j_1 \neq j_2)$. Any two values $(X_2^{(i,j_1)}, X_{14}^{(i,j_1)})$ and $(X_2^{(i,j_2)}, X_{14}^{(i,j_2)})$ from the same entry of $\mathcal{L}_i$ meet the following four conditions: (I) $\mathbf{P}^{-1}(R_2^{(i,j_1)} \oplus R_2^{(i,j_2)}) = (\star, \star, \star, \star, \star, \star, 0, 0)$; (II) $L_2^{(i,j_1)} \oplus L_2^{(i,j_2)} = (0, u^{(i,j_1,j_2)}, u^{(i,j_1,j_2)}, u^{(i,j_1,j_2)}, u^{(i,j_1,j_2)}, u^{(i,j_1,j_2)}, 0, 0)$, where $u^{(i,j_1,j_2)}$ is an indeterminate 8-bit value. (III) $\mathbf{P}^{-1}(L_{14}^{(i,j_1)} \oplus L_{14}^{(i,j_2)}) = (\star, \star, \star, \star, \star, \star, 0, 0)$; and (IV) $R_{14}^{(i,j_1)} \oplus R_{14}^{(i,j_2)} = (0, g^{(i,j_1,j_2)}, g^{(i,j_1,j_2)}, g^{(i,j_1,j_2)}, g^{(i,j_1,j_2)}, g^{(i,j_1,j_2)}, 0, 0)$, where $g^{(i,j_1,j_2)}$ is an indeterminate 8-bit value. Observe that the first condition has already been guaranteed by the pattern of the plaintexts. It is expected that for a guessed value of the 68 bits of K_B there are $2^5 \times 2^{223} \times 2^{-8\times16} = 2^{100}$ quartets $((X_2^{(i,j_1)}, X_{14}^{(i,j_1)}), (X_2^{(i,j_2)}, X_{14}^{(i,j_2)}))$ meeting the above four conditions. It takes $2^{68} \times 2^{117} = 2^{185}$ memory accesses to obtain the qualified quartets. In the following, we can perform in almost the same way as in the above 14-round Camellia-192 attack. This attack requires 2^{117} chosen plaintexts, and its memory complexity is dominated by the 2^{117} plaintext-ciphertext pairs $(P^{(i,j)}, C^{(i,j)})$ and the table $\mathcal{L}_i$, which is $2 \times 2^{117} \times 16 + 2 \times 2^{112} \times 16 \approx 2^{122.1}$ bytes of memory.

4.5 Attacking 16-Round Camellia-256 without FL/FL^{-1} Functions

First, the following property holds for Camellia-256.

Property 5 *For Camellia-256, the round subkeys K_1, K_2, K_{15} and K_{16} involve only the 128 bits of K_B; and if K_R is represented as a bit string $K_R = (k_1, k_2, \cdots, k_{128})$, then $K_3 = (k_{16}, \cdots, k_{23})||(k_{24}, \cdots, k_{31})||(k_{32}, \cdots, k_{39})||(k_{40}, \cdots, k_{47})||(k_{48}, \cdots, k_{55})||(k_{56}, \cdots, k_{63})||(k_{64}, \cdots, k_{71})||(k_{72}, \cdots, k_{79})$, $K_{4,2} = (k_{88}, \cdots, k_{105})$, $K_{13,2} = (k_{69}, \cdots, k_{76})$, $K_{14} = (k_{125}, \cdots, k_{128}, k_1, \cdots, k_4)||(k_5, \cdots, k_{12})||(k_{13}, \cdots, k_{20})||(k_{21}, \cdots, k_{28})||(k_{29}, \cdots, k_{36})||(k_{37}, \cdots, k_{44})||(k_{45}, \cdots, k_{52})||(k_{53}, \cdots, k_{60})$.*

Then, using Wu et al.'s 8-round impossible differentials we can attack Rounds 1 to 16 of Camellia-256 (without FL/FL^{-1} functions), where the 8-round impossible differentials are used from Rounds 5 to 12. Fig. 2-(b) illustrates the attack (including the dashed rounds).

We obtain 2^{123} plaintext-ciphertext pairs in a known-plaintext attack scenario. Then, guess K_B, compute $(K_1, K_2, K_{15}, K_{16})$ using the guessed K_B, and perform in almost the same way as in the first 14-round Camellia-192 attack in Section 4.4. By Property 5 we need to guess a total of 219 subkey bits.

The attack requires 2^{123} known plaintexts and a memory of $2 \times 2 \times 2^{123} \times 16 = 2^{129}$ bytes. At the last step, there remain about $2^{219} \times (1-2^{-8})^{2^{2\times123-1-8\times18-8\times11}} = 2^{219} \times (1-2^{-8})^{2^{13}} \approx 2^{172.92}$ possible values for the required 219-bit subkey. For these remaining subkey guesses, we still have 37 unknown bits for K_R, and thus

there are a total of $2^{172.92} \times 2^{37} = 2^{209.92}$ possible values for (K_B, K_R). By Property 4, it takes $2^{209.92} \times \frac{6}{16} \approx 2^{208.51}$ 16-round Camellia-256 encryptions to obtain the possible values for (K_L, K_A), so it requires $2^{209.92} + 2^{209.92} \times 2^{-128} \approx 2^{209.92}$ trials to find the correct user key. Similarly, the time complexity of the attack is dominated by the partial encryptions and decryptions in Rounds 1, 2, 15 and 16, which is approximately $2^{128} \times 2^{123} \times \frac{4}{16} = 2^{249}$ 16-round Camellia-256 encryptions. Note that we can similarly break the last 16 rounds (i.e., Rounds 9 to 24) with approximately the same complexity.

5 Conclusions

In this paper, we have shown a flaw in the approach used to choose plaintexts/ciphertexts in certain previously published square-like cryptanalytic results for Camellia and given correct approaches. Finally, by taking advantage of the early abort technique and a few observations on the key schedule, we have presented impossible differential attacks on 10-round Camellia-128 with FL/FL^{-1} functions, 11-round Camellia-192 with FL/FL^{-1} functions, 14-round Camellia-192 without FL/FL^{-1} functions and 16-round Camellia-256 without FL/FL^{-1} functions. These are better than any previously published cryptanalytic results for the respective versions of Camellia in terms of the numbers of attacked rounds.

Acknowledgments

The authors are very grateful to Yasuo Hatano for some discussions and to Prof. Wenling Wu, Lei Duo, Yongjin Yeom and the anonymous referees for their comments.

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.E. (eds.) SAC 2000. LNCS, vol. 2012, pp. 39–56. Springer, Heidelberg (2001)
2. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 12–23. Springer, Heidelberg (1999)
3. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. Journal of Cryptology 4(1), 3–72. Springer (1991)
4. Biryukov, A., Nikolic, I.: Automatic search for related-key differential characteristics in byte-oriented block ciphers: application to AES, Camellia, Khazad and others. In: Gilbert, H. (eds.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 322–344. Springer, Heidelberg (2010)
5. Chen, J., Jia, K., Yu, H., Wang, X.: New impossible differential attacks of reduced-round Camellia-192 and Camellia-256. In: Hawkes, P., Parampalli, U. (eds.) ACISP 2011. LNCS, vol. 6812, pp. 16–33. Springer, Heidelberg (2011)

6. Cryptography Research and Evaluatin Committees (CRYPTREC), report 2002.
7. Daemen, J., Knudsen, L.R., Rijmen, V.: The block cipher Square. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 149–165. Springer, Heidelberg (1997)
8. Duo, L., Li, C., Feng, K.: New observation on Camellia. In: Preneel, B., Tavares, S.E. (eds.) SAC 2005. LNCS, vol. 3897, pp. 51–64. Springer, Heidelberg (2006)
9. Duo, L., Li, C., Feng, K.: Square like attack on Camellia. In: Qing, S., Imai, H., Wang, G. (eds.) ICICS 2007. LNCS, vol. 4861, pp. 269–283. Springer, Heidelberg (2007)
10. Hatano, Y., Sekine, H., Kaneko, T.: Higher order differential attack of Camellia(II). In Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp.39–56. Springer, Heidelberg (2003)
11. International Standardization of Organization (ISO), International Standard – ISO/IEC 18033-3, Information technology – Security techniques – Encryption algorithms – Part 3: Block ciphers, July, 2005.
12. Knudsen, L.R.: DEAL — a 128-bit block cipher. Technical report, Department of Informatics, University of Bergen, Norway (1998).
13. Lai, X.: Higher order derivatives and differential cryptanalysis. In: Communications and Cryptography, pages 227–233, 1994. Academic Publishers.
14. Lu, J.: Cryptanalysis of block ciphers. PhD thesis, University of London, UK (2008)
15. Lu, J., Dunkelman, O., Keller, N., Kim, J.: New impossible differential attacks on AES. In: Chowdhury, D.R., Rijmen, V., Das., A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 279–293. Springer, Heidelberg (2008).
16. Lu, J., Kim, J., Keller, N., Dunkelman, O.: Improving the efficiency of impossible differential cryptanalysis of reduced Camellia and MISTY1. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 370–386. Springer, Heidelberg (2008)
17. Lu, J., Wei, Y., Kim, J., Pasalic, E.: The higher-order meet-in-the-middle attack and its application to the Camellia block cipher. (Manuscript, 2011)
18. Lu, J., Wei, Y., Pasalic, E., Fouque, P.-A.: Meet-in-the-middle attack on reduced versions of the Camellia block cipher. (Manuscript, 2011)
19. Mala, H., Shakiba, M., Dakhilalian, M., Bagherikaram, G.: New results on impossible differential cryptanalysis of reduced-round Camellia-128. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 281–294. Springer, Heidelberg (2009)
20. New European Schemes for Signatures, Integrity, and Encryption (NESSIE), final report of European project IST-1999-12324.
21. Wu, W., Feng, D., Chen, H.: Collision attack and pseudorandomness of reduced-round Camellia. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 256–270. Springer, Heidelberg (2005)
22. 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. Springer (2007)
23. Yeom, Y., Park, S., Kim, Iljun.: On the security of Camellia against the square attack. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2356, pp. 89–99. Springer, Heidelberg (2002)
24. Yeom, Y., Park, S., Kim, I.: A study of integral type cryptanalysis on Camellia. In Proceedings of the 2003 Symposium on Cryptography and Information Security, pp. 453–456 (2003)