Collision attack on reduced-round Camellia

WU Wenling & FENG Dengguo

State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences, Beijing 100080, China

Correspondence should be addressed to Wu Wenling (email:wwl@is.iscas.ac.cn)

Received May 17, 2004

Abstract Camellia is the final winner of 128-bit block cipher in NESSIE. In this paper, we construct some efficient distinguishers between 4-round Camellia and a random permutation of the blocks space. By using collision-searching techniques, the distinguishers are used to attack on 6, 7, 8 and 9 rounds of Camellia with 128-bit key and 8, 9 and 10 rounds of Camellia with 192/256-bit key. The 128-bit key of 6 rounds Camellia can be recovered with 2^{10} chosen plaintexts and 2^{15} encryptions. The 128-bit key of 7 rounds Camellia can be recovered with 2^{12} chosen plaintexts and $2^{54.5}$ encryptions. The 128-bit key of 8 rounds Camellia can be recovered with 2^{13} chosen plaintexts and $2^{112.1}$ encryptions. The 128-bit key of 9 rounds Camellia can be recovered with $2^{113.6}$ chosen plaintexts and 2^{121} encryptions. The 192/256-bit key of 8 rounds Camellia can be recovered with 2^{13} chosen plaintexts and $2^{111.1}$ encryptions. The 192/256-bit key of 9 rounds Camellia can be recovered with 2^{13} chosen plaintexts and $2^{175.6}$ encryptions. The 256-bit key of 10 rounds Camellia can be recovered with 2^{14} chosen plaintexts and $2^{239.9}$ encryptions.

Keywords: block cipher, collision attack, key, data complexity, time complexity.

DOI: 10.1360/ 03yf0293

Camellia[1] is a 128-bit block cipher which was published by NTT and Mitsubishi in 2000 and recently selected as the final selection of the NESSIE[2] project. The security of Camellia has been studied by many researchers[3—9]. The security of Camellia against higher-order differential cryptanalysis is discussed in ref. [3]. A truncated differential attack on 8-round variant of Camellia without FL/FL^{-1} functions is presented in ref. [4], requiring $2^{55.6}$ encryptions and $2^{83.6}$ chosen plaintexts. Truncated and impossible differential cryptanalysis of Camellia without FL/FL^{-1} functions is described in ref. [5]. A differential attack on 9-round Camellia without FL/FL^{-1} functions is proposed in ref. [6], requiring 2^{105} chosen plaintexts. The security of Camellia against Square attack is discussed in refs. [7, 8]. Furthermore, Yeom et al. have studied integral properties and applied them to Camellia. In this paper we present collision attacks on reduced-round variants of Camellia without FL/FL^{-1} and whitening function layers as mentioned below.

The attack on 6-round of 128-bit key Camellia is more efficient than known attacks. The complexities of the attack on 7 (8, 9, 10)-round Camellia without FL/FL^{-1} functions are less than that of previous attacks.

Section 1 briefly describes the structure of Camellia. 4-round distinguishers and properties are explained in section 2. In section 3, we show how to use the 4-round distinguishers to attack 6, 7, 8 and 9 rounds of Camellia with 128-bit key. In section 4, we describe attacks on 8, 9 and 10 rounds of Camellia with 192/256-bit key. Finally, in section 5 we conclude this paper.

1 Description of Camellia

Camellia has a 128-bit block size and supports 128, 192 and 256-bit keys. The design of Camellia is based on the Feistel structure and its number of rounds is 18(128-bit key) or 24(192/256-bit key). The FL/FL^{-1} function layer is inserted at every 6 rounds. Before the first round and after the last round, there are pre- and post-whitening layers which use bitwise exclusive-or operations with 128-bit subkeys, respectively. But we will consider Camellia without FL/FL^{-1} function layer and whitening layers and call it modified Camellia.

Let L_{r-1} and R_{r-1} be the left and the right halves of the r-th round inputs, and k_r be the r-th round subkey.

Then the Feistel structure of Camellia can be written as

$$L_r = R_{r-1} \oplus F(L_{r-1} \oplus k_r),$$
$$R_r = L_{r-1}.$$

Here F is the round function defined below.

$$F: F_2^{64} \to F_2^{64}$$
$$X_{(64)} \to Y_{(64)} = P(S(X_{64}),$$

where S and P are defined as follows.

$$S: F_2^{64} \to F_2^{64}$$
$$l_{1(8)} \| l_{2(8)} \| l_{3(8)} \| l_{4(8)} \| l_{5(8)} \| l_{6(8)} \| l_{7(8)} \| l_{8(8)} \to$$
$$l'_{1(8)} \| l'_{2(8)} \| l'_{3(8)} \| l'_{4(8)} \| l'_{5(8)} \| l'_{6(8)} \| l'_{7(8)} \| l'_{8(8)}$$
$$l'_{1(8)} = s_1(l_{1(8)}), l'_{2(8)} = s_2(l_{2(8)}), l'_{3(8)} = s_3(l_{3(8)}), l'_{4(8)} = s_4(l_{4(8)}),$$
$$l'_{5(8)} = s_2(l_{5(8)}), l'_{6(8)} = s_3(l_{6(8)}), l'_{7(8)} = s_4(l_{7(8)}), l'_{8(8)} = s_1(l_{8(8)}).$$

$$P: F_2^{64} \to F_2^{64}$$
$$Z_{1(8)} \| Z_{2(8)} \| Z_{3(8)} \| Z_{4(8)} \| Z_{5(8)} \| Z_{6(8)} \| Z_{7(8)} \| Z_{8(8)} \to$$
$$Z'_{1(8)} \| Z'_{2(8)} \| Z'_{3(8)} \| Z'_{4(8)} \| Z'_{5(8)} \| Z'_{6(8)} \| Z'_{7(8)} \| Z'_{8(8)}$$

$$
\begin{aligned}
Z_1' &= Z_1 \oplus Z_3 \oplus Z_4 \oplus Z_6 \oplus Z_7 \oplus Z_8, & Z_5' &= Z_1 \oplus Z_2 \oplus Z_6 \oplus Z_7 \oplus Z_8,\\
Z_2' &= Z_1 \oplus Z_2 \oplus Z_4 \oplus Z_5 \oplus Z_7 \oplus Z_8, & Z_6' &= Z_2 \oplus Z_3 \oplus Z_5 \oplus Z_7 \oplus Z_8,\\
Z_3' &= Z_1 \oplus Z_2 \oplus Z_3 \oplus Z_5 \oplus Z_6 \oplus Z_8, & Z_7' &= Z_3 \oplus Z_4 \oplus Z_5 \oplus Z_6 \oplus Z_8,\\
Z_4' &= Z_2 \oplus Z_3 \oplus Z_4 \oplus Z_5 \oplus Z_6 \oplus Z_7, & Z_8' &= Z_1 \oplus Z_4 \oplus Z_5 \oplus Z_6 \oplus Z_7.
\end{aligned}
$$

Below the key schedule of Camellia is briefly described. First two 128-bit variables K_L and K_R are generated from the user key. Then two 128-bit variables K_A and K_B are generated from K_L and K_R. Note that K_B is used only when the user key is of 192 or 256 bits. The round subkeys are generated by rotating K_L, K_R, K_A and K_B. Details are shown in ref. [1].

2 4-round distinguishers

Choose

$$L_0 = (\alpha_1, \alpha_2, ..., \alpha_8), \qquad R_0 = (x, \beta_2, ..., \beta_8),$$

where x take values in F_2^8, α_i and β_j are constants in F_2^8. Thus, the input of the 2nd round can be written as follows:

$$L_1 = (x \oplus \gamma_1, \gamma_2, ..., \gamma_8), \qquad R_1 = (\alpha_1, \alpha_2, ..., \alpha_8),$$

where γ_i are entirely determined by $\alpha_i (1 \le i \le 8)$, $\beta_j (2 \le j \le 8)$ and k_1, so γ_i are constants when the user key is fixed. In the 2nd round a transformation on L_1 using $F(\circ, k_2)$ is as follows:

$$L_1 = (x \oplus \gamma_1, \gamma_2, ..., \gamma_8) \xrightarrow{F(\circ, k_2)} (y \oplus \theta_1, y \oplus \theta_2, y \oplus \theta_3, \theta_4, y \oplus \theta_5, \theta_6, \theta_7, y \oplus \theta_8),$$

where $y = s_1(x \oplus \gamma_1 \oplus k_{2,1})$, $k_{2,1}$ is the first byte of k_2, θ_i are entirely determined by $\gamma_i (1 \le i \le 8)$ and k_2. Thus θ_i are constants when the user key is fixed. Therefore, the output of the 2nd round is

$$
\begin{aligned}
&L_2 = (y \oplus \omega_1, y \oplus \omega_2, y \oplus \omega_3, \omega_4, y \oplus \omega_5, \omega_6, \omega_7, y \oplus \omega_8),\\
&R_2 = L_1 = (x \oplus \gamma_1, \gamma_2, ..., \gamma_8),
\end{aligned}
$$

where $\omega_i = \theta_i \oplus \alpha_i$ are constants. In the 3rd round a transformation on L_2 using $F(\circ, k_3)$ is as follows:

$$
\begin{aligned}
L_2 &= (y \oplus \omega_1, y \oplus \omega_2, y \oplus \omega_3, \omega_4, y \oplus \omega_5, \omega_6, \omega_7, y \oplus \omega_8)\\
&\xrightarrow{F(\circ, k_3)} (z_1, z_2, ..., z_8).
\end{aligned}
$$

Thus, we have the left half of output for the 3rd round

$$L_3 = (z_1 \oplus x \oplus \gamma_1, z_2 \oplus \gamma_2, z_3 \oplus \gamma_3, ..., z_8 \oplus \gamma_8).$$

So the right half of output for the 4th round is as follows

$R_4 = L_3 = (z_1 \oplus x \oplus \gamma_1, z_2 \oplus \gamma_2, z_3 \oplus \gamma_3, ..., z_8 \oplus \gamma_8).$

Now we analyze the relations among bytes of R_4. By observing the equation $(z_1, z_2, ..., z_8) = F(L_2, k_3)$, we get the following equations

$$z_3 \oplus z_4 \oplus z_5 \oplus z_6 \oplus z_7 = s_4(\omega_7 \oplus k_{3,7}), \tag{2.1}$$

$$z_2 \oplus z_3 \oplus z_4 \oplus z_6 \oplus z_7 \oplus z_8 = s_1(y \oplus \omega_1 \oplus k_{3,1}), \tag{2.2}$$

$$z_2 \oplus z_3 \oplus z_5 \oplus z_6 \oplus z_8 = s_3(\omega_6 \oplus k_{3,6}), \tag{2.3}$$

$$z_1 \oplus z_7 \oplus z_8 = s_4(\omega_4 \oplus k_{3,4}) \oplus s_3(\omega_6 \oplus k_{3,6}), \tag{2.4}$$

$$z_3 \oplus z_4 \oplus z_5 = s_4(\omega_4 \oplus k_{3,4}) \oplus s_2(y \oplus \omega_2 \oplus k_{3,2}) \oplus s_3(\omega_6 \oplus k_{3,6}), \tag{2.5}$$

$$z_2 \oplus z_4 \oplus z_5 \oplus z_6 \oplus z_7 = s_4(\omega_4 \oplus k_{3,4}) \oplus s_3(y \oplus \omega_3 \oplus k_{3,3}) \oplus s_3(\omega_6 \oplus k_{3,6}), \tag{2.6}$$

$$z_2 \oplus z_5 = s_4(\omega_4 \oplus k_{3,4}) \oplus s_2(y \oplus \omega_5 \oplus k_{3,5}) \oplus s_3(\omega_6 \oplus k_{3,6}), \tag{2.7}$$

$$z_4 \oplus z_6 = s_4(\omega_4 \oplus k_{3,4}) \oplus s_1(y \oplus \omega_8 \oplus k_{3,8}) \oplus s_3(\omega_6 \oplus k_{3,6}). \tag{2.8}$$

Because s_1 is a permutation, $y = s_1(x \oplus \gamma_1 \oplus k_{2,1})$ differs when x takes different values. As a consequence, $s_1(y \oplus \omega_1 \oplus k_{3,1})$ will have different values. Similarly, $s_2(y \oplus \omega_2 \oplus k_{3,2})$, $s_3(y \oplus \omega_3 \oplus k_{3,3})$, $s_2(y \oplus \omega_5 \oplus k_{3,5})$ and $s_1(y \oplus \omega_8 \oplus k_{3,8})$ have the same property as $s_1(y \oplus \omega_1 \oplus k_{3,1})$. Obviously, $s_4(\omega_4 \oplus k_{3,4})$, $s_3(\omega_6 \oplus k_{3,6})$ and $s_4(\omega_7 \oplus k_{3,7})$ are constants. Thus, from the above discussion we know that $z_3 \oplus z_4 \oplus z_5 \oplus z_6 \oplus z_7$, $z_2 \oplus z_3 \oplus z_5 \oplus z_6 \oplus z_8$ and $z_1 \oplus z_7 \oplus z_8$ are constants, and $z_2 \oplus z_3 \oplus z_4 \oplus z_6 \oplus z_7 \oplus z_8$, $z_3 \oplus z_4 \oplus z_5$, $z_2 \oplus z_4 \oplus z_5 \oplus z_6 \oplus z_7$, $z_2 \oplus z_5$ and $z_4 \oplus z_6$ each will have different values when x takes different values. Therefore we get the following lemma by considering $R_4 = L_3 = (z_1 \oplus x \oplus \gamma_1, z_2 \oplus \gamma_2, z_3 \oplus \gamma_3, ..., z_8 \oplus \gamma_8)$.

Lemma. Let $P = (L_0, R_0)$ and $P^* = (L_0^*, R_0^*)$ be two plaintexts of 4-round Camellia, let $C = (L_4, R_4)$ and $C^* = (L_4^*, R_4^*)$ be the corresponding ciphertexts, and let $R_{0,i}$ denote the ith byte of R_0. If $L_0 = L_0^*$, $R_{0,1} \neq R_{0,1}^*, R_{0,j} = R_{0,j}^* (2 \leqslant j \leqslant 8)$, then R_4 and R_4^* satisfy

(1) $R_{4,3} \oplus R_{4,4} \oplus R_{4,5} \oplus R_{4,6} \oplus R_{4,7} = R_{4,3}^* \oplus R_{4,4}^* \oplus R_{4,5}^* \oplus R_{4,6}^* \oplus R_{4,7}^*$,

(2) $R_{4,2} \oplus R_{4,3} \oplus R_{4,5} \oplus R_{4,6} \oplus R_{4,8} = R_{4,2}^* \oplus R_{4,3}^* \oplus R_{4,5}^* \oplus R_{4,6}^* \oplus R_{4,8}^*$,

(3) $R_{4,2} \oplus R_{4,3} \oplus R_{4,4} \oplus R_{4,6} \oplus R_{4,7} \oplus R_{4,8} \neq R_{4,2}^{*} \oplus R_{4,3}^{*} \oplus R_{4,4}^{*} \oplus R_{4,6}^{*} \oplus R_{4,7}^{*} \oplus R_{4,8}^{*}$,

(4) $R_{4,1} \oplus R_{4,7} \oplus R_{4,8} \neq R_{4,1}^{*} \oplus R_{4,7}^{*} \oplus R_{4,8}^{*}$,

(5) $R_{4,3} \oplus R_{4,4} \oplus R_{4,5} \neq R_{4,3}^{*} \oplus R_{4,4}^{*} \oplus R_{4,5}^{*}$,

(6) $R_{4,2} \oplus R_{4,4} \oplus R_{4,5} \oplus R_{4,6} \oplus R_{4,7} \neq R_{4,2}^{*} \oplus R_{4,4}^{*} \oplus R_{4,5}^{*} \oplus R_{4,6}^{*} \oplus R_{4,7}^{*}$,

(7) $R_{4,2} \oplus R_{4,5} \neq R_{4,2}^{*} \oplus R_{4,5}^{*}$,

(8) $R_{4,4} \oplus R_{4,6} \neq R_{4,4}^{*} \oplus R_{4,6}^{*}$.

The above properties in the lemma provide some efficient 4-round distinguishers. They will be used to attack reduced-round Camellia.

3 Attacks on reduced-round Camellia with 128-bit key

3.1 Attacking 6-round Camellia with 128-bit key

This section explains the attack on 6-round Camellia with 128-bit key in some detail. First we recover the first byte $k_{1,1}$ of k_1 and the seventh byte $k_{6,7}$ of k_6. From the key schedule for 128-bit key, we know that $k_{6,7}[2—8]=k_{1,1}[1—7]$, so we only need to guess 9 bits. Using Lemma (1), we construct the following algorithm to recover $k_{1,1}$ and $k_{6,7}$.

Algorithm 1

Step 1. For each possible value t of $k_{1,1}$ choose two plaintexts $P0^t = (L0_0^t, R0_0^t)$ and $P1^t = (L1_0^t, R1_0^t)$ as follows.

$$L0_0^t = (i_0, \alpha_2, ..., \alpha_8),$$
$$R0_0^t = (s_1(i_0 \oplus k_{1,1}), s_1(i_0 \oplus k_{1,1}), s_1(i_0 \oplus k_{1,1}), \beta_4, s_1(i_0 \oplus k_{1,1}), \beta_6, \beta_7, s_1(i_0 \oplus k_{1,1})),$$

$$L1_0^t = (i_1, \alpha_2, ..., \alpha_8),$$
$$R1_0^t = (s_1(i_1 \oplus k_{1,1}), s_1(i_1 \oplus k_{1,1}), s_1(i_1 \oplus k_{1,1}), \beta_4, s_1(i_1 \oplus k_{1,1}), \beta_6, \beta_7, s_1(i_1 \oplus k_{1,1})),$$

where α_i and β_j are constants, $0 \leqslant i_0 < i_1 \leqslant 255$, and the corresponding ciphertexts are $C0^t = (L0_6^t, R0_6^t)$ and $C1^t = (L1_6^t, R1_6^t)$.

Step 2. For each possible value of $(t, k_{6,7})$, compute

$$\Delta_0 = s_4(R0_{6,7}^t \oplus k_{6,7}) \oplus (L0_{6,3}^t \oplus L0_{6,4}^t \oplus L0_{6,5}^t \oplus L0_{6,6}^t \oplus L0_{6,7}^t),$$

$$\Delta_1 = s_4(R1_{6,7}^t \oplus k_{6,7}) \oplus (L1_{6,3}^t \oplus L1_{6,4}^t \oplus L1_{6,5}^t \oplus L1_{6,6}^t \oplus L1_{6,7}^t).$$

Check if Δ_0 equals Δ_1. If so, record the corresponding value of $(t, k_{6,7})$. Otherwise, move to next value of $(t, k_{6,7})$.

Step 3. For the recorded value of $(t, k_{6,7})$ in Step 2, choose some other plaintexts $P2^t (\neq P1^t, P0^t)$, compute Δ_2, and check if Δ_2 equals Δ_1, if so, record the corresponding value of $(t, k_{6,7})$, otherwise, discard the value of $(t, k_{6,7})$. If there is more than one recorded value, then repeat Step 3 on the newly recorded values.

Take q values at random over F_2^8, the probability that they are the same is $2^{-8(q-1)}$. So invalid subkey will pass Step 2 with a probability 2^{-8}, and there are about $2^9 \times 2^{-8} = 2$ remaining values after Step 2. So the attack requires less than 3×2^8 chosen plaintexts. The main time complexity of attack is from Step 2, where the time complexity of computing each Δ is about the same as the 1-round encryption, so the time complexity of attack is less than 2^9 encryptions.

Given $k_{1,1}$, we can choose plaintexts such that the outputs of the first round meet the requirement of distinguishers in section 3. Thus, R_5 satisfies Lemma (3), and from $R_5 = L_6 \oplus F(R_6, k_6)$ and that $s_1(R_{6,1} \oplus k_{6,1})$ is the result of XOR of the 2nd, 3rd, 4th, 6th, 7th and 8th byte of $F(L_6, k_6)$, we have $R_{5,2} \oplus R_{5,3} \oplus R_{5,4} \oplus R_{5,6} \oplus R_{5,7} \oplus R_{5,8} = L_{6,2} \oplus L_{6,3} \oplus L_{6,4} \oplus L_{6,6} \oplus L_{6,7} \oplus L_{6,8} \oplus s_1(R_{6,1} \oplus k_{6,1})$. Using this equation and Lemma (3), we can construct the following algorithm to recover $k_{6,1}$.

Algorithm 2

Step 1. Choose 64 plaintexts $P^i = (L_0^i, R_0^i)$ $(0 \leq i \leq 63)$ as follows:

$$L_0^i = (i\ , \alpha_2, \ldots, \alpha_8),$$
$$R_0^i = (s_1(i\ \oplus k_{1,1}), s_1(i\ \oplus k_{1,1}), s_1(i\ \oplus k_{1,1}), \beta_4, s_1(i\ \oplus k_{1,1}), \beta_6, \beta_7, s_1(i\ \oplus k_{1,1})),$$

where α_i and β_j are constants. Denote by $C^i = (L_6^i, R_6^i)$ the corresponding ciphertexts of the above plaintexts.

Step 2. For each possible value of $k_{6,1}$, compute

$$\Delta_i = s_1(R_{6,1}^i \oplus k_{6,1}) \oplus (L_{6,2}^i \oplus L_{6,3}^i \oplus L_{6,4}^i \oplus L_{6,6}^i \oplus L_{6,7}^i \oplus L_{6,8}^i).$$

Check if there are collisions among Δ_i. If so, discard the value of $k_{6,1}$. Otherwise, output $k_{6,1}$.

Step 3. From the output values of $k_{6,1}$ in Step 2, choose some other plaintexts, and repeat Step 2.

The probability that at least one collision occurs when we throw 64 balls into 256 buckets at random is larger than $1-e^{-64(64-1)/2\times 2^8} \geq 1-2^{-11}$. So the probability of wrong output (invalid subkey) in Step 2 is less than 2^{-11}. For the 256 possible values of $k_{6,1}$, at most 64 more plaintexts are needed in Step 3. Thus, the attack requires less than 2^7 chosen plaintexts and 2^{12} encryptions.

Similarly, using Lemma (2) and the plaintexts chosen in Algorithm 2, we can recover $k_{6,6}$ by computing $\Delta_i = s_3(R^i_{6,6} \oplus k_{6,6}) \oplus (L^i_{6,2} \oplus L^i_{6,3} \oplus L^i_{6,5} \oplus L^i_{6,6} \oplus L^i_{6,8})$.

Check if Δ_i is a constant. If so, output the value of $k_{6,6}$, otherwise, discard the value of $k_{6,6}$. Here the attack requires 2^{10} encryptions.

And using $k_{6,6}$, Lemma (4) and the plaintexts chosen in Algorithm 2, we can recover $k_{6,4}$ by computing $\Delta_i = s_4(R^i_{6,4} \oplus k_{6,4}) \oplus s_3(R^i_{6,6} \oplus k_{6,6}) \oplus (L^i_{6,1} \oplus L^i_{6,7} \oplus L^i_{6,8})$. The attack requires 2^{12} encryptions.

And using Lemma (5) and the plaintexts chosen in Algorithm 2, we can recover $k_{6,2}$ by computing $\Delta_i = s_4(R^i_{6,4} \oplus k_{6,4}) \oplus s_2(R^i_{6,2} \oplus k_{6,2}) \oplus s_3(R^i_{6,6} \oplus k_{6,6}) \oplus (L^i_{6,3} \oplus L^i_{6,4} \oplus L^i_{6,5})$. The attack requires 2^{12} encryptions.

And using Lemma (6) and the plaintexts chosen in Algorithm 2, we can recover $k_{6,3}$ by computing $\Delta_i = s_4(R^i_{6,4} \oplus k_{6,4}) \oplus s_3(R^i_{6,3} \oplus k_{6,3}) \oplus s_3(R^i_{6,6} \oplus k_{6,6}) \oplus (L^i_{6,2} \oplus L^i_{6,4} \oplus L^i_{6,5} \oplus L^i_{6,6} \oplus L^i_{6,7})$. The attack requires 2^{12} encryptions.

And using Lemma (7) and the plaintexts chosen in Algorithm 2, we can recover $k_{6,5}$ by computing $\Delta_i = s_4(R^i_{6,4} \oplus k_{6,4}) \oplus s_2(R^i_{6,5} \oplus k_{6,5}) \oplus s_3(R^i_{6,6} \oplus k_{6,6}) \oplus (L^i_{6,2} \oplus L^i_{6,5})$. The attack requires 2^{12} encryptions.

And using Lemma (8) and the plaintexts chosen in Algorithm 2, we can recover $k_{6,8}$ by computing $\Delta_i = s_4(R^i_{6,4} \oplus k_{6,4}) \oplus s_3(R^i_{6,6} \oplus k_{6,6}) \oplus s_1(R^i_{6,8} \oplus k_{6,8}) \oplus (L^i_{6,4} \oplus L^i_{6,6})$. The attack requires 2^{12} encryptions.

Now we have recovered $k_{1,1}$ and k_6, using less than 2^{10} chosen plaintexts and $6\times 2^{12}+2^{10}+2^9$ encryptions. Similarly, by decrypting the 6th round, we can recover k_5. Therefore, the attack on the 6-round Camellia requires less than 2^{10} chosen plaintexts and 2^{15} encryptions.

3.2 Attacking 7-round Camellia with 128-bit key

From the structure of the round function, we have

$$R_{6,7} = L_{7,7} \oplus s_3(R_{7,3} \oplus k_{7,3}) \oplus s_4(R_{7,4} \oplus k_{7,4}) \oplus s_2(R_{7,5} \oplus k_{7,5}) \oplus s_3(R_{7,6} \oplus k_{7,6}) \oplus s_1(R_{7,8} \oplus k_{7,8}).$$

Similar to Algorithm 1 we can construct the following algorithm to recover $k_{1,1}$ and $(k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8})$.

Algorithm 3

Step 1. For each possible value t of $k_{1,1}$, choose 7 plaintexts $Pj^t = (Lj_0^t, Rj_0^t)$, $(1 \leqslant j \leqslant 7)$ as follows:

$$Lj_0^t = (i_j, \alpha_2, ..., \alpha_8),$$
$$Rj_0^t = (s_1(i_j \oplus k_{1,1}), s_1(i_j \oplus k_{1,1}), s_1(i_j \oplus k_{1,1}), \beta_4, s_1(i_j \oplus k_{1,1}), \beta_6, \beta_7, s_1(i_j \oplus k_{1,1})).$$

where α_i and β_j are constants, $0 \leqslant i_j \leqslant 255$, and the the corresponding ciphertexts are $Cj^t = (Lj_7^t, Rj_7^t)$.

Step 2. For each fixed t, and for each possible value of $(k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8})$ first compute Δ_1 and Δ_2, where

$$\Delta_j = s_4(Rj_{6,7}^t \oplus k_{6,7}) \oplus (Rj_{7,3}^t \oplus Rj_{7,4}^t \oplus Rj_{7,5}^t \oplus Rj_{7,6}^t \oplus Rj_{7,7}^t),$$

$$\begin{aligned} Rj_{6,7}^t = {} & Lj_{7,7}^t \oplus s_3(Rj_{7,3}^t \oplus k_{7,3}) \oplus s_4(Rj_{7,4}^t \oplus k_{7,4}) \\ & \oplus s_2(Rj_{7,5}^t \oplus k_{7,5}) \oplus s_3(Rj_{7,6}^t \oplus k_{7,6}) \oplus s_1(Rj_{7,8}^t \oplus k_{7,8}). \end{aligned}$$

Check if Δ_1 equals Δ_2. If so, output the value of $(k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8})$. Otherwise, discard the value of $(k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8})$.

For the output values of $(k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8})$, compute Δ_3, check if Δ_3 equals Δ_1. If so, output the value of $(k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8})$. Otherwise, discard the value of $(k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8})$. Similar process will go through Δ_4 up to Δ_7.

Step 3. For the output values of $(t, k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8})$ in Step 2, choose some other plaintexts $P8^t (\neq Pj^t, 1 \leqslant j \leqslant 7)$, compute Δ_8, check if Δ_8 equals Δ_1. If so, output the value of $(t, k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8})$. Otherwise, discard the value of $(t, k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8})$. If there is more than one output value, then repeat Step 3.

Invalid values of $(k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8})$ that can pass Step 2 will be successful with probability 2^{-48}. Thus it is likely that there is only one output value for any fixed t after Step 2, so there are about 2^8 different values after Step 2. Thus, the attack requires $7 \times 2^8 + 2^8 + 2^8 = 9 \times 2^8$ chosen plaintexts. The main time complexity of the attack is in Step 2, and the time of computing each Δ is about the same as 1-round encryption, so the time complexity of an attack is less than that of

$(2\times 2^8\times 2^{48}+2^8\times 2^{40}+2^8\times 2^{33})\div 7<2^{54}+2^{52}$ encryptions.

Now we have recovered $k_{1,1}$ and $(k_{6,7},k_{7,3},k_{7,4},k_{7,5},k_{7,6},k_{7,8})$, we can recover the other bytes of k_7, k_6 and k_5 similarly, so we can get the user key of 7-round Camellia, the attack requires less than 2^{12} chosen plaintexts and $2^{54.5}$ encryptions.

3.3 Attacking 8-round Camellia with 128-bit key

Similar to Algorithm 3, we can recover $k_{1,1}$ and $(k_{6,7},k_{7,3},k_{7,4},k_{7,5},k_{7,6},k_{7,8},k_8)$. Given $k_{1,1}$, we can get 7 bits of $k_{6,7}$ from the key schedule. Thus, $(k_{6,7},k_{7,3},k_{7,4},k_{7,5},k_{7,6},k_{7,8},k_8)$ have 2^{105} possible values. Here the attack requires 14 chosen plaintexts at Step 1. Invalid values of $(k_{6,7},k_{7,3},k_{7,4},k_{7,5},k_{7,6},k_{7,8},k_8)$ that pass Step 2 will be successful with probability 2^{-104}. There are about 2^9 output values of Step 2. So, the attack requires 2^{12} chosen plaintexts. The main time complexity of the attack is in Step 2, where the time of computing each Δ is about the 2-round encryption, so the time complexity of the attack is less than that of $2^{112}+2^{103}+2^{96}$ encryptions. Now we have recovered $k_{1,1}$ and $(k_{6,7},k_{7,3},k_{7,4},k_{7,5},k_{7,6},k_{7,8},k_8)$, we can decrypt the 8th round and recover the other bytes of k_7, so we can get the user key of 8-round Camellia, the attack requires less than 2^{13} chosen plaintexts and $2^{112.1}$ encryptions.

3.4 Attacking 9-round Camellia with 128 bit key

If we use the 4-round distinguisher from the 2nd to the 5th round of encryption as in the case of 8-round, then the time complexity of recovering 9-round Camellia key is larger than 2^{128} which is apparently useless. So we will use the 4-round distinguisher only from the 4th to the 7th round. First guess $k_1,k_{2,1},k_{2,2},k_{2,3},k_{2,5}$, $k_{2,8},k_{3,1},k_{8,7},k_{9,3},k_{9,4},k_{9,5},k_{9,6}$ and $k_{9,8}$. When $(k_1,k_{2,1},k_{2,2},k_{2,3},k_{2,5},k_{2,8})$ is given, we only need to guess 3 bits of $(k_{9,3},k_{9,4},k_{9,5},k_{9,6},k_{9,8})$.

Algorithm 4

Step 1. For each possible value t of $(k_1,k_{2,1},k_{2,2},k_{2,3},k_{2,5},k_{2,8},k_{3,1})$, choose 3 plaintexts $Pj^t=(Lj_0^t,Rj_0^t)$, $(1\leqslant j\leqslant 3)$, such that

$$Lj_2^t=(i_j,\alpha_2,\ldots,\alpha_8),$$
$$Rj_2^t=(s_1(i_j\oplus k_{3,1}),s_1(i_j\oplus k_{3,1}),s_1(i_j\oplus k_{3,1}),\beta_4,s_1(i_j\oplus k_{3,1}),\beta_6,\beta_7,s_1(i_j\oplus k_{3,1})),$$

where α_i and β_j are constants, $0\leqslant i_j\leqslant 255$, and the corresponding ciphertexts are $Cj^t=(Lj_9^t,Rj_9^t)$.

Step 2. For each fixed value of t, and for each possible value of $(k_{8,7},k_{9,3},k_{9,4},k_{9,5},k_{9,6},k_{9,8})$, compute Δ_1 and Δ_2, where

$$\Delta_j = s_4(Rj^t_{8,7} \oplus k_{8,7}) \oplus (Rj^t_{9,3} \oplus Rj^t_{9,4} \oplus Rj^t_{9,5} \oplus Rj^t_{9,6} \oplus Rj^t_{9,7}).$$

$$Rj^t_{8,7} = Lj^t_{9,7} \oplus s_3(Rj^t_{9,3} \oplus k_{9,3}) \oplus s_4(Rj^t_{9,4} \oplus k_{9,4}) \oplus s_2(Rj^t_{9,5} \oplus k_{9,5}) \oplus s_3(Rj^t_{9,6} \oplus k_{9,6}) \oplus s_1(Rj^t_{9,8} \oplus k_{9,8}).$$

Check if Δ_1 equals Δ_2. If so, output the value of $(k_{8,7}, k_{9,3}, k_{9,4}, k_{9,5}, k_{9,6}, k_{9,8})$. Otherwise, discard the value of $(k_{8,7}, k_{9,3}, k_{9,4}, k_{9,5}, k_{9,6}, k_{9,8})$.

For the output values of $(k_{8,7}, k_{9,3}, k_{9,4}, k_{9,5}, k_{9,6}, k_{9,8})$, compute Δ_3, check if Δ_3 equals Δ_1. If so, output the value of $(k_{8,7}, k_{9,3}, k_{9,4}, k_{9,5}, k_{9,6}, k_{9,8})$. Otherwise, discard the value of $(k_{8,7}, k_{9,3}, k_{9,4}, k_{9,5}, k_{9,6}, k_{9,8})$.

Step 3. For the output values of $(t, k_{8,7}, k_{9,3}, k_{9,4}, k_{9,5}, k_{9,6}, k_{9,8})$ in Step 2, choose some other plaintexts $P4^t(\neq P3^t, P2^t, P1^t)$, compute Δ_4, check if Δ_4 equals Δ_1. If so, output the value of $(t, k_{8,7}, k_{9,3}, k_{9,4}, k_{9,5}, k_{9,6}, k_{9,8})$. Otherwise, discard the value of $(t, k_{8,7}, k_{9,3}, k_{9,4}, k_{9,5}, k_{9,6}, k_{9,8})$. If there are more than one output value, then repeat Step 3.

Wrong values will pass Step 2 successfully with probability 2^{-16}. Thus there are about $2^{123} \times 2^{-16} = 2^{107}$ output values in Step 2. So the attack requires less than $3 \times 2^{112} + 2^{108}$ chosen plaintexts. The main time complexity of the attack is in Step 2, the time of computing each Δ is about the 1-round encryption, so the time complexity of the attack is less than $2^{120} + 2^{119} + 2^{118} + 2^{117}$ encryptions.

Now we know $k_1, k_{2,1}, k_{2,2}, k_{2,3}, k_{2,5}, k_{2,8}, k_{3,1}, k_{8,7}, k_{9,3}, k_{9,4}, k_{9,5}, k_{9,6}$ and $k_{9,8}$, we can recover the other bytes of k_9 and get the user key of 9-round Camellia. The attack requires less than $2^{113.6}$ chosen plaintexts and 2^{121} encryptions.

4 Attacks reduced-round Camellia with 192/256-bit key

4.1 Attacking 8-round Camellia with 192/256-bit key

First guess $k_{1,1}$ and $(k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}, k_8)$. When $k_{1,1}$ is given, we can get 8 bits of k_8 from the key schedule, so $(k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}, k_8)$ have 2^{104} possible values. Similar to section 3.3, we can attack 8-round Camellia with 192/256-bit key, requiring 2^{13} chosen plaintexts and $2^{111.1}$ encryptions.

4.2 Attacking 9-round Camellia with 192/256-bit key

Now we use the 4-round distinguisher from the 2nd to the 5th round of encryption. First guess $k_{1,1}$, $k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}$, k_8 and k_9. When $k_{1,1}$ is given, we can get 8 bits of k_8 from the key schedule, so we need guess 176 bits subkey. Using Lemma (1) we construct algorithm similar to Algorithm 3 and recover $k_{1,1}$ and

$(k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}, k_8, k_9)$. Here the attack requires 22 chosen plaintexts at Step 1. Invalid values of $(k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}, k_8, k_9)$ that pass Step 2 will be successful with probability 2^{-168}. Thus it is likely that there is only one output value for any fixed t after Step 2, so there are about 2^8 different values after Step 2. Thus, the attack requires $22 \times 2^8 + 2^8 + 2^8 = 3 \times 2^{11}$ chosen plaintexts. The main time complexity of the attack is in Step 2, where the time of computing each Δ is about the 3-round encryption, so the time complexity of the attack is less than that of $2^{175} + 2^{174}$ encryptions.

Now we have known $k_{1,1}$ and $(k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}, k_8, k_9)$, we can decrypt the 9th and 8th round and recover the other bytes of k_7 and get the user key of 9-round Camellia, the attack requires less than 2^{13} chosen plaintexts and $2^{175.6}$ encryptions.

4.3 Attacking 10-round Camellia with 256-bit key

First guess $k_{1,1}$, $k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}$, k_8, k_9 and k_{10}. When $k_{1,1}$ is given, we can get 8 bits of k_8 from the key schedule. So we need guess 240 bits subkey. Using Lemma (1) we construct the following algorithm.

Algorithm 5

Step 1. For each possible value t of $k_{1,1}$, choose 30 plaintexts $Pj^t = (Lj_0^t, Rj_0^t)$, $(1 \leqslant j \leqslant 30)$ as follows:

$$Lj_0^t = (i_j, \alpha_2, ..., \alpha_8),$$
$$Rj_0^t = (s_1(i_j \oplus k_{1,1}), s_1(i_j \oplus k_{1,1}), s_1(i_j \oplus k_{1,1}), \beta_4, s_1(i_j \oplus k_{1,1}), \beta_6, \beta_7, s_1(i_j \oplus k_{1,1})),$$

where α_i and β_j are constants, $0 \leqslant i_j \leqslant 255$, and the corresponding ciphertexts are $Cj^t = (Lj_{10}^t, Rj_{10}^t)$.

Step 2. For each fixed value of t, for each possible value of $(k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}, k_8, k_9, k_{10})$, compute Δ_1 and Δ_2, where

$$\Delta_j = s_4(Rj_{6,7}^t \oplus k_{6,7}) \oplus (Rj_{7,3}^t \oplus Rj_{7,4}^t \oplus Rj_{7,5}^t \oplus Rj_{7,6}^t \oplus Rj_{7,7}^t),$$

$$Rj_{6,7}^t = Lj_{7,7}^t \oplus s_3(Rj_{7,3}^t \oplus k_{7,3}) \oplus s_4(Rj_{7,4}^t \oplus k_{7,4})$$
$$\oplus s_2(Rj_{7,5}^t \oplus k_{7,5}) \oplus s_3(Rj_{7,6}^t \oplus k_{7,6}) \oplus s_1(Rj_{7,8}^t \oplus k_{7,8}),$$

$$Lj_7^t = Rj_8^t, Rj_7^t = Lj_8^t \oplus F(Rj_8^t, k_8), Lj_8^t = Rj_9^t, Rj_8^t = Lj_9^t \oplus F(Rj_9^t, k_9),$$

$$Lj_9^t = Rj_{10}^t, Rj_9^t = Lj_{10}^t \oplus F(Rj_{10}^t, k_{10}).$$

Check if Δ_1 equals Δ_2. If so, output the value of $(k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6},$

$k_{7,8}, k_8, k_9, k_{10})$. Otherwise, discard the value of $(k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}, k_8, k_9, k_{10})$.

For the output values of $(k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}, k_8, k_9, k_{10})$, compute Δ_3, check if Δ_3 equals Δ_1. If so, output the value of $(k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}, k_8, k_9, k_{10})$. Otherwise, discard the value of $(k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}, k_8, k_9, k_{10})$. Similar process will go through Δ_4 up to Δ_{30}.

Step 3. For the output values of $(t, k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}, k_8, k_9, k_{10})$ in Step 2, choose some other plaintexts $P31^t (\neq Pj^t, 1 \leqslant j \leqslant 30)$, compute Δ_{31}, check if Δ_{31} equals Δ_1. If so, output the value of $(t, k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}, k_8, k_9, k_{10})$. Otherwise, discard the value of $(t, k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}, k_8, k_9, k_{10})$. If there is more than one output value, then repeat Step 3.

Invalid values of $(k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}, k_8, k_9, k_{10})$ that can pass Step 2 will be successful with probability 2^{-232}. Thus it is likely that there is only one output value for any fixed t after Step 2, so there are about 2^8 different values after Step 2. Thus, the attack requires $30 \times 2^8 + 2^8 + 2^8 = 2^{13}$ chosen plaintexts. The main time complexity of the attack is in Step 2, and the time of computing each Δ is about the same as 4-round encryption, so the time complexity of an attack is less than that of $2^{239} + 2^{238} + 2^{237}$ encryptions.

Now we have known $(k_{1,1}, k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}, k_8, k_9, k_{10})$, we can decrypt the 10th, 9th and 8th rounds and recover the other bytes of k_7 and get the user key of 10-

Table 1 Comparison of attacks on Camellia

Rounds	FL/FL^{-1}	Methods	Plaintexts	Time	Notes
6	×	Higher Order DC	2^{17}	$2^{19.4}$	ref. [3] (128-bit key)
6	×	Square Attack	$2^{11.7}$	2^{112}	ref. [8] (128-bitkey)
6	×	Collision Attack	2^{10}	2^{15}	this paper (128-bit key)
7	×	Higher Order DC	2^{19}	$2^{61.2}$	ref. [3] (128-bit key)
7	√	Square Attack	$2^{58.3}$	$2^{80.2}$	ref. [8] (128-bit key)
7	×	Collision Attack	2^{12}	$2^{54.5}$	this paper (128-bit key)
8	×	Truncated DC	$2^{83.6}$	$2^{55.6}$	ref. [4] (128-bit key)
8	√	Integral Attack	$2^{59.7}$	$2^{137.6}$	ref. [9] (256-bit key)
8	×	Collision Attack	2^{13}	$2^{112.1}$	this paper (128-bit key)
8	×	Collision Attack	2^{13}	$2^{111.1}$	this paper (192/256-bit key)
9	×	Higher Order DC	2^{21}	$2^{190.8}$	ref. [3] (256-bit key)
9	√	Integral Attack	$2^{60.5}$	$2^{202.2}$	ref. [9] (256-bit key)
9	×	Collision Attack	$2^{113.6}$	2^{121}	this paper (128-bit key)
9	×	Collision Attack	2^{13}	$2^{175.6}$	this paper (192/256-bit key)
10	×	Higher Order DC	2^{21}	$2^{254.7}$	ref. [3] (256-bit key)
10	×	Collision Attack	2^{14}	$2^{239.9}$	this paper (256-bit key)
11	√	Higher Order DC	2^{93}	$2^{255.6}$	ref. [3] (256-bit key)

round Camellia. The attack requires less than 2^{14} chosen plaintexts and $2^{239.9}$ encryptions.

5 Concluding remarks

In this paper we have shown 4-round distinguishers of Camellia, and discussed the security of Camellia by using the 4-round distinguishers and collision-searching techniques. Table 1 compares the performance of some known attacks on Camellia.

Acknowledgements This work was supported by the National Natural Science Foundation of China (Grant No. 60373047) and the State 863 Project (Grant No. 2003AA144030), and 973 Project (Grant No. 2004CB318004).

References

1. http://www.cryptonessie.org.
2. Aoki, K., Ichikawa, T., Kanda, M. et al., Specification of Camellia—a 128-bit block cipher, Selected Areas in Cryptography—SAC'2000, Berlin: Springer-Verlag, 2000, 183—191.
3. Hatano, Y., Sekine, H., Kaneko, T., Higher order differential attack of Camellia (II), Selected Areas in Cryptography—SAC'02, Berlin: Springer-Verlag, 2002, 39—56.
4. Lee, S., Hong, S., Lim, J. et al., Truncated differential cryptanalysis of Camellia, ICISC2001, Berlin: Springer-Verlag, 1993, 32—38.
5. Sugita, M., Kobara, K., Imai, H., Security of reduced version of the block cipher Camellia against truncated and impossible differential cryptanalysis, Asiacrypt'01, Berlin: Springer-Verlag, 2001, 193—207.
6. Shirai, T., Kanamaru, S., Abe, G., Improved upper bounds of differential and linear characteristic probability for Camellia, Fast Software Encryption-FSE'02, Berlin: Springer-Verlag, 2002,128—142.
7. He Yeping, Qing Sihan, Square attack on reduced Camellia cipher, ICICS2001, Berlin: Springer-Verlag, 2001, 238—245.
8. Yeom, Y., Park, S., Kim, I., On the security of Camellia against the square attack, Fast Software Encryption-FSE'02, Berlin: Springer-Verlag, 2002, 89—99.
9. Yeom, Y., Park, S., Kim, I., A study of Integral type cryptanalysis on Camellia, The 2003 Symposium on Cryptography and Security –SCS'03, Hamamatsu, Japan, 2003, 26—29.