Collision Attack and Pseudorandomness of Reduced-Round Camellia[1]

Wu Wenling, Feng Dengguo, and Chen Hua

State Key Laboratory of Information Security, Institute of Software,
Chinese Academy of Sciences, Beijing 100080, P. R. China
{wwl, feng, chenhua}@is.iscas.ac.cn

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 random permutation of the blocks space. By using collision-searching techniques, the distinguishers are used to attack 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 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. Furthermore, we prove that the 4-round primitive-wise idealized Camellia is not pseudorandom permutation and the 5-round primitive-wise idealized Camellia is super-pseudorandom permutation for non-adaptive adversaries.

Keywords: Block cipher; Camellia; Data complexity; Time complexity; Pseudorandomness.

1 Introduction

Camellia[1] is a 128-bit block cipher which was published by NTT and Mitsubishi in 2000 and selected as the final selection of the NESSIE[2] project. The security of Camellia has been studied by many researchers using various cryptanalytic methods, for instance: higher-order differential attack[3,4], truncated differential attack[5], truncated and impossible differential attacks[6], differential attack[7], square attack[8,9], integral attack[10]. In this paper we present collision attacks on reduced-round variants of Camellia without FL/FL^{-1} and whitening function layers. 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.

In addition to cryptanalytic methods mentioned above, pseudorandomness is also an important cryptographic criterion of iterated block ciphers. In their celebrated paper[11], Luby and Rackoff introduced a theoretical model for the

[1] This work was supported by Chinese Natural Science Foundation (Grant No.60373047 and 60025205) and 863 Project (Grant No. 2003AA14403).

H. Handschuh and A. Hasan (Eds.): SAC 2004, LNCS 3357, pp. 252–266, 2005.

security of block ciphers by using the notion of pseudorandom and super-pseudorandom permutations, which was later developed by Patarin[12], Maurer[13], Vaudenay[14], and other researchers. This approach studies the pseudorandomness of block cipher by assuming that each round function is ideally random. Luby and Rackoff idealized DES by replacing each round function with one large random function, then they proved that the idealized three round DES yields a pseudorandom permutation and the idealized four round DES yields a super-pseudorandom permutation. For this kind of idealization, the three round idealized Camellia is a pseudorandom permutation and the four round idealized Camellia is a super-pseudorandom permutation because Camellia has the same Feistel structure as DES. Iwata and Kurosawa[15] introduced a primitive-wise idealization in which some of the primitive operations of the round function(e.g., linear transformation and etc.) are left untouched and some of them (e.g., S-boxes and etc.) are replaced with small random functions or permutations. It is not known whether such a primitive-wise idealization DES is pseudorandom (or super-pseudorandom). Similarly, the same problem has been open for Camellia, which is solved in this paper. In section 6, Camellia is idealized by replacing only the S-boxes with small random functions. We then prove that the 4-round primitive-wise idealized Camellia is not pseudorandom permutation and the 5-round primitive-wise idealized Camellia is super-pseudorandom permutation for non-adaptive adversaries.

This paper is organized as follows: Section 2 briefly introduces the structure of Camellia and the basic definitions on pseudorandomness. 4-round distinguishers are explained in section 3. In section 4, 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 5, we describe attacks on 9 and 10 rounds of Camellia with 192/256-bit key. Section 6 present our results on the pseudorandomness and super-pseudorandomness of Camellia, and Section 7 concludes the paper.

2 Preliminaries

2.1 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

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

here F is the round function defined below:

$$\begin{aligned} F : \{0,1\}^{64} \times \{0,1\}^{64} &\longrightarrow \{0,1\}^{64} \\ (X_{64}, k_{64}) &\longrightarrow Y_{(64)} = P(S(X_{(64)} \oplus k_{(64)})). \end{aligned}$$

where S and P are defined as follows:

$$\begin{aligned} S : \{0,1\}^{64} &\longrightarrow \{0,1\}^{64} \\ & l_{1(8)}||l_{2(8)}||l_{3(8)}||l_{4(8)}||l_{5(8)}||l_{6(8)}||l_{7(8)}||l_{8(8)} \\ & \qquad \longrightarrow l^*_{1(8)}||l^*_{2(8)}||l^*_{3(8)}||l^*_{4(8)}||l^*_{5(8)}||l^*_{6(8)}||l^*_{7(8)}||l^*_{8(8)} \end{aligned}$$

$$\begin{array}{lll} l^*_{1(8)} = s_1(l_{1(8)}), & l^*_{2(8)} = s_2(l_{2(8)}), & l^*_{3(8)} = s_3(l_{3(8)}), \\ l^*_{4(8)} = s_4(l_{4(8)}), & l^*_{5(8)} = s_2(l_{5(8)}), & l^*_{6(8)} = s_3(l_{6(8)}), \\ l^*_{7(8)} = s_4(l_{7(8)}), & l^*_{8(8)} = s_1(l_{8(8)}). & \end{array}$$

$$\begin{aligned} P : \{0,1\}^{64} &\longrightarrow \{0,1\}^{64} \\ & Z_{1(8)}||Z_{2(8)}||Z_{3(8)}||Z_{4(8)}||Z_{5(8)}||Z_{6(8)}||Z_{7(8)}||Z_{8(8)} \\ & \qquad \longrightarrow Z^*_{1(8)}||Z^*_{2(8)}||Z^*_{3(8)}||Z^*_{4(8)}||Z^*_{5(8)}||Z^*_{6(8)}||Z^*_{7(8)}||Z^*_{8(8)} \end{aligned}$$

$$\begin{array}{ll} Z^*_1 = Z_1 \oplus Z_3 \oplus Z_4 \oplus Z_6 \oplus Z_7 \oplus Z_8, & Z^*_5 = Z_1 \oplus Z_2 \oplus Z_6 \oplus Z_7 \oplus Z_8, \\ Z^*_2 = Z_1 \oplus Z_2 \oplus Z_4 \oplus Z_5 \oplus Z_7 \oplus Z_8, & Z^*_6 = Z_2 \oplus Z_3 \oplus Z_5 \oplus Z_7 \oplus Z_8, \\ Z^*_3 = Z_1 \oplus Z_2 \oplus Z_3 \oplus Z_5 \oplus Z_6 \oplus Z_8, & Z^*_7 = Z_3 \oplus Z_4 \oplus Z_5 \oplus Z_6 \oplus Z_8, \\ Z^*_4 = Z_2 \oplus Z_3 \oplus Z_4 \oplus Z_5 \oplus Z_6 \oplus Z_7, & Z^*_8 = Z_1 \oplus Z_4 \oplus Z_5 \oplus Z_6 \oplus Z_7. \end{array}$$

Below briefly describes the key schedule of Camellia. 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. The round subkeys are generated by rotating K_L, K_R, K_A and K_B. Details are shown in [1]

2.2 Pseudorandomness and Super-Pseudorandomness

Let $\{0,1\}^n$ denote the set of binary strings of length n, let F_n denote the set of functions from $\{0,1\}^n$ to $\{0,1\}^n$ and P_n denote the set of permutations from $\{0,1\}^n$ to $\{0,1\}^n$. A n-bit block cipher can be regarded as a subset of permutations $B_n \subset P_n$ obtained from all the keys. Let $\mathcal{A}$ be a computationally unbounded distinguisher with an oracle $\mathcal{O}$. The oracle chooses randomly a permutation π from P_n or B_n. The aim of the distinguisher $\mathcal{A}$ is to distinguish if the oracle $\mathcal{O}$ implements P_n or B_n. Let p_0 denote the probability that $\mathcal{A}$ outputs 1 when $\mathcal{O}$ implements P_n and p_1 denote the probability that $\mathcal{A}$ outputs 1 when $\mathcal{O}$ implements B_n. That is $p_0 = Pr(\mathcal{A} \textit{ outputs 1 } \mid \mathcal{O} \leftarrow P_n)$ and $p_1 = Pr(\mathcal{A} \textit{ outputs 1 } \mid \mathcal{O} \leftarrow B_n)$. Then the advantage of the distinguisher $\mathcal{A}$ is defined as

$$Adv_A = \mid p_1 - p_0 \mid$$

Assume that the distinguisher $\mathcal{A}$ is restricted to make at most q queries to the oracle $\mathcal{O}$, where q is some polynomial in n. We say that $\mathcal{A}$ is pseudorandom distinguisher if it queries x and the oracle answers $y = \pi(x)$, where π is randomly chosen permutation by $\mathcal{O}$. We say that $\mathcal{A}$ is super-pseudorandom distinguisher if it is also allowed to query y and receives $x = \pi^{-1}(y)$ from the oracle.

Definition 1. A function $h : N \to R$ is negligible if for any constant $c > 0$ and all sufficiently large $n \in N$, $h(n) < \frac{1}{n^c}$.

Definition 2. Let B_n be an efficiently computable permutation ensemble. B_n is called a pseudorandom permutation ensemble if Adv_A is negligible for any pseudorandom distinguisher $\mathcal{A}$.

Definition 3. Let B_n be an efficiently computable permutation ensemble. B_n is called a super-pseudorandom permutation ensemble if Adv_A is negligible for any super-pseudorandom distinguisher $\mathcal{A}$.

In definition 2 and 3, a permutation ensemble is efficiently computable if all permutations in the ensemble can be computed efficiently. See[16] for the rigorous definition of this. It is reasonable assumption that B_n is an efficiently computable permutation ensemble if it is obtained from an n-bit block cipher. In Section 6 , we consider a non-adaptive distinguisher which sends all the queries to the oracle at the same time.

3 4-Round Distinguishers

Choose

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

where x take values in $\{0,1\}^8$, α_i and β_j are constants in $\{0,1\}^8$. Thus, the input of 2nd round can be written as follows:

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

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 $F(\bullet, k_2)$ is as follows:

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

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 2nd round is

$$L_2 = (y \oplus \varpi_1, y \oplus \varpi_2, y \oplus \varpi_3, \varpi_4, y \oplus \varpi_5, \varpi_6, \varpi_7, y \oplus \varpi_8),$$
$$R_2 = L_1 = (x \oplus \gamma_1, \gamma_2, \cdots, \gamma_8),$$

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

$$L_2 = (y \oplus \varpi_1, y \oplus \varpi_2, y \oplus \varpi_3, \varpi_4, y \oplus \varpi_5, \varpi_6, \varpi_7, y \oplus \varpi_8) \xrightarrow{F(\bullet, k_3)} (z_1, z_2, \cdots, z_8).$$

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

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

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

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

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

$$\begin{aligned}
&z_3 \oplus z_4 \oplus z_5 \oplus z_6 \oplus z_7 = s_4(\varpi_7 \oplus k_{3,7}) \\
&z_2 \oplus z_3 \oplus z_4 \oplus z_6 \oplus z_7 \oplus z_8 = s_1(y \oplus \varpi_1 \oplus k_{3,1}) \\
&z_2 \oplus z_3 \oplus z_5 \oplus z_6 \oplus z_8 = s_3(\varpi_6 \oplus k_{3,6}) \\
&z_1 \oplus z_7 \oplus z_8 = s_4(\varpi_4 \oplus k_{3,4}) \oplus s_3(\varpi_6 \oplus k_{3,6}) \\
&z_3 \oplus z_4 \oplus z_5 = s_4(\varpi_4 \oplus k_{3,4}) \oplus s_2(y \oplus \varpi_2 \oplus k_{3,2}) \oplus s_3(\varpi_6 \oplus k_{3,6}) \\
&z_2 \oplus z_4 \oplus z_5 \oplus z_6 \oplus z_7 = s_4(\varpi_4 \oplus k_{3,4}) \oplus s_3(y \oplus \varpi_3 \oplus k_{3,3}) \oplus s_3(\varpi_6 \oplus k_{3,6}) \\
&z_2 \oplus z_5 = s_4(\varpi_4 \oplus k_{3,4}) \oplus s_2(y \oplus \varpi_5 \oplus k_{3,5}) \oplus s_3(\varpi_6 \oplus k_{3,6}) \\
&z_4 \oplus z_6 = s_4(\varpi_4 \oplus k_{3,4}) \oplus s_1(y \oplus \varpi_8 \oplus k_{3,8}) \oplus s_3(\varpi_6 \oplus k_{3,6})
\end{aligned}$$

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, $s_1(y \oplus \varpi_1 \oplus k_{3,1})$ will have different values. Similarly,$s_2(y \oplus \varpi_2 \oplus k_{3,2}), s_3(y \oplus \varpi_3 \oplus k_{3,3}), s_2(y \oplus \varpi_5 \oplus k_{3,5})$ and $s_1(y \oplus \varpi_8 \oplus k_{3,8})$ have the same property as $s_1(y \oplus \varpi_1 \oplus k_{3,1})$. Obviously, $s_4(\varpi_4 \oplus k_{3,4}), s_3(\varpi_6 \oplus k_{3,6})$ and $s_4(\varpi_7 \oplus k_{3,7})$ 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 theorem by considering $R_4 = L_3 = (z_1 \oplus x \oplus \gamma_1, z_2 \oplus \gamma_2, z_3 \oplus \gamma_3, \cdots, z_8 \oplus \gamma_8)$.

Theorem 1. *Let $P = (L_0, R_0)$ and $P_0^* = (L_0^*, R_0^*)$ be two plaintexts of 4-round Camellia, $C = (L_4, R_4)$ and $C_4^* = (L_4^*, R_4^*)$ be the corresponding ciphertexts, $R_{0,i}$ denote the i^{th} byte of R_0. If $L_0 = L_0^*, R_{0,1} \neq R_{0,1}^*, R_{0,j} = R_{0,j}^* (2 \leq j \leq 8)$, then R_4 and R_4^* satisfy:*

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

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

$$\begin{aligned}&R_{4,2} \oplus R_{4,3} \oplus R_{4,4} \oplus R_{4,6} \oplus R_{4,7} \oplus R_{4,8} \\ &\qquad \neq R_{4,2}^* \oplus R_{4,3}^* \oplus R_{4,4}^* \oplus R_{4,6}^* \oplus R_{4,7}^* \oplus R_{4,8}^*\end{aligned} \tag{3}$$

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

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

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

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

$$R_{4,4} \oplus R_{4,6} \neq R_{4,4}^* \oplus R_{4,6}^* \tag{8}$$

The above (in)equations in the theorem 1 provide some efficient 4-round distinguishers,which will be used to attack and show the pseudorandomness of reduced-round Camellia.

4 Attacks on Reduced-Round Camellia with 128 Bit Key

4.1 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 $k_{6,7}[2 \sim 8] = k_{1,1}[1 \sim 7]$, so we only need to guess 9 bits. Using the equation (1) of theorem 1, we construct the following algorithm to recover$(k_{1,1}, k_{6,7})$:

Algorithm 1
Step1. 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:

$$L0_0^t = (i_0, \alpha_2, \cdots, \alpha_8),$$
$$R0_0^t = (s_1(i_0 \oplus k_{1,1}), s_1(i_0 \oplus k_{1,1}), s_1(i_0 \oplus k_{1,1}), \beta_4, s_1(i_0 \oplus k_{1,1}), \beta_6, \beta_7, s_1(i_0 \oplus k_{1,1})),$$
$$L1_0^t = (i_1, \alpha_2, \cdots, \alpha_8),$$
$$R1_0^t = (s_1(i_1 \oplus k_{1,1}), s_1(i_1 \oplus k_{1,1}), s_1(i_1 \oplus k_{1,1}), \beta_4, s_1(i_1 \oplus k_{1,1}), \beta_6, \beta_7, s_1(i_1 \oplus k_{1,1})).$$

where α_i and β_j are constants, $0 \leq i_0 < i_1 \leq 255$. The corresponding ciphertexts are $C0^t = (L0_6^t, R0_6^t)$ and $C1^t = (L1_6^t, R1_6^t)$.
Step2. For each possible value of $(t, k_{6,7})$, compute

$$\triangle_0 = s_4(R0_{6,7}^t \oplus k_{6,7}) \oplus (L0_{6,3}^t \oplus L0_{6,4}^t \oplus L0_{6,5}^t \oplus L0_{6,6}^t \oplus L0_{6,7}^t),$$
$$\triangle_1 = s_4(R1_{6,7}^t \oplus k_{6,7}) \oplus (L1_{6,3}^t \oplus L1_{6,4}^t \oplus L1_{6,5}^t \oplus L1_{6,6}^t \oplus L1_{6,7}^t).$$

Check if $\triangle_0$ equals $\triangle_1$. If so, record the corresponding value of $(t, k_{6,7})$. Otherwise, move to next value of $(t, k_{6,7})$.
Step3. For the recorded value of $(t, k_{6,7})$ in Step2, choose some other plaintexts $P2^t (\neq P0^t, P1^t)$, compute $\triangle_2$, and check if $\triangle_2$ equals $\triangle_0$, if so, record the corresponding value of $(t, k_{6,7})$, otherwise, discard the value of $(t, k_{6,7})$. If there are more than one recorded value, then repeat Step 3 on the newly recorded values.

Take q values at random over $\{0,1\}^8$, the probability of that they are the same is $2^{-8(q-1)}$. So invalid subkey will pass step2 with a probability 2^{-8}, and there are about $2^9 \times 2^{-8} = 2$ remaining values after step2. So the attack requires less than 3×2^8chosen plaintexts. The main time complexity of attack is from step2, where the time complexity of computing each $\triangle$ is about the same as the 1-round encryption, so the time complexity of attack is less than 2^9 encryptions.

Knowing $k_{1,1}$, we can choose plaintexts such that the outputs of the first round meet the requirement of Theorem 1. Thus, R_5 satisfies Theorem 1, 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 $\oplus$ of the 2nd ,3rd ,4th ,6th,7th and 8th byte of $F(R_6, k_6)$, 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 inequation (3) in Theorem 1, we can construct the following algorithm to recover $k_{6,1}$:

Algorithm 2
Step1. Choose 64 plaintexts $P^i = (L_0^i, R_0^i)(0 \leq i \leq 63)$ as follows:

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

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.
Step2. For each possible value of $k_{6,1}$, compute

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

Check if there are collisions among $\triangle_i$. If so, discard the value of $k_{6,1}$. Otherwise, output $k_{6,1}$.
Step3. From the output values of $k_{6,1}$ in Step2, choose some other plaintexts, and repeat Step2.

The probability of 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 Step2 is less than 2^{-11}. For the 256 possible values of $k_{6,1}$, at most 64 more plaintexts are needed in Step3. Thus, the attack requires less than 2^7 chosen plaintexts and 2^{12} encryptions.

Similarly, using equation (2) in Theorem 1 and the plaintexts chosen in Algorithm 2, we can recover $k_{6,6}$ by computing

$$\triangle_i = s_3(R_{6,6}^i \oplus k_{6,6}) \oplus (L_{6,2}^i \oplus L_{6,3}^i \oplus L_{6,5}^i \oplus L_{6,6}^i \oplus L_{6,8}^i).$$

Check if $\triangle_i$ 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 $k_{6,6}$, inequation (4) in Theorem 1 and the plaintexts chosen in Algorithm 2, we can recover $k_{6,4}$ by computing

$$\triangle_i = s_4(R_{6,4}^i \oplus k_{6,4}) \oplus s_3(R_{6,6}^i \oplus k_{6,6}) \oplus (L_{6,1}^i \oplus L_{6,7}^i \oplus L_{6,8}^i).$$

and the attack requires 2^{12} encryptions.

And using inequation (5) in Theorem 1 and the plaintexts chosen in Algorithm 2, we can recover $k_{6,2}$ by computing

$$\triangle_i = s_4(R_{6,4}^i \oplus k_{6,4}) \oplus s_2(R_{6,2}^i \oplus k_{6,2}) \oplus s_3(R_{6,6}^i \oplus k_{6,6}) \oplus (L_{6,3}^i \oplus L_{6,4}^i \oplus L_{6,5}^i).$$

and the attack requires 2^{12} encryptions.

And using inequation (6) in Theorem 1 and the plaintexts chosen in Algorithm 2, we can recover $k_{6,3}$ by computing

$$\triangle_i = s_4(R_{6,4}^i \oplus k_{6,4}) \oplus s_3(R_{6,3}^i \oplus k_{6,3}) \oplus s_3(R_{6,6}^i \oplus k_{6,6}) \oplus (L_{6,2}^i \oplus L_{6,4}^i \oplus L_{6,5}^i \oplus L_{6,6}^i \oplus L_{6,7}^i).$$

and the attack requires 2^{12} encryptions.

And using inequation (7) in Theorem 1 and the plaintexts chosen in Algorithm 2, we can recover $k_{6,5}$ by computing

$$\triangle_i = s_4(R_{6,4}^i \oplus k_{6,4}) \oplus s_2(R_{6,5}^i \oplus k_{6,5}) \oplus s_3(R_{6,6}^i \oplus k_{6,6}) \oplus (L_{6,2}^i \oplus L_{6,5}^i).$$

and the attack requires 2^{12} encryptions.

And using inequation (8) in Theorem 1 and the plaintexts chosen in Algorithm 2, we can recover $k_{6,8}$ by computing

$$\triangle_i = s_4(R^i_{6,4} \oplus k_{6,4}) \oplus s_1(R^i_{6,8} \oplus k_{6,8}) \oplus s_3(R^i_{6,6} \oplus k_{6,6}) \oplus (L^i_{6,4} \oplus L^i_{6,6}).$$

and 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.

Similarly we can get the user key of 7(8)-round Camellia. For 7-round Camellia, the attack requires less than 2^{12} *chosen plaintexts and* $2^{54.5}$ *encryptions. For 8-round Camellia, the attack requires less than* 2^{13} *chosen plaintexts and* $2^{112.1}$ *encryptions.*

4.2 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 $k_1, k_{2,1}, k_{2,2}, k_{2,3}, k_{2,5}, k_{2,8}, k_{3,1}, k_{8,7}, k_{9,3}, k_{9,4}, k_{9,5}, k_{9,6}, k_{9,8}$, 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 3

Step1. 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 $Pj^t = (Lj^t_0, Rj^t_0)(1 \leq j \leq 3)$ such that

$$Lj^t_2 = (i_j, \alpha_2, \cdots, \alpha_8),$$
$$Rj^t_2 = (s_1(i_j \oplus k_{3,1}), s_1(i_j \oplus k_{3,1}), s_1(i_j \oplus k_{3,1}), \beta_4, s_1(i_j \oplus k_{3,1}), \beta_6, \beta_7, s_1(i_j \oplus k_{3,1})).$$

where α_i and β_j are constants, $0 \leq i_j \leq 255$, and the the corresponding ciphertexts are $Cj^t = (Lj^t_9, Rj^t_9)$.

Step2. 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 $\triangle_1$ and $\triangle_2$, where

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

Check if $\triangle_1$ equals $\triangle_2$. 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 $\triangle_3$, check if $\triangle_3$ equals $\triangle_1$. 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 $(k_{8,7}, k_{9,3}, k_{9,4}, k_{9,5}, k_{9,6}, k_{9,8})$.

Step3. 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 Step2, Choose some other plaintexts $P4^t(\neq Pj^t, 1 \leq j \leq 3)$, compute $\triangle_4$, check if $\triangle_4$ equals $\triangle_1$. 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 Step3.

Wrong values will pass step2 successfully with probability 2^{-16}. Thus there are about $2^{123} \times 2^{-16} = 2^{107}$ output values in step2. So, the attack requires less than $3 \times 2^{112} + 2^{108}$ chosen plaintexts. The main time complexity of the attck is in Step2, the time of computing each $\triangle$ is about the 1-round encryption, so the time complexity of the attck is less than $(2 \times 2^{112} \times 2^{11} + 2^{116}) \times 1/9 < 2^{120} + 2^{119} + 2^{118} + 2^{117}$ encryptions.

Now we know $k_1, k_{2,1}, k_{2,2}, k_{2,3}, k_{2,5}, k_{2,8}, k_{3,1}, k_{8,7}, k_{9,3}, k_{9,4}, k_{9,5}, k_{9,6}, k_{9,8}$, 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.

5 Attacks Reduced-Round Camellia with 192/256 Bit Key

5.1 Attacking 9-Round Camellia with 192/256 Bit Key

First guess $k_{1,1}, k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}, k_8, k_9$. 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 equation (1) in Theorem 1, we can construct the following algorithm:

Algorithm 4
Step1. For each possible value t of $k_{1,1}$, Choose 22 plaintexts $Pj^t = (Lj_0^t, Rj_0^t)$ $(1 \leq j \leq 22)$ as follows:

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

where α_i and β_j are constants, $0 \leq i_j \leq 255$, and the the corresponding ciphertexts are $Cj^t = (Lj_9^t, Rj_9^t)$.
Step2. For each fixed value of t, for each possible value of $(k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}, k_8, k_9)$, First compute $\triangle_1$ and $\triangle_2$, where

$$\begin{aligned}
&\triangle_j = s_4(Rj_{6,7}^t \oplus k_{6,7}) \oplus (Rj_{7,3}^t \oplus Rj_{7,4}^t \oplus Rj_{7,5}^t \oplus Rj_{7,6}^t \oplus Rj_{7,7}^t),\\
&Lj_7^t = Rj_8^t,\ Rj_7^t = Lj_8^t \oplus F(Rj_8^t, k_8), \qquad Lj_8^t = Rj_9^t, \quad Rj_8^t = Lj_9^t \oplus F(Rj_9^t, k_9),\\
&Rj_{6,7}^t = Lj_{7,7}^t \oplus s_3(Rj_{7,3}^t \oplus k_{7,3}) \oplus s_4(Rj_{7,4}^t \oplus k_{7,4}) \oplus s_2(Rj_{7,5}^t \oplus k_{7,5})\\
&\qquad \oplus s_3(Rj_{7,6}^t \oplus k_{7,6}) \oplus s_1(Rj_{7,8}^t \oplus k_{7,8}).
\end{aligned}$$

Check if $\triangle_1$ equals $\triangle_2$. If so, output the value of $(k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}, k_8, k_9)$. Otherwise, discard the value of $(k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}, k_8, k_9)$.

For the output values of $(k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}, k_8, k_9)$, compute $\triangle_3$, check if $\triangle_3$ equals $\triangle_1$. If so, output the value of$(k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8},$

$k_8, k_9)$. Otherwise, discard the value of $(k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}, k_8, k_9)$. Similar process will go through $\triangle_4$ up to $\triangle_{22}$.

Step3. For the output values of $(t, k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}, k_8, k_9)$ in Step2, choose some other plaintexts $P23^t (\neq Pj^t, 1 \leq j \leq 22)$, compute $\triangle_{23}$, check if $\triangle_{23}$ equals $\triangle_1$. If so, output the value of $(t, k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}, k_8, k_9)$. Otherwise, discard the value of$(t, k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}, k_8, k_9)$. If there are more than one output value, then repeat Step3.

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 can pass Step2 will be successful with probability 2^{-168}. Thus it is likely that there is only one output value for any fixed t after Step2, so there are about 2^8 different values after step2. 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 Step2, and the time of computing each $\triangle$ is about the same as 3-round encryption, so the time complexity of an attack is less than that of $(2 \times 2^8 \times 2^{168} + 2^8 \times 2^{160} + 2^8 \times 2^{153}) \times 1/3 < 2^{175} + 2^{174}$ encryptions.

Now we have known $(k_{1,1}, k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}, k_8, k_9)$, we can decrypt the ninth and eighth 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.

5.2 Attacking 10-Round Camellia with 256 Bit Key

First guess $k_{1,1}, k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}, k_8, k_9, k_{10}$. 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 equation (1) in Theorem 1, we construct the following algorithm:

Algorithm 5

Step1. For each possible value t of $k_{1,1}$, Choose 30 plaintexts $Pj^t = (Lj_0^t, Rj_0^t)$ $(1 \leq j \leq 30)$ as follows:

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

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

Step2. For each fixed value of t, for each possible value of $(k_{6,7}, k_{7,3}, k_{7,4}, k_{7,5}, k_{7,6}, k_{7,8}, k_8, k_9, k_{10})$, First compute $\triangle_1$ and $\triangle_2$,where

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

$$\begin{aligned}
&Lj_7^t = Rj_8^t, \quad Rj_7^t = Lj_8^t \oplus F(Rj_8^t, k_8),\\
&Lj_8^t = Rj_9^t, \quad Rj_8^t = Lj_9^t \oplus F(Rj_9^t, k_9),\\
&Lj_9^t = Rj_{10}^t, \quad Rj_9^t = Lj_{10}^t \oplus F(Rj_{10}^t, k_{10}),\\
&Rj_{6,7}^t = Lj_{7,7}^t \oplus s_3(Rj_{7,3}^t \oplus k_{7,3}) \oplus s_4(Rj_{7,4}^t \oplus k_{7,4}) \oplus s_2(Rj_{7,5}^t \oplus k_{7,5})\\
&\qquad \oplus s_3(Rj_{7,6}^t \oplus k_{7,6}) \oplus s_1(Rj_{7,8}^t \oplus k_{7,8}).
\end{aligned}$$

Check if $\triangle_1$ equals $\triangle_2$. 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 $\triangle_3$, check if $\triangle_3$ equals $\triangle_1$. 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 $\triangle_4$ up to $\triangle_{30}$.

Step3. 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 Step2, choose some other plaintexts $P31^t(\neq Pj^t, 1 \leq j \leq 30)$, compute $\triangle_{31}$, check if $\triangle_{31}$ equals $\triangle_1$. 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 are more than one output value, then repeat Step3.

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 Step2 will be successful with probability 2^{-232} . Thus it is likely that there is only one output value for any fixed t after Step2, so there are about 2^8 different values after step2. 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 Step2, and the time of computing each $\triangle$ is about the same as 4-round encryption, so the time complexity of an attack is less than that of $(2\times 2^8\times 2^{232}+2^8\times 2^{224}+2^8\times 2^{217})\times 4/10 < 2^{239}+2^{238}+2^{237}$ 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 tenth, ninth and eighth round and recover the other bytes of k_7 and get the user key of 10-round Camellia. The attack requires less than 2^{14} chosen plaintexts and $2^{239.9}$ encryptions.

6 Pseudorandomness of Primitive-Wise Idealized Camellia

6.1 Primitive-Wise Idealization of Camellia

Let n denote the length of a plaintext which can be written as $n = 16m$, where m is an integer. Now we idealize Camellia as shown in Fig.1, where each f_{ij} is an independent random function from $\{0,1\}^m$ to $\{0,1\}^m$.

6.2 Pseudorandomness of Primitive-Wise Idealized Camellia

Let $P = (L_0, R_0)$ denote the plaintext, (L_i, R_i) denote the output of the *ith* round primitive-wise idealized Camellia. Let $L_i = (L_{i,1}, L_{i,2}, \ldots, L_{i,8})$ and $R_i = (R_{i,1}, R_{i,2}, \ldots, R_{i,8})$, where each of $L_{i,j}$ and $R_{i,j}$ is m bits long.

Theorem 2. *The four round primitive-wise idealized Camellia is not a pseudorandom permutation.*

Proof. Let B_n be the set of permutations over $\{0,1\}^n$ obtained from the four round primitive-wise idealized Camellia. We consider a distinguisher $\mathcal{A}$ as follows.

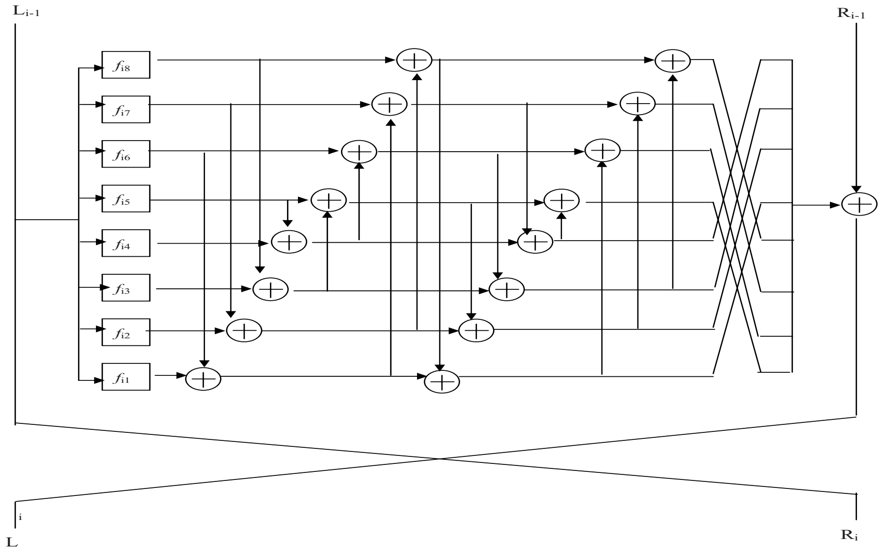

Fig. 1. The i-th round of the primitive-wise idealized Camellia

1. $\mathcal{A}$ randomly chooses two plaintexts $P = (L_0, R_0)$ and $P^* = (L_0^*, R_0^*)$ such that

$$L_0 = L_0^* \ \ and \ \ R_{0,1} \neq R_{0,1}^*, \quad R_{0,j} = R_{0,j}^*(2 \leq j \leq 8) \tag{9}$$

2. $\mathcal{A}$ sends them to the oracle and receives the ciphertexts $C = (L_4, R_4)$ and $C^* = (L_4^*, R_4^*)$ from the oracle.
3. Finally, $\mathcal{A}$ outputs 1 if and only if

$$R_{4,3} \oplus R_{4,4} \oplus R_{4,5} \oplus R_{4,6} \oplus R_{4,7} = R_{4,3}^* \oplus R_{4,4}^* \oplus R_{4,5}^* \oplus R_{4,6}^* \oplus R_{4,7}^*$$
$$R_{4,2} \oplus R_{4,3} \oplus R_{4,5} \oplus R_{4,6} \oplus R_{4,8} = R_{4,2}^* \oplus R_{4,3}^* \oplus R_{4,5}^* \oplus R_{4,6}^* \oplus R_{4,8}^*$$

Suppose that the oracle implements the truly random permutation ensemble P_n. Then it is clear that $p_0 = 2^{-2m}$. Next suppose that the oracle implements the four round primitive-wise idealized Camellia. Using Theorem 1, we get $p_1 = 1$. Therefore, we obtained that

$$Adv_A = \mid p_1 - p_0 \mid \geq 1 - 2^{-2m} \tag{10}$$

which is non-negligible. Hence, the four round primitive-wise idealized Camellia is not a pseudorandom permutation.

We will use the following lemma of which the proof is trivial:

Lemma 1. *Let $f_1, f_2, \ldots, f_t$ be random functions from $\{0,1\}^m$ to $\{0,1\}^m$. If $x = (x_1, x_2, \ldots, x_t)$ and $y = (y_1, y_2, \ldots, y_t)$ are two distinct t-uple of $\{0,1\}^m$, and δ is a given value of $\{0,1\}^m$, then*

$$Pr[f_1(x_1) \oplus \ldots f_t(x_t) \oplus f_1(y_1) \oplus \ldots f_t(y_t) = \delta] \leq 2^{-m}$$

.
We next prove the following theorem.

Theorem 3. *The five round primitive-wise idealized Camellia is a pseudorandom permutation for non-adaptive adversaries.*

Proof. Suppose that $\mathcal{A}$ makes q oracle calls. In the *ith* oracle call, $\mathcal{A}$ sends a plaintexts $P^i = (L_0^i, R_0^i)$ to the oracle and receives the ciphertexts $C^i = (L_5^i, R_5^i)$. Let $L_3^i = (L_{3,1}^i, \ldots, L_{3,8}^i)$ denote the inputs to $(f_{41}, \ldots, f_{48})$ and $L_4^i = (L_{4,1}^i, \ldots, L_{4,8}^i)$ denote the inputs to $(f_{51}, \ldots, f_{58})$.

Without loss of generality, we assume that $P^1, \ldots, P^q$ are all distinct. Let T_{3l} be the event that $L_{3,l}^1, L_{3,l}^2, \ldots, L_{3,l}^q$ are all distinct for $l = 1, \ldots, 8$, and T_3 be the event that all $T_{31}, \ldots, T_{38}$ occur. Let T_{4l} be the event that $L_{4,l}^1, L_{4,l}^2, \ldots, L_{4,l}^q$ are all distinct for $l = 1, \ldots, 8$, and T_4 be the event that all $T_{41}, \ldots, T_{48}$ occur. If T_3 and T_4 occur, then $C^1, \ldots, C^q$ are completely random since $f_{41}, \ldots, f_{48}, f_{51}, \ldots, f_{58}$ are truly random functions. Therefore, Adv_A is upper bounded by

$$Adv_A = \mid p_1 - p_0 \mid \leq 1 - Pr(T_3 \cap T_4) \tag{11}$$

Further, it is easy to see that

$$\begin{aligned} 1 - Pr(T_3 \cap T_4) \leq & \sum_{1 \leq i < j \leq q} Pr(L_{3,1}^i = L_{3,1}^j) + \ldots + \sum_{1 \leq i < j \leq q} Pr(L_{3,8}^i = L_{3,8}^j) + \\ & \sum_{1 \leq i < j \leq q} Pr(L_{4,1}^i = L_{4,1}^j) + \ldots + \sum_{1 \leq i < j \leq q} Pr(L_{4,8}^i = L_{4,8}^j) \end{aligned} \tag{12}$$

Fix $i \neq j$ arbitrarily. We show that all $Pr(L_{3,1}^i = L_{3,1}^j), \ldots, Pr(L_{3,8}^i = L_{3,8}^j)$, $Pr(L_{4,1}^i = L_{4,1}^j), \ldots, Pr(L_{4,8}^i = L_{4,8}^j)$ are sufficiently small. First we show $Pr(L_{3,1}^i = L_{3,1}^j)$ is sufficiently small.

Let E_{2l} be the event that $L_{2,l}^i = L_{2,l}^j$ for $l = 1, \ldots, 8$. Since $P^i \neq P^j$, by Lemma 1 we have $Pr(L_1^i = L_1^j) \leq 2^{-m}$. If $L_1^i \neq L_1^j$, then $(L_{1,1}^i, L_{1,3}^i, L_{1,4}^i, L_{1,6}^i, L_{1,7}^i, L_{1,8}^i) \neq (L_{1,1}^j, L_{1,3}^j, L_{1,4}^j, L_{1,6}^j, L_{1,7}^j, L_{1,8}^j)$ or $(L_{1,1}^i, L_{1,2}^i, L_{1,3}^i, L_{1,5}^i, L_{1,6}^i, L_{1,8}^i) \neq (L_{1,1}^j, L_{1,2}^j, L_{1,3}^j, L_{1,5}^j, L_{1,6}^j, L_{1,8}^j)$. From the 2nd round function of idealized Camellia, we have that

$$\begin{aligned} L_{2,1}^i &= L_{0,1}^i \oplus f_{21}(L_{1,1}^i) \oplus f_{23}(L_{1,3}^i) \oplus f_{24}(L_{1,4}^i) \oplus f_{26}(L_{1,6}^i) \oplus f_{27}(L_{1,7}^i) \oplus f_{28}(L_{1,8}^i) \\ L_{2,3}^i &= L_{0,3}^i \oplus f_{21}(L_{1,1}^i) \oplus f_{22}(L_{1,2}^i) \oplus f_{23}(L_{1,3}^i) \oplus f_{25}(L_{1,5}^i) \oplus f_{26}(L_{1,6}^i) \oplus f_{28}(L_{1,8}^i) \end{aligned}$$

Therefore, by using Lemma 1 we get $Pr(E_{21} \mid L_1^i \neq L_1^j) \leq 2^{-m}$ or $Pr(E_{23} \mid L_1^i \neq L_1^j) \leq 2^{-m}$, hence $Pr(E_{21}) \leq Pr(L_1^i = L_1^j) + Pr(E_{21} \mid L_1^i \neq L_1^j) \leq 2^{-m+1}$ or $Pr(E_{23}) \leq Pr(L_1^i = L_1^j) + Pr(E_{23} \mid L_1^i \neq L_1^j) \leq 2^{-m+1}$, and therefore, $Pr(E_{21} \cap E_{23}) \leq 2^{-m+1}$.

Similarly, from Lemma 1 and the following equation

$$L_{3,1}^i = L_{1,1}^i \oplus f_{31}(L_{2,1}^i) \oplus f_{33}(L_{2,3}^i) \oplus f_{34}(L_{2,4}^i) \oplus f_{36}(L_{2,6}^i) \oplus f_{37}(L_{2,7}^i) \oplus f_{38}(L_{2,8}^i)$$

we have $Pr(L_{3,1}^i = L_{3,1}^j \mid \overline{E_{21} \cap E_{23}}) \leq 2^{-m}$. Hence, we have

$$\begin{aligned} & Pr(L_{3,1}^i = L_{3,1}^j) \\ & = Pr(L_{3,1}^i = L_{3,1}^j \mid E_{21} \cap E_{23}) Pr(E_{21} \cap E_{23}) + Pr(L_{3,1}^i = L_{3,1}^j \mid \overline{E_{21} \cap E_{23}}) Pr(\overline{E_{21} \cap E_{23}}) \\ & \leq Pr(E_{21} \cap E_{23}) + Pr(L_{3,1}^i = L_{3,1}^j \mid \overline{E_{21} \cap E_{23}}) \\ & \leq 2^{-m+1} + 2^{-m} = 3 \times 2^{-m} \end{aligned} \tag{13}$$

Similarly for $l = 2, \ldots, 8$, we can get $Pr(L^i_{3,l} = L^j_{3,l}) \leq 3 \times 2^{-m} (l = 2, \ldots, 8)$.

Next we show $Pr(L^i_{4,l} = L^j_{4,l})$ is sufficiently small for $l = 1, \ldots, 8$. For simplicity, we only consider the case $Pr(L^i_{4,1} = L^j_{4,1})$.

Let E_{3l} be the event that $L^i_{3,l} = L^j_{3,l}$ for $l = 1, \ldots, 8$. Let $W_1 = E_{31} \cap E_{33} \cap E_{34} \cap E_{36} \cap E_{37} \cap E_{38}$. Because

$$L^i_{4,1} = L^i_{2,1} \oplus f_{41}(L^i_{3,1}) \oplus f_{43}(L^i_{3,3}) \oplus f_{44}(L^i_{3,4}) \oplus f_{46}(L^i_{3,6}) \oplus f_{47}(L^i_{3,7}) \oplus f_{48}(L^i_{3,8})$$
$$L^j_{4,1} = L^j_{2,1} \oplus f_{41}(L^j_{3,1}) \oplus f_{43}(L^j_{3,3}) \oplus f_{44}(L^j_{3,4}) \oplus f_{46}(L^j_{3,6}) \oplus f_{47}(L^j_{3,7}) \oplus f_{48}(L^j_{3,8})$$

we have $Pr(L^i_{4,1} = L^j_{4,1} \mid \overline{W_1}) \leq 2^{-m}$. Therefore, we obtain

$$\begin{aligned} &Pr(L^i_{4,1} = L^j_{4,1}) = Pr(L^i_{4,1} = L^j_{4,1} \mid W_1)Pr(W_1) + Pr(L^i_{4,1} = L^j_{4,1} \mid \overline{W_1})Pr(\overline{W_1}) \\ &\leq Pr(W_1) + Pr(L^i_{4,1} = L^j_{4,1} \mid \overline{W_1}) \\ &\leq Pr(L^i_{3,1} = L^j_{3,1}) + 2^{-m} \leq 4 \times 2^{-m} \end{aligned} \tag{14}$$

Similarly for $l = 2, \ldots, 8$, we have $Pr(L^i_{4,l} = L^j_{4,l}) \leq 4 \times 2^{-m}$.

Since we have $\binom{q}{2}$ choices of (i, j) pairs, so we have

$$\begin{aligned} 1 - Pr(T_3 \cap T_4) \leq &\sum_{1 \leq i < j \leq q} Pr(L^i_{3,1} = L^j_{3,1}) + \ldots + \sum_{1 \leq i < j \leq q} Pr(L^i_{3,8} = L^j_{3,8}) + \\ &\sum_{1 \leq i < j \leq q} Pr(L^i_{4,1} = L^j_{4,1}) + \ldots \sum_{1 \leq i < j \leq q} Pr(L^i_{4,8} = L^j_{4,8}) \\ &\leq \binom{q}{2} \times 8 \times 3 \times 2^{-m} + \binom{q}{2} \times 8 \times 4 \times 2^{-m} \\ &< \frac{28q^2}{2^m} \end{aligned} \tag{15}$$

Since $q = poly(n), m = \frac{n}{16}$, we have that Adv_A is negligible for any $\mathcal{A}$. This shows that the five round primitive-wise idealized Camellia is a pseudorandom permutation for non-adaptive adversaries.

Similar to the above, we can prove the following corollary.

Corollary 1. *The five round primitive-wise idealized Camellia is a super-pseudorandom permutation for non-adaptive adversaries.*

7 Concluding Remarks

In this paper we have proposed some 4-round distinguishers of Camellia, and discussed the security of Camellia by using the 4-round distinguishers and collision-searching techniques. 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. Furthermore, we have shown that the four round primitive-wise idealized Camellia is not pseudorandom permutation and the five round primitive-wise idealized Camellia is super-pseudorandom permutation for non-adaptive adversaries.

References

1. K.Aoki,T.Ichikawa,M.Kanda,M.Matsui,S.Moriai,J.Nakajima and T.Tokita, "Specification of Camellia-a 128-bit Block Cipher," *Selected Areas in Cryptography - SAC'2000*, Springer-Verlag 2000,pp. 183-191.
2. http://www.cryptonessie.org
3. T.Kawabata, T.Kaneko, "A study on higher order differential attack of Camellia," *Proceedings of the 2nd NESSIE workshop*,2001.
4. Y.Hatano,H.Sekine, and T.Kaneko, "Higher order differential attack of Camellia(II),"*Selected Areas in Cryptography-SAC'02*, LNCS 2595,Springer-Verlag 2002, pp.39-56.
5. S. Lee, S. Hong, S. Lee, J. Lim and S. Yoon, "Truncated Differential Cryptanalysis of Camellia", *Information Security and Cryptology-ICISC'01*,LNCS 2288, Springer-Verlag,2001,pp.32-38.
6. M.Sugita,K.Kobara, and H.Imai, "Security of reduced version of the block cipher Camellia against truncated and impossible differential cryptanalysis," *Advances in Cryptology– Asiacrypt'01*,LNCS 2248,Springer-Verlag 2001, pp.193-207.
7. T.Shirai,S.Kanamaru,and G.Abe, "Improved upper bounds of differential and linear characteristic probability for Camellia," *Fast Software Encryption-FSE'02*,LNCS 2365,Springer-Verlag 2002,pp.128-142.
8. He Ye-ping and Qing Si-han, "Square attack on Reduced Camellia Cipher," *Information and Communication Security-ICICS'01*,LNCS 2229, Springer-Verlag 2001,pp.238-245.
9. Y.Yeom, S.Park, and I. Kim, "On the security of Camellia against the square attack," *Fast Software Encryption-FSE'02*,LNCS 2356, Springer-Verlag 2002,pp.89-99.
10. Y.Yeom, I. Park, and I. Kim, "A study of Integral type cryptanalysis on Camellia," *The 2003 Symposium on Cryptography and Information Security-SCIS'03*.
11. M. Luby and C. Rackoff, " How to construct pseudorandom permutations from pseudorandom functions," *SIAM Journal on Computing*, Vol.17,No.2,(1988),pp.373-386.
12. J.Patarin, " New results on pseudorandom permutation generators based on the DES Scheme," *Advances in Cryptology–Crypto'91*, Springer-Verlag 1991,pp.72-77.
13. U.M.Maurer, "A simplified and generalized treatment of Luby-Rackoff pseudorandom permutation generators," *Advances in Cryptology-Eurocrypt'92*, LNCS 658,Springer-Verlag 1992,pp.239-255.
14. S. Vaudenay, "Provable security for block ciphers by decorrelation," *In Proc. of STACS'98*, LNCS 1373, Springer-Verlag 1998,pp.249-275.
15. T.Iwata and K. Kurosawa, "On the Pseudorandomness of the AES Finalists-RC6 and Serpent," *Fast Software Encryption–FES'2000*, LNCS 1978, Springer-Verlag 2000,pp.231-243,.
16. M.Naor and O.Reingold, "On the construction of pseudorandom permutations Luby-Rackoff revisited," *Journal of Cryptology*, Vol.12, No.1,pp.29-66,1999.