On the Security of CAMELLIA against the Square Attack

Yongjin Yeom, Sangwoo Park, and Iljun Kim

National Security Research Institute
161 Gajeong-dong, Yuseong-gu, Daejeon, 305-350 Korea
{yjyeom, psw}@etri.re.kr, ijkim@dingo.etri.re.kr

Abstract. Camellia is a 128 bit block cipher proposed by NTT and Mitsubishi. We discuss the security of Camellia against the square attack. We find a 4 round distinguisher and construct a basic square attack. We can attack 5 round Camellia by guessing one byte subkey and using 2^{16} chosen plaintexts. Cosidering the key schdule, we may extend this attack up to 9 round Camellia including the first FL/FL^{-1} function layer.

1 Introduction

Camellia[5] is a 128-bit block cipher which was announced by NTT and Mitsubishi in 2000. It has the modified Feistel structure with irregular rounds, so called the FL/FL^{-1} function layer. The round function is based on that of the block cipher E2[13] by NTT whereas the FL/FL^{-1} layer comes from MISTY[18] by Mitsubishi. Camellia was submitted to the standardization and the evaluation projects such as ISO/IEC JTC 1/SC 27, CRYPTREC, and NESSIE. Recently, Camellia was selected as an algorithm for the second phase of the NESSIE project.

Currently, the most efficient methods analyzing Camellia are truncated differential cryptanalysis and higher order differential attack. Kanda and Matsumoto[12] studied the security against truncated differential cryptanalysis from the designer's standpoint. They found the upper bound of the best bytewise characteristic probability and proved that Camellia with more than 11 rounds are secure against truncated differential cryptanalysis. Most analyses on Camellia consider simplified version without FL/FL^{-1} function layers. For instance, S. Lee et al.[15] attacked eight round Camellia using truncated differential cryptanalysis. M. Sugita et al.[19] found a nontrivial 9 round bytewise characteristics and a seven round impossible differential for Camellia. Kawabata and Kaneko[11] showed that Camellia can be attacked by higher order differential attack up to 10 rounds. Some other analyses can be found in [5].

The square attack was a dedicated attack on the block cipher SQUARE[6] and applied to block ciphers of the SPN structure such as Rijndael[7, 8, 16], CRYPTON[9], and Hierocrypt[3]. In order to apply the square attack on the Feistel structure, Lucks[17] introduced the saturation attack, as a variation of the square attack. He analyzed the Twofish algorithm of the modified Feistel structure. Recently, Y. He and S. Qing[10] showed that six round Camellia are breakable by square attack.

In this paper, we apply the square attack to Camellia including FL/FL^{-1} function layers. We suggest the basic attack which breaks 5 round Camellia using 2^{16} chosen plaintexts and 2^8 key guessings. Also, the key schedule is considered so that the square attack on 256 bit Camellia is faster than exhaustive key search up to 9 rounds.

Section 2 briefly describes the structure of Camellia. A basic attack based on a 4 round distinguisher is given in Section 3 and extensions of the basic attack up to 9 round Camellia is proposed in Section 4.

2 Description of the 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 in order to thwart future unknown attacks. 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.

One round substitution and permutation structure is adopted as the round function F. Let $X_L^{(r)}$ and $X_R^{(r)}$ be the left and the right halves of the r round inputs, respectively, and $k^{(r)}$ be the r round subkey. Then the Feistel structure of Camellia can be written as

$$\begin{aligned} X_L^{(r+1)} &= X_R^{(r)} \oplus F(X_L^{(r)}, k^{(r)}), \\ X_R^{(r+1)} &= X_L^{(r)}. \end{aligned}$$

In the following substitution S, the four types of S-boxes s_1, s_2, s_3, and s_4 are used. Each of them is affinely equivalent to an inversion over $GF(2^8)$. Actually, s_2, s_3, s_4 are variations of s_1. The only property of S-boxes used for the square attack is that they are one-to-one functions. The substitution function $S : \{0,1\}^{64} \to \{0,1\}^{64}$ which consists of S-boxes is also a one-to-one function defined by

$$(x_1, \ldots, x_8) \xrightarrow{S} (s_1(x_1), s_2(x_2), s_3(x_3), s_4(x_4), s_2(x_5), s_3(x_6), s_4(x_7), s_1(x_8))\,.$$

The permutation function $P : \{0,1\}^{64} \to \{0,1\}^{64}$ maps $(z_1, \ldots, z_8)$ to $(z'_1, \ldots, z'_8)$ defined by

$$\begin{aligned} z'_1 &= z_1 \oplus z_3 \oplus z_4 \oplus z_6 \oplus z_7 \oplus z_8, \\ z'_2 &= z_1 \oplus z_2 \oplus z_4 \oplus z_5 \oplus z_7 \oplus z_8, \\ z'_3 &= z_1 \oplus z_2 \oplus z_3 \oplus z_5 \oplus z_6 \oplus z_8, \\ z'_4 &= z_2 \oplus z_3 \oplus z_4 \oplus z_5 \oplus z_6 \oplus z_7, \\ z'_5 &= z_1 \oplus z_2 \oplus z_6 \oplus z_7 \oplus z_8, \end{aligned}$$

$$z'_6 = z_2 \oplus z_3 \oplus z_5 \oplus z_7 \oplus z_8,$$
$$z'_7 = z_3 \oplus z_4 \oplus z_5 \oplus z_6 \oplus z_8,$$
$$z'_8 = z_1 \oplus z_4 \oplus z_5 \oplus z_6 \oplus z_7.$$

We can also express the function P in the matrix form:

$$\begin{pmatrix} z'_8 \\ z'_7 \\ z'_6 \\ z'_5 \\ z'_4 \\ z'_3 \\ z'_2 \\ z'_1 \end{pmatrix} = \begin{pmatrix} 0\,1\,1\,1\,1\,0\,0\,1 \\ 1\,0\,1\,1\,1\,1\,0\,0 \\ 1\,1\,0\,1\,0\,1\,1\,0 \\ 1\,1\,1\,0\,0\,0\,1\,1 \\ 0\,1\,1\,1\,1\,1\,1\,0 \\ 1\,0\,1\,1\,0\,1\,1\,1 \\ 1\,1\,0\,1\,1\,0\,1\,1 \\ 1\,1\,1\,0\,1\,1\,0\,1 \end{pmatrix} \begin{pmatrix} z_8 \\ z_7 \\ z_6 \\ z_5 \\ z_4 \\ z_3 \\ z_2 \\ z_1 \end{pmatrix}.$$

The round function $F : \{0,1\}^{64} \times \{0,1\}^{64} \to \{0,1\}^{64}$ is defined as a composition of S and P functions as follows:

$$(X, k) \xrightarrow{F} P(S(X \oplus k)).$$

At every 6 rounds the functions FL and FL^{-1} are inserted. We denote bitwise-and, bitwise-or operations by $\cap$, $\cup$ and a n bit rotation by $\lll n$. The left 64 bit half (X_L, X_R) is mapped to (Y_L, Y_R) by the function FL.

$$(X_L \| X_R, kl_L \| kl_R) \xrightarrow{FL} (Y_L, Y_R),$$

where

$$Y_R = ((X_L \cap kl_L) \lll 1) \oplus X_R,$$
$$Y_L = (Y_R \cup kl_R) \oplus X_L.$$

and the inverse FL^{-1} of FL is used for the right half as follows:

$$(Y_L \| Y_R, kl_L \| kl_R) \xrightarrow{FL^{-1}} (X_L, X_R),$$

where

$$X_L = (Y_R \cup kl_R) \oplus Y_L,$$
$$X_R = ((X_L \cap kl_L) \lll 1) \oplus Y_R.$$

The key schedule of Camellia will be briefly considered in Section 4.

3 Basic Square Attack

The concept of the Λ-set, which was introduced by Daemen et al. [6], plays an important role in the square attack.

Let $\mathcal{F}$ be a collection of state bytes $X = (x_1, x_2, \ldots, x_n)$ where x_i is the i-th byte of X. If the i-th bytes of elements in $\mathcal{F}$ are different one another, the i-th byte is called an 'active' byte. Likewise, the j-th byte is 'passive' (or fixed), if the j-th bytes of states in $\mathcal{F}$ have the same value.

A collection $\mathcal{F}$ of 256 state bytes is called a Λ-set, if every byte of $\mathcal{F}$ is either active or passive. More precisely, if X and Y are arbitrary elements of a Λ-set $\mathcal{F}$, then

$$\begin{cases} x_i \neq y_i, & \text{if the } i\text{-th byte is active,} \\ x_i = y_i, & \text{otherwise,} \end{cases}$$

where x_i and y_i are the i-th byte of X and Y, respectively. Note that an arbitrary collection $\mathcal{F}$ has non-active and non-passive bytes in general. The i-th byte in a collection $\mathcal{F}$ is called balanced, if $\bigoplus_{X \in \mathcal{F}} x_i = 0$.

The main operations of the Camellia are bitwise exclusive-or(XOR) and substitution using one-to-one 8×8 S-boxes s_i. If an active(passive) byte of a Λ-set is used as an input of S-boxes s_i, then the output is also active(passive). But the output of s_i is not necessarily balanced when its input is balanced.

Some properties of XOR operation can be summarized as shown in Table 1.

Table 1. Some properties of XOR operation

XOR($\oplus$)	active byte	passive byte	balanced byte
active byte	balanced byte	active byte	balanced byte
passive byte	active byte	passive byte	balanced byte
balanced byte	balanced byte	balanced byte	balanced byte

3.1 Four Round Distinguishers

Let $X_L^{(r)}$, $X_R^{(r)}$ be the left and the right inputs of the r-th round. Then we can construct a 4 round distinguisher as follows:

Choose

$$X_L^{(1)} = (\alpha_1, \alpha_2, \ldots, \alpha_8), \quad X_R^{(1)} = (A, \beta_2, \ldots, \beta_8)$$

as a Λ-set $\mathcal{F}$ of 256 input plaintexts, where α_i, β_j are constants and A is an active bytes of $\mathcal{F}$. Because $X_L^{(1)}$ is passive, the output of the first round function F is also passive. Thus, the input of the 2nd round can be written of the form

$$X_L^{(2)} = (A, \gamma_2, \ldots, \gamma_8), \quad X_R^{(2)} = (\alpha_1, \alpha_2, \ldots, \alpha_8),$$

where γ_i are constants. In the 2nd round, the input $X_L^{(2)}$ for F is transformed as follows:

$$(A, \gamma_2, \ldots, \gamma_8) \xrightarrow{F} (B, C, D, \delta_4', E, \delta_6', \delta_7', F),$$

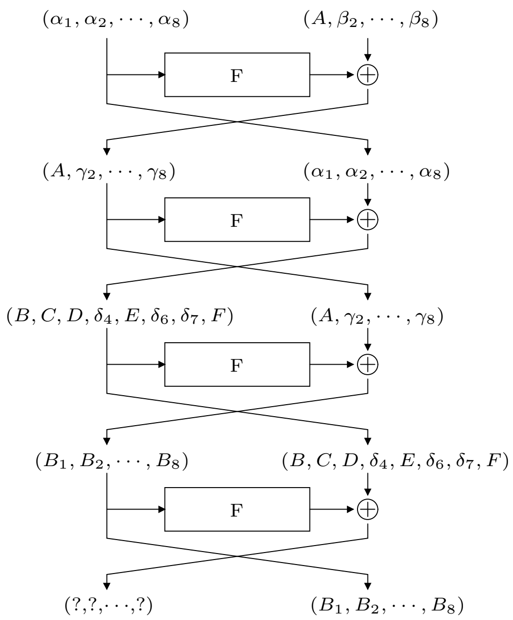

Fig. 1. A four round distinguisher

where B, C, D, E, and F are active. Thus, we have

$$X_L^{(3)} = (B, C, D, \delta_4, E, \delta_6, \delta_7, F), \quad X_R^{(3)} = (A, \gamma_2, \dots, \gamma_8)$$

as an input for the 3rd round. Applying the 3rd round function to $X_L^{(3)}$, we expect that each state byte in the left half of the input for the 4th round is balanced. This implies that all bytes in the right half of the 4th round output are balanced. Thus, we obtain a 4 round distinguisher.

Note that only 2 round functions are effectively activated in this 4 round distinguisher. This corresponds to the 2 round distinguisher for the SPN structure. If we change the position of the active byte in $X_R^{(1)}$, we obtain 8 different distinguishers.

3.2 Five Round Square Attack

From the above distinguisher, we can construct a basic square attack on the 5 round Camellia without pre- and post-whitenings.

Step 1. Guess the 1st byte k of the first round key.
Step 2. As input plaintexts, choose a Λ-set $\mathcal{F}$ of the form

$$\mathcal{F} = \{(X_L(i), X_R(i)) | 0 \le i \le 255\}, \tag{1}$$

where for arbitrarily chosen constants α_i, β_j,

$$X_L(i) = (i, \alpha_2, \dots, \alpha_8),$$
$$X_R(i) = (s_1(i \oplus k), s_1(i \oplus k), s_1(i \oplus k), \beta_4, s_1(i \oplus k), \beta_6, \beta_7, s_1(i \oplus k)).$$

Step 3. If k is a correct key, we can expect the left half of the 2nd round inputs consists of constant states. For example, the 1st output byte z'_1 of the 1st round function is $z'_1(i) = z_1 \oplus z_3 \oplus z_4 \oplus z_6 \oplus z_7 \oplus z_8$, where $z_1 = s_1(i \oplus k)$ and $z_3, z_4, \ldots, z_8$ are constants solely depending upon $\alpha_2, \ldots, \alpha_8$. Taking exclusive-or of $z'_1(i)$ and the 1st byte $s_1(i \oplus k)$ of $X_R(i)$, we have a constant byte which is independent of i. Using the same argument, we can show that each byte of $X_L^{(2)}$ is a constant.

Step 4. The right half $X_R^{(2)}$ of the 2nd round input is identical to $X_L^{(1)}$. Thus we can use the 4 round distinguisher previously mentioned.

Step 5. Let C_L, C_R be corresponding outputs of the input Λ-set $\mathcal{F}$ which is chosen in Step 2. If all bytes of C_R are balanced, then we can accept k as the correct key. Otherwise, go to Step 1 and guess another key and repeat.

For this 5 round attack, we use 2^8 times 5 round encryptions in every key guessing. A wrong key can pass the balance test with a probability 2^{-64}, i.e. negligible. Thus, the number of plaintexts needed for this attack is $2^8 \times 2^8 = 2^{16}$, and the same number of 5 round encryptions is required.

3.3 Six Round Square Attack

We can extend this basic attack to 6 round Camellia by adding a round at the beginning. The key idea for 6 round attack is to choose a collection of plaintexts whose 1 round output is a Λ-set $\mathcal{F}$ as described in (1). To do this, we assume additional 5 bytes of the first round key.

Let $k_1^{(2)}$ be the first byte of the second round key and $k_i^{(1)}$ the i-th byte of the 1st round key. Suppose that we guess $k_1^{(1)}$, $k_2^{(1)}$, $k_3^{(1)}$, $k_5^{(1)}$, $k_8^{(1)}$, and $k_1^{(2)}$, correctly. Then we can find a set $\mathcal{F}^{(1)}$ of plaintexts so that the second round input is a Λ-set $\mathcal{F}^{(2)}$ of the form

$$\mathcal{F}^{(2)} = \left\{ \left(X_L^{(2)}(i), X_R^{(2)}(i) \right) \middle| 0 \le i \le 255 \right\}, \tag{2}$$

where

$$\begin{aligned} X_L^{(2)}(i) &= (i, \alpha_2, \ldots, \alpha_8), \\ X_R^{(2)}(i) &= (s(i), s(i), s(i), \beta_4, s(i), \beta_6, \beta_7, s(i)), \\ s(i) &= s_1(i \oplus k_1^{(2)}). \end{aligned}$$

It is easy to see that the left half $X_L^{(1)}(i)$ of an input Λ-set

$$\mathcal{F}^{(1)} = \left\{ \left(X_L^{(1)}(i), X_R^{(1)}(i) \right) \middle| 0 \le i \le 255 \right\} \tag{3}$$

should be exactly equal to $X_R^{(2)}(i)$ for each i and the right half $X_R^{(1)}(i)$ of that can be determined by the subkeys $k_1^{(1)}$, $k_2^{(1)}$, $k_3^{(1)}$, $k_5^{(1)}$, and $k_8^{(1)}$. For example, the 1st output byte z'_1 of the 1st round function can be written as

$$\begin{aligned} z'_1(i) = s_1(s(i) \oplus k_1^{(1)}) \oplus s_3(s(i) \oplus k_3^{(1)}) \oplus s_4(\beta_4 \oplus k_4^{(1)}) \oplus s_3(\beta_6 \oplus k_6^{(1)}) \\ \oplus s_4(\beta_7 \oplus k_7^{(1)}) \oplus s_1(s(i) \oplus k_8^{(1)}) \\ = s_1(s(i) \oplus k_1^{(1)}) \oplus s_3(s(i) \oplus k_3^{(1)}) \oplus s_1(s(i) \oplus k_8^{(1)}) \oplus \beta, \end{aligned}$$

where $\beta = s_4(\beta_4 \oplus k_4^{(1)}) \oplus s_3(\beta_6 \oplus k_6^{(1)}) \oplus s_4(\beta_7 \oplus k_7^{(1)})$ is independent of i. Thus we can choose

$$i \oplus s_1(s(i) \oplus k_1^{(1)}) \oplus s_3(s(i) \oplus k_3^{(1)}) \oplus s_1(s(i) \oplus k_8^{(1)})$$

as the first byte S_1 of $X_R^{(1)}(i)$ so that the first byte of $X_L^{(2)}(i)$ is active. Similarly, remaining bytes of $X_R^{(1)}(i)$ can be calculated.

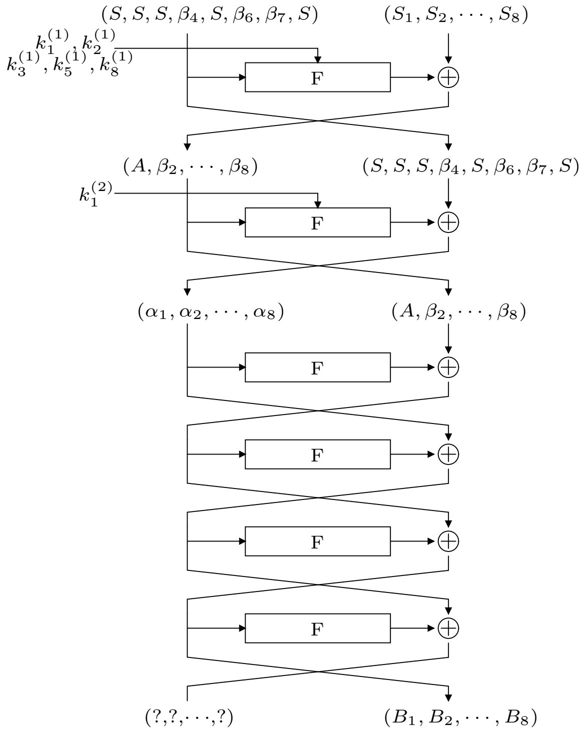

Fig. 2. A square attack on 6 round Camellia

For each Λ-set $\mathcal{F}^{(1)}$ determined by the 6 byte subkeys guessing, we can check the balance of the right half of the 6th round output. In this check, a wrong key can be accepted with a probability 2^{-64}. Thus, the 6 round attack requires $2^8 \times 2^{48}$ plaintexts and 2^{48} subkeys guessing.

By adding a round at the end, we obtain another square attack(See [10]) on 6 round Camellia. But it makes the attack exceeding 6 rounds with the FL/FL^{-1} layer much harder.

4 Key Schedule and Extension of the Basic Attack

4.1 Key Schedule of Camellia

To extend the basic attack on over 6 round Camellia with FL/FL^{-1} function layer, we consider the key schedule. The round keys are bitwise rotations of K_L, K_R, K_A, and K_B which are calculated from the master keys K_L and K_R. In this calculation, they use the reduced rounds of Camellia with a constant key. We do not describe the details here(See [1]). Table 2 shows how to select 1–10 round keys from K_L, K_R, K_A, and K_B.

Table 2. Subkeys for 192/256-bit secret key

	subkey	value
F (Round 1)	$k^{(1)}$	$(K_B \lll_0)_{L(64)}$
F (Round 2)	$k^{(2)}$	$(K_B \lll_0)_{R(64)}$
F (Round 3)	$k^{(3)}$	$(K_R \lll_{15})_{L(64)}$
F (Round 4)	$k^{(4)}$	$(K_R \lll_{15})_{R(64)}$
F (Round 5)	$k^{(5)}$	$(K_A \lll_{15})_{L(64)}$
F (Round 6)	$k^{(6)}$	$(K_A \lll_{15})_{R(64)}$
FL	$kl_{1(64)}$	$(K_R \lll_{30})_{L(64)}$
FL^{-1}	$kl_{2(64)}$	$(K_R \lll_{30})_{R(64)}$
F (Round 7)	$k^{(7)}$	$(K_B \lll_{30})_{L(64)}$
F (Round 8)	$k^{(8)}$	$(K_B \lll_{30})_{R(64)}$
F (Round 9)	$k^{(9)}$	$(K_L \lll_{45})_{L(64)}$
F (Round 10)	$k^{(10)}$	$(K_L \lll_{45})_{R(64)}$

Note that seven and eight round keys $k^{(7)}$ and $k^{(8)}$ are nothing but 30 bit rotations of the first and the second round keys, respectively. This property will be used to attack more than 6 rounds of Camellia.

4.2 An Observation on the FL/FL^{-1} Layer

Consider the reduced model of 6 round Camellia with the FL/FL^{-1} layer. As mentioned previously, if we assume 6 byte subkeys correctly, every byte of the right half of the 6th round outputs is balanced. By guessing additional 7 bits of subkey kl_2, we can partially invert FL/FL^{-1} layer.

Let (C_L, C_R) be an output of the FL^{-1} function. Then we can determine the leftmost 7 bits of the input $y_{R,4}$ for the FL^{-1} function from two bytes $x_{R,4}$

and $x_{L,4}$ using the relation

$$y_{R,4} = x_{R,4} \oplus (x_{L,4} \cap kl_{2L,4}) \lll 1,$$

where $y_{R,4}$, $x_{R,4}$, $x_{L,4}$ and $kl_{2L,4}$ are the fourth bytes of Y_R, X_R, X_L and kl_{2L}, respectively.

4.3 Square Attacks on 256 bit Camellia up to 7, 8, and 9 Rounds

Now, we consider 256 bit Camellia and its key schedule. We can extend the previous observation on the FL/FL^{-1} layer to a 7 round square attack. We should assume 7 byte subkeys $k_1^{(7)}, \ldots, k_7^{(7)}$ out of the seventh round key $k^{(7)}$ to determine two byte outputs $x_{R,4}$ and $x_{L,4}$ of FL^{-1} function. But the 7th round key $k^{(7)}$ is nothing but the 30 bit left rotation of the first round key $k^{(1)}$. In fact, we only guess additional 18 bits. Thus, the number of bits we need to guess for 7 round Camellia is $73 = 58$(round 1, 7) + 8(round 2) + 7(FL^{-1} layer). When we apply the square attack to 7 round Camellia, we can check only 7 bits of the 6th round outputs. Thus, a wrong key can pass the test with a probability 2^{-7}. We need 11 Λ-sets as input plaintexts to eliminate a wrong key. The algorithm to attack 7 round Camellia can be summarized as follows:

Step 1. Guess 6 byte subkeys $k_1^{(1)}$, $k_2^{(1)}$, $k_3^{(1)}$, $k_5^{(1)}$, $k_8^{(1)}$, and $k_1^{(2)}$ of the first and the second round.

Step 2. Prepare 11 Λ-sets as plaintexts so that inputs of the third round are of the form

$$X_L^{(3)} = (\alpha_1, \alpha_2, \ldots, \alpha_8), \quad X_R^{(3)} = (A, \beta_2, \ldots, \beta_8).$$

Note that the only byte A of them is active. Thus, we expect the right half of the 6th round outputs is balanced, if key guessing is correct.

Step 3. Partially decrypt outputs and test the balance of them.

- **3.1.** Guess additional 25 bit subkeys for FL^{-1} and the 7th round.
- **3.2.** Decrypt ciphertexts and determine 7 bits of the right half of the 6th round outputs.
- **3.3.** Check if this 7 bits are balanced for all 11 Λ-sets.
- **3.4.** If so, accept 73 bit subkeys as a correct key.
- **3.5.** Otherwise, discard 25 bit subkeys guessed in Step 3.1 and choose another 25 bits. If all possible 25 bits are checked, go to Step 1 and repeat Step 2 and Step 3.

For each subkey candidate, we need to encrypt 11 Λ-sets, which costs

$$2^{48}(\text{subkeys}) \times 2^{8}(\Lambda\text{-set size}) \times 11(\text{the number of } \Lambda\text{-sets})$$

encryptions of 7 round. Also, one round decryptions and partial invertings of FL^{-1} function are needed for them with the computational complexity

$$2^{73}(\text{subkeys}) \times 2^{8}(\Lambda\text{-set size}) \times 11(\text{the number of } \Lambda\text{-sets}).$$

Thus, total amount of cipher execution is approximately $2^{81.7}$ encryptions.

With helpful comments of anonymous referees, this attack could be improved as follows: for a given 6 byte subkeys of the first and the second rounds, first prepare 4 Λ-sets and see whether 2^{25} subkeys for the FL^{-1} and the 7th round pass balanced tests. With probability 1/8, one of the subkeys can pass these tests. In that case, the remaining 7 Λ-sets can be exercised. This procedure reduced the plaintext cost to $(4 + 7/8)2^{56}$. Also, the time complexity can be reduced to $2^{80.2}$ encryptions.

By assuming all round keys in round 8 and 9, we construct an attack algorithm on 9 round Camellia. One byte of the 8th round key is already guessed in the second round. Therefore, 193 bits of subkey guessing is needed to attack 9 round Camellia. It is of course infeasible but faster than exhaustive key search.

5 Conclusion

We have discussed the security of Camellia against the square attack. We have treated the reduced round Camellia without pre- and post-whitenings including the FL/FL^{-1} layers. The key schedule has been considered to reduce the number of subkey guess and how to treat the FL/FL^{-1} function layers has been presented.

Table 3. Summary of attacks on 256 bit Camellia

Rounds	FL/FL^{-1}	Methods	Plaintexts	Time	Comments
5	N/A	Square Attack	$2^{10.3}$	2^{48}	He & Qing[10](Pre-Whitening)
5	N/A	Square Attack	2^{16}	2^{16}	This paper
6	N/A	Higher Order DC	2^{17}	$2^{19.4}$	Kawabata & Kaneko[11]
6	N/A	Square Attack	$2^{11.7}$	2^{112}	He & Qing[10](Pre-Whitening)
6	N/A	Square Attack	2^{56}	2^{56}	This paper
7	×	Higher Order DC	2^{19}	$2^{61.2}$	Kawabata & Kaneko[11]
7	×	Truncated DC	$2^{82.6}$	192	S. Lee et al.[15]
7	○	Square Attack	$2^{58.3}$	$2^{80.2}$	This paper
8	×	Higher Order DC	2^{20}	2^{126}	Kawabata & Kaneko[11]
8	×	Truncated DC	$2^{83.6}$	$2^{55.6}$	S. Lee et al.[15]
8	○	Square Attack	$2^{59.7}$	$2^{137.6}$	This paper
9	×	Higher Order DC	2^{21}	$2^{190.8}$	Kawabata & Kaneko[11]
9	○	Square Attack	$2^{60.5}$	$2^{202.2}$	This paper
10	×	Higher Order DC	2^{21}	$2^{254.7}$	Kawabata & Kaneko[11]

Table 3 summarizes attacks on 256 bit Camellia by the number of rounds. Time complexities in the table is the number of encryptions.

Up to 9 rounds, the square attack is a faster way to attack Camellia than the brute force key search.

Acknowledgment

We would like to thank anonymous referees for their helpful comments and suggestions. As mentioned at the end of Section 4, we could reduce the plaintext requirement as well as time complexity according to the anonymous advice.

References

1. K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima and T. Tokita. *Camellia: A 128–Bit Block Cipher Suitable for Multiple Platforms*. Proceedings of Selected Areas in Cryptography (to appear in LNCS by Springer-Verlag) (2000) 41–54.
2. K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima and T. Tokita. *Camellia – A 128-bit Block Cipher*. Technical Report of IEICE, ISEC2000–6 (2000).
3. P. Barreto, V. Rijmen, J. Nakahara Jr., B. Preneel, J. Vandewalle and H. Kim. *Improved Square Attacks against Reduced–Round Hierocrypt*. Proceedings of Fast Software Encryption (to appear in LNCS by Springer-Verlag) (2001) 173–182.
4. E. Biham. *Cryptanalysis of Ladder-DES*. Fast Software Encryption, LNCS 1267, Springer-Verlag (1997) 134–138.
5. Camellia Home Page. `http://info.isl.ntt.co.jp/camellia/`.
6. J. Daemen, L. R. Knudsen and V. Rijmen. *The Block Cipher SQUARE*. Fast Software Encryption, LNCS 1267, Springer-Verlag (1997) 149–165.
7. J. Daemen and V. Rijmen. *AES Proposal: Rijndael (Document version 2)*. AES Submission (1999).
8. N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner and D. Whiting. *Improved Cryptanalysis of Rijndael*. Fast Software Encryption, LNCS 1978, Springer-Verlag (2000) 213–230.
9. C. D'Halluin, G. Bijnens, V. Rijmen and B. Preneel. *Attack on the Six Rounds of CRYPTON*. Fast Software Encryption, LNCS 1636, Springer-Verlag (1999) 46–59.
10. Y. He and S. Qing, *Square Attack on Reduced Camellia Cipher*. ICICS 2001, LNCS 2229, Springer-Verlag (2001) 238–245.
11. T. Kawabata and T. Kaneko. *A Study on Higher Order Differential Attack of Camellia*. the 2nd open NESSIE workshop (2001).
12. M. Kanda and T. Matsumoto. *Security of Camellia against Truncated Differential Cryptanalysis*. Proceedings of Fast Software Encryption (to appear in LNCS by Springer-Verlag) (2001) 298–312.
13. K. Kanda, S. Moriai, K. Aoki, H. Ueda, M. Ohkubo, Y. Takashima, K. Ohta and T. Matsumoto. *A New 128-bit Block Cipher E2*. Technical Report ISEC98-12, The Institute of Electronics, Information and Communication Engineers. (1998).
14. L. R. Knudsen. *Analysis of Camellia*. a contribution for ISO/IEC JTC1 SC27. `http://info.isl.ntt.co.jp/camellia/Publications/knudsen.ps` (2000).
15. S. Lee, S. Hong, S. Lee, J. Lim and S. Yoon. *Truncated Differential Cryptanalysis of Camellia*. ICISC 2001, (to appear in LNCS by Springer-Verlag) (2001).

16. S. Lucks. *Attacking Seven Rounds of Rijndael under 192–bit and 256–bit Keys.* Proceedings of 3rd AES Conference (2000).
17. S. Lucks. *The Saturation Attack – a Bait for Twofish.* Proceedings of Fast Software Encryption (to appear in LNCS by Springer-Verlag) (2001) 1–15.
18. M. Matsui. *New Block Encryption Algorithm MISTY.* Fast Software Encryption, LNCS 1267, Springer-Verlag (1997) 54–68.
19. M. Sugita, K. Kobara and H. Imai, *Security of Reduced Version of the Block Cipher Camellia against Truncated and Impossible Differential Cryptanalysis.* ASIACRYPT 2001, LNCS 2248, Springer-Verlag (2001) 193–207.
20. Y. L. Yin. *A Note on the Block Cipher Camellia.* a contribution for ISO/IEC JTC1 SC27. `http://info.isl.ntt.co.jp/camellia/Publications/yiqun.ps` (2000).