Trends in Mathematics
Information Center for Mathematical Sciences
Volume 8, Number 1, June, 2005, Pages 101–118

INTEGRAL CRYPTANALYSIS AND HIGHER ORDER DIFFERENTIAL ATTACK

YONGJIN YEOM

Abstract. Integral cryptanalysis and higher order differential attack are chosen(or known) plaintext attacks on block ciphers. These attacks have been developed independently and become widely used as strong tools to analyze the security of block ciphers. In this paper, basic idea of these attacks including brief historical comments is described. We give some recent applications of integral cryptanalysis on block ciphers such as Camellia and Safer++. Also, we show that integral cryptanalysis can be interpreted as a special case of higher order differential attack.

1. Introduction

In the last two decades, there was remarkable progress in cryptanalysis of block ciphers. One of the most significant advances was differential cryptanalysis(DC) introduced by Biham and Shamir in 1990. After that, many derivations of DC were developed such as truncated differential cryptanalysis(TDC), impossible differential cryptanalysis(IDC), and higher order differential cryptanalysis. In differential cryptanalysis, one observes the propagation of difference between two values. On the other hand, integral cryptanalysis considers the propagation of sums or differences of many values. Integral cryptanalysis can be applied to the block ciphers which mainly use one-to-one functions. Integral property exploits the relationship among bunches of plaintext–ciphertext pairs in contrast to differential cryptanalysis which uses only one pair.

Integral cryptanalysis was introduced by Knudsen in 2002. However, its basic idea came from Square attack and saturation attack. Square attack was a dedicated attack on the block cipher SQUARE[8] and applied to block ciphers of the SPN structure such as Rijndael[9, 11, 21], CRYPTON[13], and Hierocrypt[3]. In order to apply the Square attack on the Feistel structure, Lucks[22] introduced the saturation attack, as a variation of the Square attack. He analyzed the Twofish algorithm of the modified Feistel structure. At FSE 2002 workshop, Knudsen suggested new name "Integral Cryptanalysis" for this set of techniques. Now, the name "Integral Cryptanalysis" is widely accepted instead of Square attack or saturation attack. Integral cryptanalysis is one of the most powerful attacks on American standard block cipher AES.

In higher order differential attack suggested by Lai, one consider the differential of degree k ($k > 1$). In particular, higher order differential attack can be mounted

to the cipher with components whose algebraic degree are low. Kaneko developed controlled higher order differential attack on block ciphers such as MISTY, KASUMI, and Camellia. This attack is the most powerful attack on Camellia so far.

In this paper, we briefly describe and compare integral cryptanalysis with higher order differential attack. Also, we give some examples of attacks on the block cipher Camellia and Safer++. Lastly, we will see that integral cryptanalysis uses the *highest* order differential and integral cryptanalysis can be interpreted as a special case of higher order differential attack. Recently, these attacks are considered as important tools analyzing the security of block ciphers. We expect that these attacks will be extended and combined with other attacks on block ciphers.

2. Integral Cryptanalysis

We begin by introducing the concept of a multiset and its properties. Let V be a state vector $V = (v_1, \ldots, v_n)$, $v_i \in GF(2^8)$ and S be a multiset of vectors. The following three states of bytes play an important role in integral cryptanalysis. For some fixed index i,

- Fixed: $v_i = c$(constant) for all $V \in S$
- All: $\{v_i : V \in S\} = GF(2^8)$
- Balanced: $\bigoplus_{V \in S} v_i = c'$(constant)

The property 'All' is also called 'Active' or 'Saturated' and the property 'Fixed' is referred as 'Passive' in Square attack.

property 1. *The effects of exclusive-or(XOR) operation on states can be summarized as Table 1.*

Table 1. Properties of XOR operation

$XOR(\oplus)$	*All*	*Fixed*	*Balanced*
All	*Balanced*	*All*	*Balanced*
Fixed	*All*	*Fixed*	*Balanced*
Balanced	*Balanced*	*Balanced*	*Balanced*

property 2. *Let f be a bytewise bijective function. Then f preserves the 'Fixed' and 'All' properties.*

We denote 'Fixed', 'All', and 'Balanced' properties by 'c', 'A', and 'B', respectively.

Using these properties of multisets, we can construct the basic model of integral distinguisher which distinguishes several rounds of a block cipher from a random permutation.

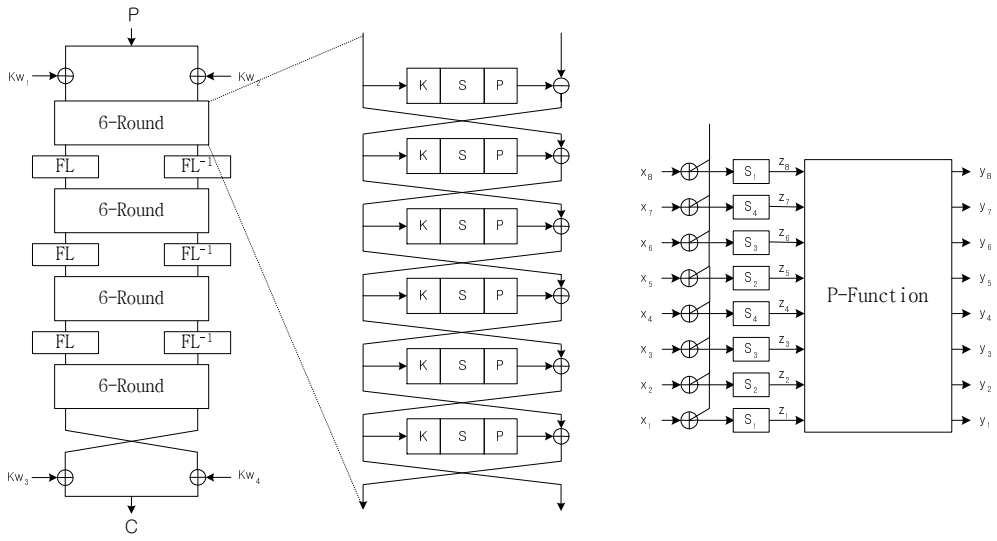
Consider a multiset which consists of 2^8 chosen plaintexts. For example, we may assume that the first byte of the multiset has property 'A' and remaining bytes are fixed. After a few rounds, it is expected that some bytes of output have property 'B' with probability 1. In other words, exclusive-or of 2^8 values in a certain byte

position of outputs is zero. We cannot expect that a random permutation has this property with high probability. Therefore, we can distinguish a few rounds of a block cipher from a random permutation using chosen plaintexts. In the following sections, we give some specific examples in detail.

3. Integral Cryptanalysis on Camellia

3.1. **Description of Camellia.** Camellia[1] is a block cipher jointly designed by NTT and Mitsubishi in 2000. The round function of Camellia is based on that of E2 by NTT. At the end of every 6 rounds, FL/FL^{-1} function layer, which is similar to that of MISTY is inserted. Camellia was submitted to the standardization and the evaluation projects such as ISO/IEC JTC 1/SC 27, CRYPTREC, and NESSIE. According to CRYPTREC Report 2001, Camellia has sufficient security margin and there are no attacks which threat full rounds of Camellia. Currently, Camellia is one of winner algorithms at the NESSIE project.

A 128-bit block cipher Camellia has a modified Feistel structure with 18(for 128 bit key) or 24(for 192, 256 bit key) rounds. 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. Figure 1 shows the encryption procedure of Camellia for 256 bit key.



(a) Encryption Procedure of Camellia (256–bit key) (b) Round function of Camellia

Figure 1. Camellia Algorithm

Let P_L and $P_R(C_L$ and $C_R)$ be the left and the right half of a plaintext P(a ciphertext C), respectively. Let $X^{(i)}$ be the input of the i-th round function F.

Then the input of i-th round can be written as $(X^{(i)}, X^{(i-1)})$. Using 8 byte round key $K^{(i)}$, i-th round performs

$$(X^{(i)}, X^{(i-1)}) \overset{F(\cdot,K^{(i)})}{\longrightarrow} (X^{(i+1)}, X^{(i)}),$$

where

$$X^{(i+1)} = F(X^{(i)}; K^{(i)}) \oplus X^{(i-1)}.$$

By the effect of pre-whitening, the input $(X^{(1)}, X^{(0)})$ of the first round is $(P_L \oplus kw_1, P_R \oplus kw_2)$. At the beginning of every 6 rounds, FL and FL^{-1} functions are applied e.g. $X^{(6)} \leftarrow FL(X^{(6)}; kl_1)$, $X^{(5)} \leftarrow FL(X^{(5)}; kl_2)$. The ciphertext (C_L, C_R) can be obtained by

$$C_L = X^{(r+1)} \oplus kw_3, \quad C_R = X^{(r)} \oplus kw_4,$$

where $r = 18$ for 128 bit key and $r = 24$ for 192, 256 bit key.

As we can see in Figure 1 (b), the round function F has SP(Substitution–Permutation) structure and consists of three parts; adding round key(K), Sbox layer(S), and linear diffusion layer(P). If $X = (x_1, x_2, \ldots, x_8)$ is used as an input for a round function with a round key $K = (k_1, k_2, \ldots, k_8)$, then we can describe $F : (X; K) \mapsto Y$ as follows;

$$\begin{aligned} X' &= X \oplus K = (x_1 \oplus k_1, x_2 \oplus k_2, \ldots, x_8 \oplus k_8) \\ Z &= S(X') = (s_1(x'_1), s_2(x'_2), s_3(x'_3), s_4(x'_4), s_2(x'_5), s_3(x'_6), s_4(x'_7), s_1(x'_8)) \\ Y &= P(Z). \end{aligned}$$

In the S-layer, four different Sboxes s_1, s_2, s_3, and s_4 are used. The diffusion layer P can be written as

$$\begin{aligned} y_1 &= z_1 \oplus z_3 \oplus z_4 \oplus z_6 \oplus z_7 \oplus z_8, \\ y_2 &= z_1 \oplus z_2 \oplus z_4 \oplus z_5 \oplus z_7 \oplus z_8, \\ y_3 &= z_1 \oplus z_2 \oplus z_3 \oplus z_5 \oplus z_6 \oplus z_8, \\ y_4 &= z_2 \oplus z_3 \oplus z_4 \oplus z_5 \oplus z_6 \oplus z_7, \\ y_5 &= z_1 \oplus z_2 \oplus z_6 \oplus z_7 \oplus z_8, \\ y_6 &= z_2 \oplus z_3 \oplus z_5 \oplus z_7 \oplus z_8, \\ y_7 &= z_3 \oplus z_4 \oplus z_5 \oplus z_6 \oplus z_8, \\ y_8 &= z_1 \oplus z_4 \oplus z_5 \oplus z_6 \oplus z_7. \end{aligned}$$

To describe two auxiliary functions FL and FL^{-1} which use two 32 bit keys kl_L and kl_R, we denote bitwise-and, bitwise-or operations by $\cap$, $\cup$ and a n bit left rotation by $\lll_n$, respectively. Let (X_L, X_R) and (Y_L, Y_R) be 64 bit input and output of FL, respectively. Then $FL : (X_L, X_R) \mapsto (Y_L, Y_R)$ is defined by

$$\begin{aligned} Y_R &= ((X_L \cap kl_L) \lll_1) \oplus X_R \\ Y_L &= (Y_R \cup kl_R) \oplus X_L. \end{aligned}$$

Literally, FL^{-1} is the inverse function of FL.

In the key schedule of 256 bit Camellia, two 128 bit intermediate keys K_A and K_B are derived from the encryption key K_L and K_R. We do not describe the detail

procedure here(See [1]). All round keys are bitwise rotations of K_L, K_R, K_A, and K_B.

TABLE 2. Round keys for 256 bit secret key (up to 12 rounds out of 24 rounds)

	subkey	value
Prewhitening	kw_1	$(K_L \lll_0)_{L(64)}$
	kw_2	$(K_L \lll_0)_{R(64)}$
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)	$K^{(10)}$	$(K_L \lll_{45})_{R(64)}$
F (Round 11)	$K^{(11)}$	$(K_A \lll_{45})_{L(64)}$
F (Round 12)	$K^{(12)}$	$(K_A \lll_{45})_{R(64)}$

3.2. **Four Round Distinguisher and Attacks on Camellia.** Choose an input multiset S of 256 vectors (P_L, P_R) as follows:

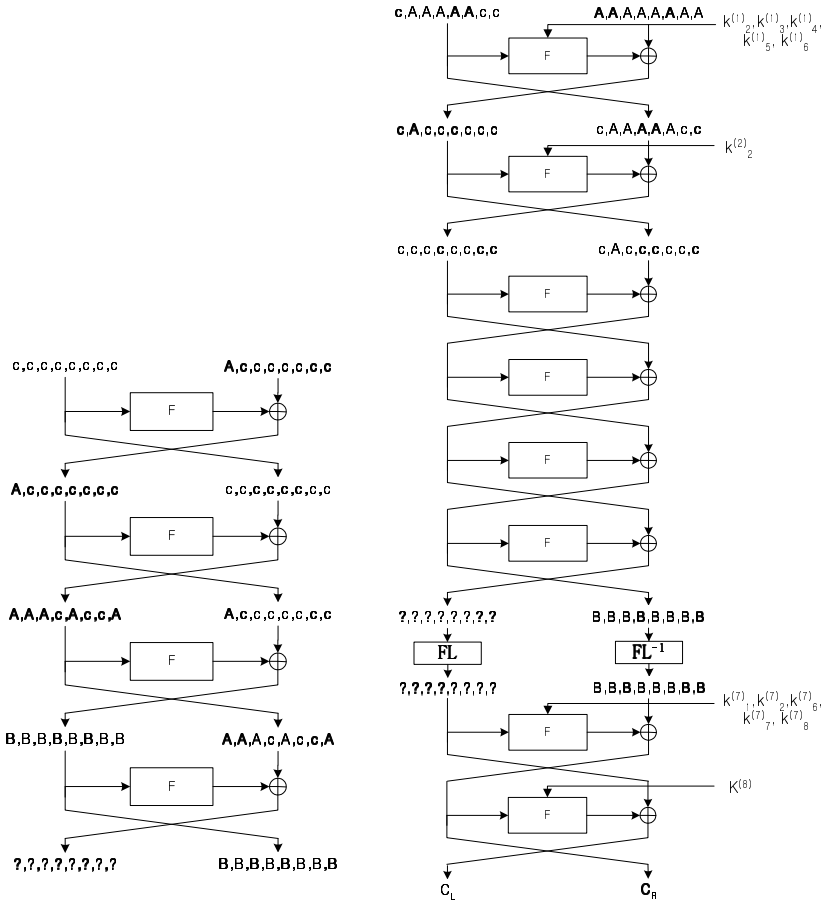
$$P_L = (cccc, cccc), \quad P_R = (Accc, cccc),$$

where A and c mean active and constant properties, respectively. Then all bytes in the right half of the 4th round output are 'Balanced' (See [35] for details). Therefore, we have a distinguisher in Figure 2 (a) which distinguishes 4 rounds of Camellia from a random permutation. Note that we may arbitrarily choose a position for 'All' property. Thus, by changing the position for 'All', we obtain 8 different distinguishers.

From the above distinguisher, we can construct basic attacks on 5 and 6 round Camellia without pre- and post-whitenings. Adding 2 rounds at the beginning of the distinguisher and guessing 6 bytes of subkeys involved in the 1st and the 2nd rounds, we can attack 6 round Camellia. We do not describe these basic attacks here(See [35]).

In [35], by guessing 7 bits of subkey kl_2 we can partially invert of FL^{-1} for more than 7 rounds attacks. However, such an inverting turns out to be redundant. In fact, if we assume 6 byte subkeys of the first two rounds correctly, every byte of the right half of the 6th round outputs is 'Balanced'. Since FL^{-1} function is linear,

'Balanced' property is preserved. Thus, the right half of input for the 7th round is 'Balanced'.



(a) Four round distinguisher (b) Attack on 8 rounds of Camellia

FIGURE 2. Integral Cryptanalysis on Camellia

From this observation, guessing additional 5 byte subkeys of the 7th round we can construct an attack algorithm for 7 rounds without assuming subkeys of FL^{-1} function.

Table 2 shows that the seventh and the eighth round keys $K^{(7)}$ and $K^{(8)}$ are nothing but 30 bit rotations of the first and the second round keys, respectively. Up to 8 rounds, we only need to guess subkeys derived from the intermediate key K_B.

We can optimize the subkey guessing by choosing the input (X_L, X_R) of the distinguisher as follows:

$$X_L = (cccc, cccc), \quad X_R = (cAcc, cccc).$$

The number of subkey bits which we need to guess for the attack on 7 rounds is

$$40(1\text{R}) + 8(2\text{R}) + 3(7\text{R}) = 51\text{bits}.$$

To attack 8 rounds, we need to guess additional 7 byte subkeys.

The algorithm to attack 8 round Camellia can be summarized as follows:

(1) Guess 48 bit subkeys $k_2^{(1)}$, $k_3^{(1)}$, $k_4^{(1)}$, $k_5^{(1)}$, $k_6^{(1)}$ and $k_2^{(2)}$ of the first and the second rounds.
(2) Prepare 8 multisets so that inputs (X_L, X_R) of the third round are of the form
$$X_L = (cccc, cccc), \quad X_R = (cAcc, cccc).$$
Note that we expect the right half of the 7th round inputs is 'Balanced' if key guessing is correct.
(3) For each key guessing, do the first part of attack.
   (a) Guess additional 59 bits of 7 and 8 round subkeys.
   (b) Partially decrypt ciphertexts and check the 'Balanced' property on the 5th byte of the 7th round inputs for all 8 multisets.
   (c) If balanced for all multisets, store subkeys in the table. Otherwise, discard them.
   (d) Repeat step (a) $\sim$ (c) until all possible 59 bit subkeys are tested.
(4) For each key in the table do the second part of attack.
   (a) Using 48 bit subkeys of 1, 2 rounds, prepare 6 multisets.
   (b) Using the remaining subkeys decrypt 7, 8 rounds and check balance.
   (c) If balanced for all multisets, accept 107 bit subkeys as a correct key.

During the first part of attack we need $2^{48} \times 2^8 \times 8$ chosen plaintexts. With probability 2^{-64} one of subkeys can pass the balance test. Thus, on the average, $2^{107}/2^{64} = 2^{43}$ keys are stored in the table. Thus, it is sufficient to prepare 6 multisets in the second part of attack. The total number of plaintexts for 8 round attack is

$$2^{48} \times 2^8 \times 8 + 2^{43} \times 2^8 \times 8 = 2^{59.03}.$$

The time complexity of the algorithm can be calculated as follows:

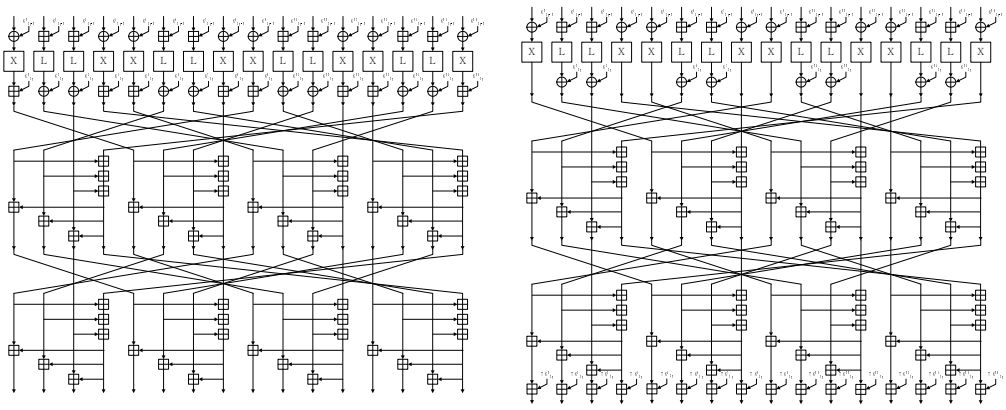
$$\begin{aligned} 2^{48} \times 2^8 \times 8 \text{ (encryption for 1st part)} &+ \\ 2^{48} \times 2^8 \times 8 \times 2^{59} \times 2/8 \text{ (decryption for 1st part)} &+ \\ 2^{43} \times 2^8 \times 6 \text{ (encryption for 2nd part)} &+ \\ 2^{43} \times 2^8 \times 6 \times 2/8 \text{(decryption for 1st part)} & \\ = \quad 2^{116} \text{ (encryptions)}. & \end{aligned}$$

By assuming all subkeys in round 9 and 10, we construct attack algorithms up to 10 rounds.

4. Integral Cryptanalysis on Safer++

4.1. **Description of Safer++.** Safer++[7], the latest version of the Safer family, is the upgrade of its predecessors, Safer and Safer+. In the year of 2000, this block cipher was proposed as one of the candidates for the NESSIE project. After two years of analysis and evaluation, it was placed on the list of candidates for the second phase even though Safer++ does not appear on the winner list of the NESSIE project.

Safer++$_{128}$ is a 128 bit block cipher with user-selected-key length 128 and 256 bits. For simplicity we denote Safer++$_{128}$ with 128 bit key by Safer++.



(a) Round function (b) Equivalent form of round function

Figure 3. Round function of Safer++ and its equivalent form

The round function of Safer++ consists of 4 parts as follows:

- Applying round key K_{2j-1}
- Nonlinear S-box layer
- Applying round key K_{2j}
- Linear diffusion layer

We count each part as 0.25 round as used in the NESSIE report. For 128 and 256 bit key lengths, Safer++ uses 7.25 rounds and 10.25 rounds, respectively.

For each round, two 128 bit subkeys are used. Round key K_{2j-1} and K_{2j} are applied for round j. At the end of the final round r, post whitening subkey K_{2r+1} are added.

There are two types of S-boxes X and L in the nonlinear layer. We only use the fact that both are one to one functions. When applying round keys, Safer++ uses two kinds of operations addition mod 256($\boxplus$) and XOR($\oplus$). The linear diffusion

layer based on the PHT maps a to b by $b = a \cdot M_\theta$, where

$$M_\theta = \begin{pmatrix}
1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 4 & 2 & 2 & 2 & 1 & 1 & 2 & 1 \\
2 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 2 & 4 & 2 & 2 \\
2 & 2 & 4 & 2 & 2 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 2 & 1 & 2 & 1 & 1 & 1 \\
4 & 2 & 2 & 2 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 \\
1 & 1 & 2 & 1 & 2 & 1 & 1 & 1 & 2 & 4 & 2 & 2 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 2 & 2 & 4 & 2 & 2 & 1 & 1 & 1 \\
1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 2 & 1 \\
1 & 1 & 2 & 1 & 4 & 2 & 2 & 2 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 2 & 4 & 2 & 2 & 1 & 1 & 2 & 1 & 2 & 1 & 1 & 1 \\
1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 2 & 2 & 4 & 2 \\
2 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 2 & 1 & 4 & 2 & 2 & 2 \\
2 & 4 & 2 & 2 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 2 & 1 \\
2 & 1 & 1 & 1 & 2 & 2 & 4 & 2 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 \\
1 & 1 & 2 & 1 & 2 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1
\end{pmatrix}.$$

The key schedule of Safer++ is very simple. From 128 bit user key $K^1, \ldots, K^{16}$, the parity byte K^0 is calculated by

$$K^0 = \bigoplus_{i=1}^{16} K^i.$$

The p-th byte $K_j^p$of the j-th round key K_j can be computed from the master key by

$$K_j^p \leftarrow K^{j+p-1 \mod 17}. \tag{1}$$

It follows from (1) that K_j^p and K_i^q are derived from the same master key byte, if $j + p = i + q$. Furthermore, one can be computed from the other. From the simplicity and absence of nonlinearity in the key schedule, once some part of round key is exposed, some other key elements can be obtained.

4.2. **Two Round Integral Distinguisher.** Piret[27] found two round distinguishers which can be used to distinguish 2 rounds of Safer++ from a random permutation. Note that balance property means the sum of all entries is zero. i.e.,

$$\boxplus_{i=1}^{k \cdot 2^n} x_i = 0.$$

Consider the following multiset with 2^{16} plaintexts

$$\{C_0, C_1, a_i, C_3;\ C_4, C_5, C_6, C_7;\ C_8, C_9, C_{10}, C_{11};\ b_i, C_{13}, C_{14}, C_{15}\}, \tag{2}$$

where C_j's are constants and a_i, b_i are active.

Provided that all C_j are zero, the output of the first round is of the form

$$\begin{aligned}
&\{2a_i + b_i, 2a_i + b_i, 4a_i + b_i, 2a_i + b_i;\ 2a_i + b_i, a_i + 2b_i, a_i + b_i, a_i + b_i; \\
&a_i + b_i, 2a_i + b_i, a_i + 2b_i, a_i + b_i;\ a_i + 4b_i, a_i + 2b_i, a_i + 2b_i, a_i + 2b_i\}.
\end{aligned} \tag{3}$$

It is easy to check that all entries in this multiset are active. Before the diffusion layer in the second round all bytes remain active. Thus, every byte of the second round output is balanced. The choice of the active bytes in (2) is not unique. For example, we may choose 0th and 11th bytes as active ones.

If we choose the positions of two active bytes a_i and b_i arbitrarily, some bytes of the first round output may not be active. In that case, each byte of the first round output is one of the followings:

A : $a_i + b_i$, $2a_i + b_i$, $a_i + 2b_i$, $4a_i + b_i$, $a_i + 4b_i$
B : $2a_i + 2b_i$, $4a_i + 2b_i$, $2a_i + 4b_i$

The byte which belongs to 'A' is active and the bytes in 'B' are balanced. It is very interesting fact that the balanced property in 'B' is preserved by one-to-one S-boxes. For example, consider $2a_i + 2b_i = 2(a_i + b_i)$. All values which can be expressed of the form $2(a_i + b_i)$ appear multiple of 2^8 times. For that reason, the output of S-box layer can be balanced, too.

Thus, if we choose a set of plaintexts in which arbitrary two bytes are active and others are fixed, then the output of the second round is balanced. Note that it is satisfied with probability 2^{-128} for a random permutation.

By adding rounds at the beginning and the end of distinguishers, we can construct attack algorithms on reduced-round Safer++. The method adding a round at the beginning is denoted by '-1R' attack. Similarly, '+1R' attack means an attack by adding a rounds at the end. Such notations are introduced by Hatano et al.[12].

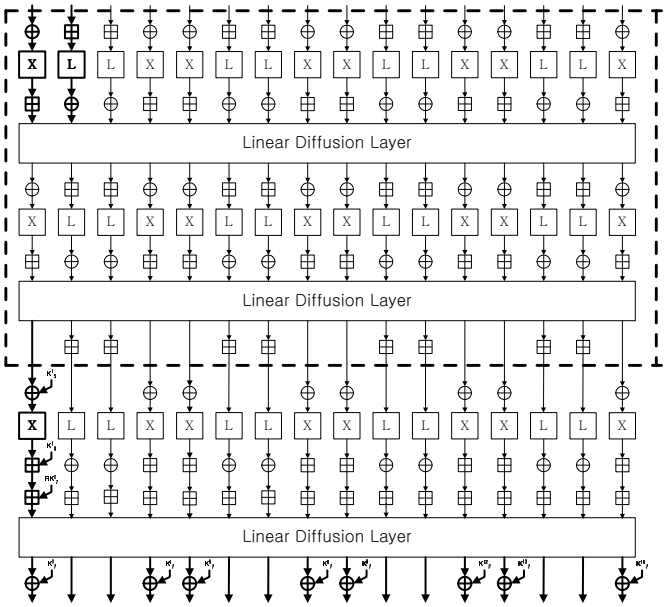


FIGURE 4. Attack on 3.25 rounds of Safer++ (+1R attack)

4.3. **'+1R' Attack on 3.25 rounds of Safer++.** To attack 3.25 rounds of Safer++ by adding a round at the end, we use the equivalent form of the 3rd round so as to reduce the number of key guesses. We should assume the following subkeys

to decipher the last round.

$$K_7^1, K_7^4, K_7^5, K_7^8, K_7^9, K_7^{12}, K_7^{13}, K_7^{16}, \quad K_6^1, K_5^1, \quad RK_7^1, \tag{4}$$

where RK_7^1 is the first byte of the equivalent round key for 3rd round. We can compute K_5^1 from K_7^{16} by using the key schedule. Also, we assume one byte $RK = RK_7^1 \boxplus K_6^1$ instead of guessing RK_7^1 as well as K_6^1. In short, the number of bytes we should assume can be reduced to nine.

(1) Prepare 9 multisets with 2^{16} entries whose two bytes are active and other bytes are passive.
(2) guess 9 bytes to determine subkeys in (4).
(3) For each multiset, decipher the third round and check the first byte is balanced.
(4) If balanced for all multisets, print subkeys and stop. Otherwise, guess another subkey and repeat 2–4.

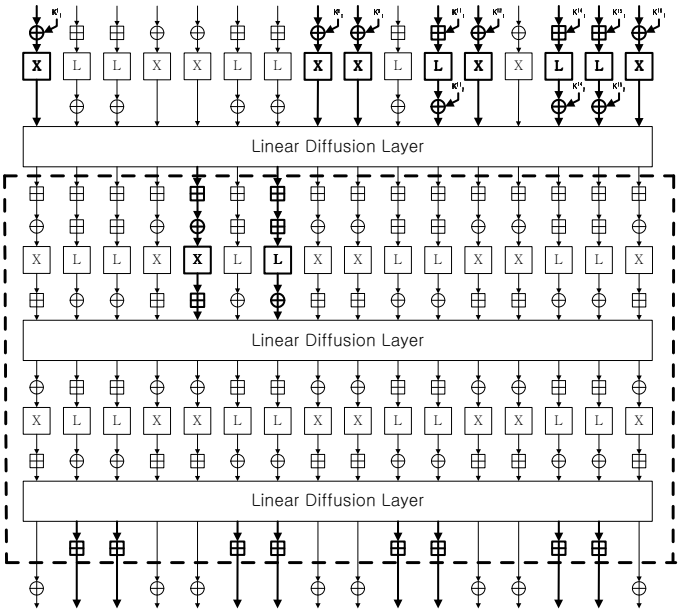


FIGURE 5. Attack on 3.25 rounds of Safer++ (-1R attack)

4.4. **'-1R' Attack on 3.25 rounds of Safer++.** By adding a round at the beginning, we have another attack on 3.25 round of Safer++. In this case, the number of byte we should assume depends upon the positions of active bytes. We may choose $\binom{16}{2} = 120$ different positions of active bytes for 2 round distinguishers. We can minimize the number of subkey guesses by choosing the 5th and the 7th bytes as active bytes. This is the unique case we guess only 8 bytes for -1R attack.

(1) Guess 8 byte subkeys as we can see in Figure 6.

(2) Choose a multiset as plaintexts so that the 5th and the 7th bytes of the second round input can be active if key guessing is correct.
(3) Encipher this multiset.
(4) Check the balanced property at every byte of $4n+2$ or $4n+3$ positions.

Since a wrong key can pass the balanced test with probability 2^{-64}, one multiset is sufficient to mount '-1R' attack. We need $2^{64} \times 2^{16}$ chosen plaintexts and time for $2^{64} \times 2^{16} \times (1 + 1/3)$ encryptions.

4.5. **'+1R-1R' Attack on 4.25 rounds of Safer++.** Combining '+1R' attack and '-1R' attack, we have '+1R-1R' attack on 4.25 rounds of Safer++. We need to guess subkeys involved in the first and the fourth rounds. As mentioned previously, there are 120 different methods selecting positions of active bytes for 2 round distinguisher. For each choice of active bytes, we can choose the byte of the third round output which is used to check balanced property. So, there are 120×16 different methods attacking 4.25 rounds by '+1R-1R' attack. Among them, we finally choose a method which minimizes the number of subkey guesses. More precisely, we choose the 13th and 14th bytes for active and check the balanced property at the 14th byte of the third round output. We need to guess 11 bytes for this attack.

To construct $2^{72} \times 2^{16} \times 11$ chosen plaintexts we need $2^{72} \times 2^{16} \times 11 \times 1/4$ decryptions. To encipher and to check the balanced, we need the time complexity corresponding to $2^{72} \times 2^{16} \times 11 \times (1 + 1/4)$ encryptions.

5. Higher Order Differential

Higher order differential attack introduced by Lai and Knudsen is a strong attack on block ciphers with components of low algebraic degree such as KN Cipher and CAST. Kaneko developed controlled higher order differential attacks on MISTY, E2, Camellia, etc.

We begin by defining higher order differentials. Let f be a function which transforms an input $x \in GF(2)^n$ to the output $y \in GF(2)^n$ under a key $k \in GF(2)^s$. Then the first order differential for the input difference a at x is defined by

$$\Delta^{(1)}_{(a)} f[k](x) = f[x](x) \oplus f[x](x \oplus a).$$

The i-th order differential is defined inductively by

$$\Delta^{(i)}_{(a_i,\ldots,a_1)} f[k](x) = \Delta^{(1)}_{(a_i)} \left(\Delta^{(i-1)}_{(a_{i-1},\ldots,a_1)} f[k](x) \right).$$

Let $V^{(m)}[a_1, \ldots, a_m]$ be a subspace of $GF(2)^n$ generated by elements $a_1, \ldots, a_m \in GF(2)^n$. Then we can rewrite the i-th order differential as a exclusive-or sum of the form

$$\Delta^{(i)}_{(a_i,\ldots,a_1)} f[k](x) = \bigoplus_{a \in V^{(i)}[a_i,\ldots,a_1]} f[k](x + a).$$

Consider a function $f : GF(2)^n \to GF(2)^n$ under a key $k \in GF(2)^s$. If $a_1, a_2, \ldots, a_i$ are linearly independent in $GF(2)^n$ and span a subspace $V^{(i)}$, we write for simplicity

$$\Delta^{(i)} f[k](x) = \bigoplus_{a \in V^{(i)}} f[k](x \oplus a).$$

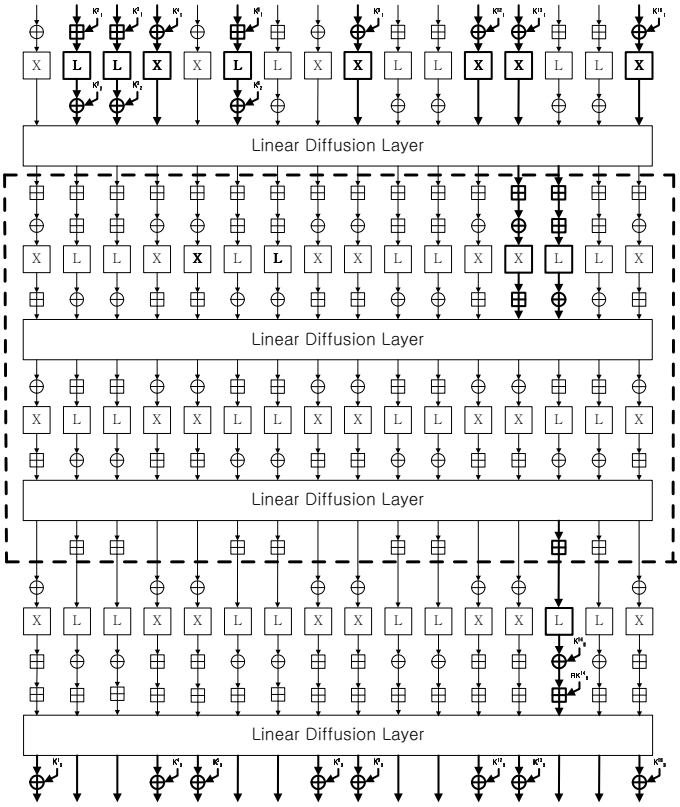


FIGURE 6. Attack on 4.25 rounds of Safer++ (+1R-1R attack)

If the algebraic degree of f with respect to x equals N, $\Delta^{(N+1)}f[k](x) = 0$ and $\Delta^{(N)}f[k](x)$ is a constant. Note that if $f : GF(2)^n \to GF(2)^n$ is a one-to-one function of x, $\Delta^{(n)}f[k](x) = 0$.

6. Higher Order Differential Attack on Camellia

6.1. **Higher Order Differential Property of Camellia.** The designers of Camellia claimed that 6 or more rounds of Camellia is secure against higher order differential attack because the algebraic degree of S-box is 7. However, Kaneko[15] proposed the 8th order differential attack on 10 rounds of Camellia by controlling plaintext technique.

Recall that we denote round input, round output, and input of P function in the round function by $X^{(r)}$, $Y^{(r)}$, and $Z^{(r)}$, respectively.

Let (C_L, C_R) be ciphertext of 6 round Camellia. Then we have

$$\begin{aligned} Z^{(6)} &= S(C_R \oplus K^{(6)}) = P^{-1}(C_L \oplus X^{(5)}) \\ &= P^{-1}(C_L) \oplus P^{-1}(X^{(3)}) \oplus Z^{(4)} \end{aligned}$$

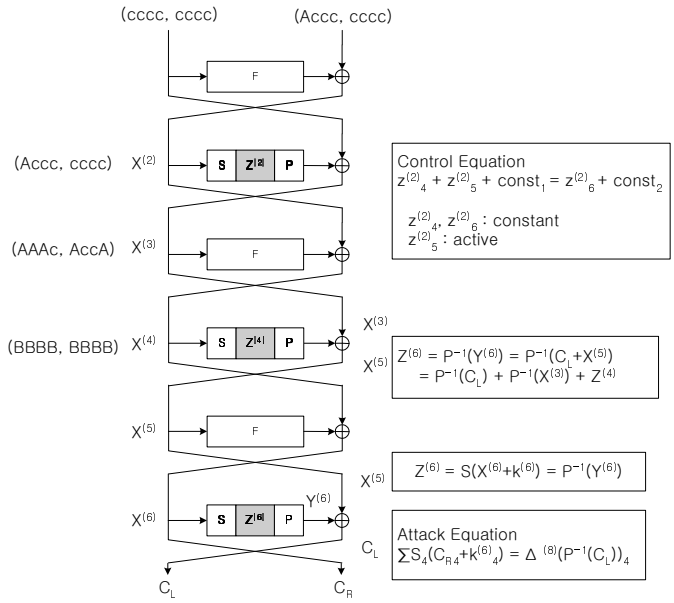


FIGURE 7. Controlled higher order differential attack on 6 rounds of Camellia

Considering the 8-th order differential on the i-th bytes, we have

$$\begin{aligned}\Delta^{(8)} z_i^{(6)} &= \bigoplus_{V^{(8)}} S_i(C_{Ri} \oplus k_i^{(6)}) \\ &= \Delta^{(8)} \left(P^{-1}(C_L)\right)_i \oplus \underbrace{\Delta^{(8)} \left(P^{-1}(X^{(3)})\right)_i}_{A_{control}} \oplus \underbrace{\Delta^{(8)} z_i^{(4)}}_{B_{control}}.\end{aligned}$$

Controlling chosen plaintext. If $A_{control}$ and $B_{control}$ are controlled to be zero, we have the following attack equation so that we find one byte round key $k_i^{(6)}$.

$$\bigoplus_{V^{(8)}} S_i(C_{Ri} \oplus k_i^{(6)}) = \Delta^{(8)} \left(P^{-1}(C_L)\right)_i . \tag{5}$$

Choose 256 chosen plaintexts (P_L, P_R) of the form

$$P_L = (cccc, cccc), \quad P_R = (Accc, cccc).$$

Recall that A and c implies active and constant properties, respectively. Then it follows from $X^{(2)} = (Accc, cccc)$ and $X^{(3)} = (AAAc, AccA)$ that

$$B_{control} = \Delta^{(8)} \left(P^{-1}(X^{(3)})\right)_i = 0.$$

Controlling procedure for the condition $A_{control} = 0$ is as follows:

(1) It is sufficient for satisfying $A_{control} = \Delta^{(8)} z_i^{(4)} = 0$ that $z_i^{(4)}$ has active property.
(2) Since $z_i^{(4)} = S(x_i^{(4)})$, $x_i^{(4)}$ should be active.

(3) For each byte index i, $x_i^{(4)} = y_i^{(3)} \oplus x_i^{(2)}$ and

$$x_i^{(2)} = \begin{cases} \text{active}, & i = 1, \\ \text{constant}, & i = 2, 3, \ldots, 8. \end{cases}$$

(4) For $i = 4$, since $x_4^{(2)} = (\mathrm{c})$, $y_4^{(3)}$ is expectecd to be active.

(5) From

$$\begin{aligned} y_4^{(3)} &= z_2^{(3)} \oplus z_3^{(3)} \oplus z_4^{(3)} \oplus z_5^{(3)} \oplus z_6^{(3)} \oplus z_7^{(3)} = Z^{(3)}[2,3,4,5,6,7] \\ &= s_2(y_2^{(2)} \oplus const_1) \oplus s_3(y_3^{(2)} \oplus const_2) \oplus s_2(y_5^{(2)} \oplus const_3) \oplus const_4 \end{aligned}$$

the condition $y_2^{(2)} \oplus const_1 = y_5^{(2)} \oplus const_3$ implies that $y_4^{(3)}$ is active, where $Z^{(3)}[2,3,4,5,6,7] = z_2^{(3)} \oplus z_3^{(3)} \oplus z_4^{(3)} \oplus z_5^{(3)} \oplus z_6^{(3)} \oplus z_7^{(3)}$.

(6) Rewriting this, to satisfiy

$$Z^{(2)}[1,2,4,5,7,8] \oplus const_1 = Z^{(2)}[1,2,6,7,8] \oplus const_2,$$

$z_4^{(2)} \oplus z_5^{(2)} \oplus const_1 = z_6^{(2)} \oplus const_2$ is required.

(7) Considering one-to-one correspondence between $z_i^{(2)}$ and P_{Ri}, above equality is satisfied at least once, if we fix P_{R4} and P_{R6} arbitrarily and take all possible values of P_{R5}.

Therefore, if we guess round key $k_4^{(6)}$ and choose plaintext P_{R5} correctly, the following attack equation holds.

$$\bigoplus_{V^{(8)}} S_4(C_{R4} \oplus k_4^{(6)}) = \Delta^{(8)} \left(P^{-1}(C_L)\right)_4 .$$

6.2. **Basic Attack on 6 Rounds of Camellia.** We describe the attack algorithm for 6 rounds of Camellia(Figure 7).

(1) Prepare 256 chosen plaintexts (P_L, P_R) which have the following property.

$$P_L = (cccc, cccc), P_R = (Accc, cccc).$$

(2) Guess a value of controlled variable $_{R5}$.

(3) By exhaustive search, choose $k_4^{(6)}$ which satisfies

$$\bigoplus_{V^{(8)}} S_4(C_{R4} \oplus k_4^{(6)}) = \Delta^{(8)} \left(P^{-1}(C_L)\right)_4 .$$

(4) For all possible value of P_{R5}, repeat step 2 and 3.

(5) For each P_{R5} we expect that one or two candidates for $k_4^{(6)}$ are found.

(6) Choose new value (P_{R2}, P_{R7}, P_{R8}) which is not concerned with the attack equation repeat all steps until only one key candidate remains.

6.3. **Attacks on More Rounds of Camellia.** We can apply the technique, which adds rounds at the beginning of the distinguisher in integral cryptanalysis, to higher order differential attack. Adding a round at the beginning in Figure 7 we have an attack algorithm on 7 rounds of Camellia. Guessing round keys of 7th to 10th round, we obtain a chosen plaintext attack on 10 rounds of Camellia (see [34] for details). On the other direction, using the 16th order differential, Hatano et al.[12] proposed a 11 rounds higher order differential attack which is a kind of chosen ciphertext attack.

6.4. **Comparison of Integral Cryptanalysis and Higher Order Differential Attack.** Recall the definition of higher order differential

$$\Delta^{(i)}_{(a_i,\ldots,a_1)} f[k](x) = \bigoplus_{a \in V^{(i)}[a_i,\ldots,a_1]} f[k](x+a).$$

At the integral cryptanalysis on Camellia we use input for 4 round distinguisher of the form

$$(cccccccc, Acccccc).$$

In this case we regard the distinguisher as a function of the 1st byte x of right half under the fixed key k. Define $f : GF(2)^8 \to GF(2)^{8\cdot 8}$ which maps the first byte of right half to the right half of the distinguisher output. Note that if we use 256 chosen plaintexts as input for the distinguisher we find balance property on the right half of its output. Precisely, for input of the form

$$\left\{(c_0c_1c_2c_3c_5c_5c_6c_7c_8, xc_{10}c_{11}c_{12}c_{13}c_{14}c_{15}c_{16}) | x \in GF(2)^8, c_i \text{ is fixed for all } i\right\}$$

we expect that exclusive-or of each byte of right half of 4 round output equals zero. Rewriting this property usung the function f which we have just defined,

$$\bigoplus_{x \in GF(2)^8} f[k](x) = 0 \text{ (balance)}.$$

This is nothing but the 8th order differential of f. In fact,

$$\Delta^{(8)} f[k](x) = \bigoplus_{a \in GF(2)^8} f[k](x+a) = \bigoplus_{x \in GF(2)^8} f[k](x).$$

Integral cryptanalysis which uses the highest order differential can be interpreted as a special case of higher order differential attack. In the cryptanalysis of block ciphers, it is quite interesting that the highest order *differential* attack can be called *integral* cryptanalysis.

7. Conclusion

In this paper we present the basic idea and applications of integral cryptanalysis and higher order differential attack. As we can see in the previous section, these attacks can be interpreted as higher order differential attacks in a unified manner. In fact, these attacks have been developed affecting each other by exchanging key ideas. In many block ciphers, these attacks become popular tools for analyzing the security of ciphers.

We summarize the results of attacks discussed in this paper and compare them with those of other attacks in Table 3.

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).

TABLE 3. Summary of attacks on Camellia for 256 bit key and on Safer++ for 128 bit key

Algorithm	Attack	Round	Plaintexts	Time	By	References
Camellia	Square	6	$2^{11.7}$	2^{112}	He &Qing	ICICS 2001[14]
Camellia	TDC	8	$2^{83.6}$	$2^{55.6}$	Lee et al.	ICISC 2001[20]
Camellia	Square	9	$2^{60.5}$	$2^{202.2}$	Yeom, Park, Kim	FSE 2002[35]
Camellia	Higher Order	10	2^{21}	$2^{254.7}$	Kawabata, Kaneko	NESSIE Worksop[15]
Camellia	Integral	10	$2^{60.6}$	$2^{246.3}$	Yeom et al.	SCIS 2003[33]
Camellia	Higher Order	10	$2^{20.6}$	$2^{209.6}$	Yeom et al.	WISC 2002[34]
Camellia	Higher Order	11	2^{93}	$2^{255.6}$	Hatano et al.	SAC 2003[12]
Safer++	LC	3.25	2^{81}	2^{103}	Nakahara et al.	FSE 2001[24]
Safer++	Integral	3.75	2^{64}	2^{108}	Piret, Quisquater	NESSIE Report[27]
Safer++	Integral	4.25	2^{92}	2^{92}	Yeom, Park	CSS 2003[32]
Safer++	Boomerang	5.75	2^{108}	2^{108}	Biryukov et al.	CRYPTO 2003[5]

[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] A. Biryukov, C. De Canniére, G. Dellkrantz, *Cryptanalysis of Safer++*, IACR ePrint Archive 2003/109, (to appear at CRYPTO 2003), (2003).

[6] Camellia Home Page. `http://info.isl.ntt.co.jp/camellia/.`

[7] Cylink Corporation, *Nomination of Safer++ as Candidate Algorithm for the New European Schemes for Signatures, Integrity, and Encryption(NESSIE)*, NESSIE submission, (2000).

[8] J. Daemen, L. R. Knudsen and V. Rijmen. *The Block Cipher SQUARE*. Fast Software Encryption, LNCS 1267, Springer-Verlag (1997) 149–165.

[9] J. Daemen and V. Rijmen. *AES Proposal: Rijndael (Document version 2)*. AES Submission (1999).

[10] O. Dunkelman, *Safety Margins for NESSIE submissions – Safer++ and Hierocrypt (L1/3)*, NESSIE Public Report, NES/DOC/TEC/WP3/015/a, (2002).

[11] 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.

[12] Y. Hatano, H. Sekine, T. Kaneko, *Higher Order Differential Attack of Camellia(II)*, Proceedings of Selected Areas in Cryptography (2002).

[13] 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.

[14] Y. He and S. Qing, *Square Attack on Reduced Camellia Cipher*. ICISC 2001, LNCS 2229, Springer-Verlag (2001) 238–245.

[15] T. Kawabata and T. Kaneko. *A Study on Higher Order Differential Attack of Camellia*. the 2nd open NESSIE workshop (2001).

[16] 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.

[17] 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).

[18] L. R. Knudsen. *Analysis of Camellia.* a contribution for ISO/IEC JTC1 SC27. `http://info.isl.ntt.co.jp/camellia/Publications/knudsen.ps` (2000).
[19] L. Knudsen, *Integral Cryptanalysis*, Proceedings of Fast Software Encryption (2002).
[20] 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).
[21] S. Lucks. *Attacking Seven Rounds of Rijndael under 192–bit and 256–bit Keys.* Proceedings of 3rd AES Conference (2000).
[22] S. Lucks. *The Saturation Attack – a Bait for Twofish.* Proceedings of Fast Software Encryption (to appear in LNCS by Springer-Verlag) (2001) 1–15.
[23] M. Matsui. *New Block Encryption Algorithm MISTY.* Fast Software Encryption, LNCS 1267, Springer-Verlag (1997) 54–68.
[24] J. Nakahara Jr, B. Preneel, J. Vandewalle, Linear Cryptanalysis of reduced-round versions of the Safer block cipher family, Proceedings of Fast Software Encryption (2001).
[25] J. Nakahara Jr, B. Preneel, J. Vandewalle, Linear Cryptanalysis of Reduced-Round SAFER++, Second Open NESSIE Workshop (2001).
[26] J. Nakahara Jr, B. Preneel, *Impossible Differential Attacks on Reduced-Round Safer Ciphers*, NESSIE Public Report, NES/DOC/KUL/WP5/30/1, (2003).
[27] G. Piret, J. Quisquater, *Integral Cryptanalysis on reduced-round Safer++*, NESSIE Public Report, NES/DOC/UCL/WP5/002/1. (2003).
[28] B. Preneel, et. al., *NESSIE security report version 1.0*, NESSIE Public Report, NES/SOC/ENS/WP5/D20/1, (2002).
[29] B. Preneel, et. al., *NESSIE security report version 2.0*, NESSIE Public Report, NES/SOC/ENS/WP5/D20/1, (2003).
[30] 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.
[31] 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).
[32] Y. Yeom, I. Park, *Optimization of Integral Cryptanalysis on Reduced-Round Safer++*, Proceedings of Computer Security Symposium 2003 (2003).
[33] Y. Yeom, I. Park, I. Kim, *A Study of Integral Type Cryptanalysis on Camellia*, Proceedings of the 2003 Symposium on Cryptography and Information Security (2003).
[34] Y. Yeom, S. Park, H. Kim, I. Park, *Improved Higher Order Differential Attack and Square Attack on Camellia*, Proceedings of WISC 2002 (2002).
[35] Y. Yeom, S. Park, I. Kim, "On the Security of CAMELLIA against the Square Attack", FSE 2002 LNCS 2365, Springer-Verlag (2002) 89–99.

National Security Research Institute, 161 Gajeong-dong, Yuseong-gu, Daejeon, 305-350 Korea

E-mail address: `yjyeom@etri.re.kr`