The 128-bit Blockcipher CLEFIA

Security and Performance Evaluations

Revision 1.0

June 1, 2007

Sony Corporation

NOTICE

Acknowledgments

Tetsu Iwata of Nagoya University has contributed to the development of CLEFIA.

Contact

E-MAIL :
clefia-q@jp.sony.com

Postal Address :
Information Technologies Laboratories
Sony Corporation
1-7-1 Konan, Minato-ku, Tokyo 108-0075 Japan

Revision History

June 1, 2007	1.0 revision

Contents

1 Introduction

This document describes evaluation results of CLEFIA. The contents of this document is divided into two parts depending on the aspects of evaluations; the first aspect is on the security, and the second is on the performance of CLEFIA.

The first half of this document mentions security of CLEFIA. To estimate security of CLEFIA, all known attacks of blockciphers are considered. After checking the applicability of each attack, immunity of CLEFIA against each attack is evaluated in detail by estimating how many rounds of CLEFIA can be attacked. There are twenty types of attacks considered in total for CLEFIA. All names of such attacks are listed as:

1. Differential Cryptanalysis
2. Linear Cryptanalysis
3. Differential-Linear Cryptanalysis
4. Boomerang Attack
5. Amplified Boomerang Attack
6. Rectangle Attack
7. Truncated Differential Cryptanalysis
8. Truncated Linear Cryptanalysis
9. Impossible Differential Cryptanalysis
10. Saturation Cryptanalysis
11. Collision Attack
12. Higher Order Differential Cryptanalysis
13. Interpolation Cryptanalysis
14. XSL Attack
15. χ^2 Cryptanalysis
16. Slide Attack
17. Related-Cipher Cryptanalysis
18. Related-Key Cryptanalysis
19. Related-Key Boomerang Cryptanalysis

20. Related-Key Rectangle Cryptanalysis

These evaluated results are shown in this order in the following sections.

The latter half of this document is for the performance results of CLEFIA. CLEFIA was designed to achieve high efficiency in both software and hardware implementations. In software, CLEFIA with 128-bit key achieves about 13 cycles/byte, 1.48 Gbps on 2.4 GHz AMD Athlon 64. A hardware implementation of 128-bit key CLEFIA is very small, occupying only 5,979 gates, with high efficiency of 268.63 Kbps/gate by 0.09 μm CMOS ASIC library. This is in the smallest class as well as the most efficient class among all known 128-bit blockciphers.

This document is organized as follows: in Section 2, security with regard to the data processing part of CLEFIA is described. Then in Section 3, security with regard to the key scheduling part of CLEFIA is described. Detailed results of software and hardware implementations are shown in Section 4 and 5, respectively.

2 Cryptanalysis I — Data Processing Part

In this section, the following attacks are considered to evaluate the security of the data processing part of CLEFIA.

1. Differential Cryptanalysis
2. Linear Cryptanalysis
3. Differential-Linear Cryptanalysis
4. Boomerang Attack
5. Amplified Boomerang Attack
6. Rectangle Attack
7. Truncated Differential Cryptanalysis
8. Truncated Linear Cryptanalysis
9. Impossible Differential Cryptanalysis
10. Saturation Cryptanalysis
11. Collision Attack
12. Higher Order Differential Cryptanalysis
13. Interpolation Cryptanalysis
14. XSL Attack
15. χ^2 Cryptanalysis

2.1 Differential Cryptanalysis

Differential cryptanalysis is a general technique for the analysis of blockciphers, which was proposed by Biham and Shamir [5,6]. There are two ways to evaluate the immunity of blockciphers against differential attack,

1. To show there is no differential which can be used to distinguish from random permutations
2. To show there is no differential characteristic which can be used to distinguish from random permutations

So far, it is believed to be difficult to achieve the first goal for many ciphers. Although there is a useful theory proposed by Hong *et al.* to evaluate maximal differential probability of SPN structure, we cannot use the theory for the evaluations of CLEFIA as AES and FOX because CLEFIA doesn't use SPN type structure [12, 15, 17].

We adopt the remaining approach to estimate probabilities of differential characteristics. It is known that this can be achieved by counting guaranteed numbers of active S-boxes. This approach of counting active S-boxes is adopted by AES, Camellia and other well-known blockciphers as well [2, 12]. The gap between the maximal differential probability and the maximum differential characteristic probability was not very clear so far, but the relationship between them was discussed in detail by Daemen and Rijmen [11]. From their result, there is a certain statistical relationship between them if we accept some statistical assumptions. Thus the characteristics based approach can be considered as a reasonable way to estimate the immunity against differential cryptanalysis.

2.1.1 Active S-boxes

In general, there are two ways to show the guaranteed number of differential active S-boxes of blockciphers. One is to use proved lower bounds of active S-boxes, the other is to estimate the lower bounds by using a search algorithm. We checked the both methods for CLEFIA to see which approach has tighter lower bound. As a result, we found that the implied bounds by theoretical proofs are not tighter than the search based bounds. Therefore, we use the results obtained from computer search for the security estimation of CLEFIA.

Table 1 shows the guaranteed numbers of active S-boxes of CLEFIA obtained by the computer search. Now we focus on the columns indexed by 'DSM(D)' which show the guaranteed numbers of differential active S-boxes corresponding to the round numbers in columns indexed by 'r'.

2.1.2 Differential Probability of S-boxes used for Estimation

We need the differential probabilities of S-boxes to estimate the immunity against differential cryptanalysis. There are two S-boxes S_0 and S_1, each S-box has maximum differential probability $DP_{max}^{S_0} = 2^{-4.67}$ and $DP_{max}^{S_1} = 2^{-6.0}$, respectively. From designers' point of view, CLEFIA's security against differential cryptanalysis should be estimated assuming all S-boxes to be S_0, which has a larger maximum differential probability.

2.1.3 Differential Attack

Combining guaranteed 28 active S-boxes for 12-round CLEFIA and S_0's DP_{max}, it is shown that the maximum differential characteristic probabil-

Table 1: Guaranteed Numbers of Active S-boxes

r	DSM (D)	DSM (L)	r	DSM (D)	DSM (L)
1	0	0	14	34	34
2	1	1	15	36	36
3	2	5	16	38	39
4	6	6	17	40	42
5	8	10	18	44	46
6	12	15	19	46	48
7	14	16	20	50	50
8	18	18	21	52	52
9	20	20	22	55	55
10	22	23	23	56	58
11	24	26	24	59	62
12	28	30	25	62	64
13	30	32	26	65	66

ity $DCP_{max}^{\text{12-round}} \leq 2^{28\times(-4.67)} = 2^{-130.76}$. This means there is no useful 12-round differential characteristic for an attacker. Additionally, there are two reasons that the actual values of DCP_{max} is expected to be smaller than the above estimation. The first reason is that it is very difficult to construct a differential characteristic in which all the 28 active S-boxes use the highest differential probability $2^{-4.67}$ simultaneously. The second reason is that CLEFIA also employs S_1 with a smaller maximum differential probability. Consequently, it seems to be difficult for an attacker to find 12-round differentials which can be used to distinguish CLEFIA from random permutations. From this observation, we believe that the full-round CLEFIA is secure against differential cryptanalysis taking the most efficient key recovery attack into consideration.

2.2 Linear Cryptanalysis

Linear cryptanalysis is a general technique for the analysis of blockciphers, which was proposed by Matsui [26]. In order to evaluate the immunity against linear cryptanalysis, a similar method to differential attack can be used, which is a method utilizing the knowledge of the guaranteed number of active S-boxes and maximum linear probability of S-boxes.

The columns indexed by 'DSM(L)' in Table 1 show the guaranteed numbers of linear active S-boxes corresponding to the number of rounds in columns indexed by 'r'. Since the maximum linear probability $LP_{max}^{S_0} = 2^{-4.38}$ and $LP_{max}^{S_1} = 2^{-6.00}$ respectively, we assume all S-boxes in CLEFIA are S_0. Combining 30 active S-boxes for 12-round CLEFIA and S-box property, the maximum linear characteristic probability $LCP_{max}^{\text{12-round}} \leq 2^{30\times-4.38} = 2^{-131.40}$. It is difficult to construct a linear approximation in which all the 30 active S-boxes use the highest linear probability $2^{-4.38}$ simultaneously. Moreover, CLEFIA employs stronger S-box S_1 with $DP_{max}^{S_1} = 2^{-6.00}$, so that we believe that it is difficult for an attacker to find 12-round linear-hulls which can be used to distinguish CLEFIA from random permutations.

2.3 Differential-Linear Cryptanalysis

Differential-Linear cryptanalysis is a general technique for the analysis of blockciphers, which was proposed by Langford and Hellman [24]. This cryptanalysis uses both of differential characteristics and linear approximations. Letting p be a probability of a certain differential characteristic and letting q be a probability of a certain linear approximation, the complexity of the differential-linear cryptanalysis would have the complexity order of about p^2q^2. Based on characteristic based analysis, an 8-round distinguisher consists of a 3-round differential characteristic holding 2-active S-boxes with probability $2^{2\times(-4.67)} = 2^{-9.34}$ and a 5-round linear approximation holding 10-active S-boxes with probability $2^{10\times(-4.38)}$ is the best combination. This is smaller than the best differential distinguisher and best linear distinguisher. Therefore, we consider that full-round CLEFIA has strong immunity against differential-linear cryptanalysis.

2.4 Boomerang Attack

Boomerang attack is an adaptive chosen plaintext and ciphertext attack proposed by Wagner [38]. It is based on a pair of short differential characteristics used in a specially built quartet. The main idea behind the boomerang attack is to use two short differentials with high probabilities instead of one differential of more rounds with low probability.

Let n be the block size in bits and k be the key length in bits. We assume that CLEFIA $E : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n$ can be described as a cascade $E = E_1 \circ E_0$, such that for E_0 there exists a differential $\alpha \to \beta$ with probability p, and for E_1 there exists a differential $\gamma \to \delta$ with probability q. The boomerang attack uses the first characteristic ($\alpha \to \beta$) for E_0 with respect to the pairs (P_0, P_1) and (P_2, P_3), and uses the second characteristic ($\gamma \to \delta$) for E_1 with respect to the pairs (C_0, C_2) and (C_1, C_3). The attack is based on the following boomerang process:

- Ask for the encryption of a pair of plaintexts (P_0, P_1) such that $P_0 \oplus P_1 = \alpha$ and denote the corresponding ciphertexts by (C_0, C_1).
- Calculate $C_2 = C_0 \oplus \delta$ and $C_3 = C_1 \oplus \delta$, and ask for the decryption of the pair (C_2, C_3). Denote the corresponding plaintexts by (P_2, P_3).

- Check whether $P_2 \oplus P_3 = \alpha$.

It is easy to see that for a random permutation the probability that the last condition is satisfied is 2^{-n}. For E, however, the probability that the pair (P_0, P_1) is a right pair with respect to the first differential $(\alpha \rightarrow \beta)$ is p. The probability that both pairs (C_0, C_2) and (C_1, C_3) are right pairs with respect to the second differential is q^2. If all these are right pairs, then they satisfy

$$E_1^{-1}(C_2) \oplus E_1^{-1}(C_3) = \beta = E_0(P_2) \oplus E_0(P_3),$$

and thus, with probability p also $P_2 \oplus P_3 = \alpha$. Therefore, the total probability of this quartet of plaintexts and ciphertexts to satisfy the boomerang conditions is $(pq)^2$. Therefore,

$$pq > 2^{-64}$$

must hold for the boomerang distinguisher (and the boomerang key recovery attacks) to work on CLEFIA.

All possible values for p and q can be derived from Table 1 and additional investigation. Regarding the differential probability of the S-boxes, the maximum differential probability of S_0, which is constructed of 4-bit S-boxes, is $2^{-4.67}$, and the maximum differential probability of S_1, which is based on the inversion over GF(2^8), is 2^{-6}. Therefore, we evaluate the security against the boomerang attack using the value of $2^{-4.67}$, which is the maximum differential probability of S_0, from a pessimistic (designers') point of view.

We consider two types of boomerang distinguisher for CLEFIA, which are called distinguisher I and distinguisher II. As shown in Table 2, CLEFIA with at most 9 rounds can be distinguished from a random permutation by using distinguishers I and II. The distinguisher II is an alternative to the distinguisher I, whose estimated probability is worse for the attacker. The distinguisher III is not actually a distinguisher which is referred to show the evidence that 10-round extension from distinguisher I has too low probability. Since it is expected that key-recovery attacks can be mounted for CLEFIA with up to 9 or a few more rounds, the full-round CLEFIA has enough security against the boomerang attacks.

Some of differential characteristics used in the above distinguisher are shown here. For E_0 in I, the 3.5-round differential characteristic shown in Figure 1 is used, where $a \in \{0,1\}^{32}$, $b \in \{0,1\}^{32}$, $c \in \{0,1\}^{32}$ is a non-zero value such that $w_8(a) = 1$, $w_8(b) = 4$, $w_8(c) = 4$, respectively.

For E_1 in distinguisher I, the 5.5-round differential characteristic shown in Figure 2 (Left) is used, where $d \in \{0,1\}^{32}$, $e \in \{0,1\}^{32}$, $f \in \{0,1\}^{32}$, $g \in \{0,1\}^{32}$ $h \in \{0,1\}^{32}$ is a non-zero value such that $w_8(d) = 1$, $w_8(e) = 4$, $w_8(f) = 4$, $w_8(g) = 1$, $w_8(h) = 4$, respectively.

Moreover, E_1 in distinguisher III can be obtained from E_1 in distinguisher I by simply adding one round at the last iteration (right, Figure 2).

Table 2: Boomerang Distinguishers

Case	$E_1 \circ E_0$		E
I	(E_0, E_1)	(3.5-round, 5.5-round)	9-round
	(p, q)	$(\leq (2^{-4.68})^2, \leq (2^{-4.67})^8)$	$(pq)^2 \leq 2^{-93.40}$
II	(E_0, E_1)	(4.5-round, 4.5-round)	9-round
	(p, q)	$(\leq (2^{-4.68})^6, \leq (2^{-4.67})^6)$	$(pq)^2 \leq 2^{-112.08}$
III	(E_0, E_1)	(3.5-round, 6.5-round)	10-round
	(p, q)	$(\leq (2^{-4.68})^2, \leq (2^{-4.67})^{12})$	$(pq)^2 \leq 2^{-130.76}$

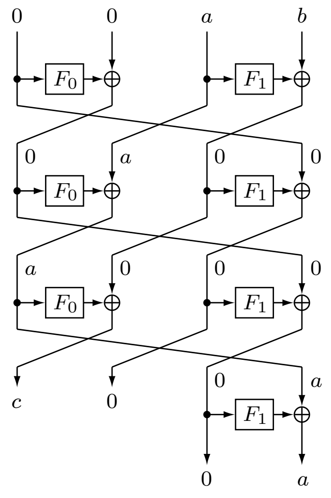

Figure 1: 3.5-round Differential Characteristic for E_0 (Distinguisher I)

2.5 Amplified Boomerang Attack

Amplified boomerang attack is a chosen plaintext variant of the boomerang attack [19]. The key idea behind the transformation is to encrypt many plaintext pairs with input difference α, and to look for quartets that conform to the requirements of the boomerang process.

The analysis in [19] shows that out of N plaintext pairs, the number of right quartets is expected to be $N^2 2^{-(n+1)} p^2 q^2$, where n is the block size in bits. Therefore, in the case of 9-round CLEFIA, it is expected to see one right quartet from $N = 2^{111.30}$ plaintext pairs. These plaintext pairs can be obtained from $2^{92.30}$ structures of $2^8 \times 4$ plaintext pairs. Therefore, the attack requires $2^{92.30} \times 2^8 \times 4 = 2^{102.30}$ chosen plaintexts.

In the amplified boomerang attack scenario, CLEFIA with at most 9 rounds can be distinguished from a random permutation. Since it is expected that key-recovery attacks can be mounted for CLEFIA with up to 9 or a few

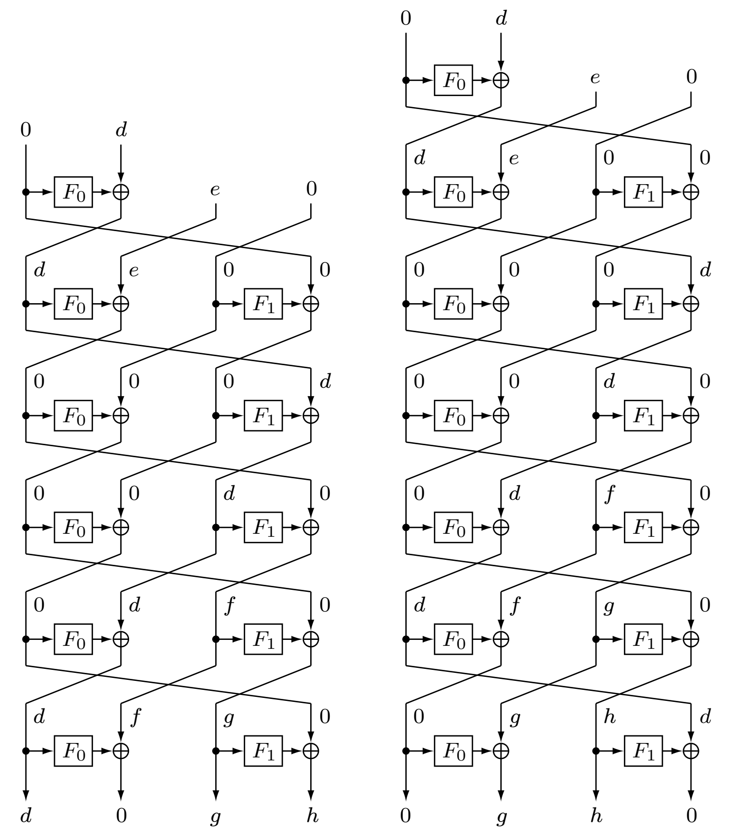

Figure 2: 5.5-round and 6.5-round Characteristics for E_1 (Distinguisher I, II)

more rounds, the full-round CLEFIA has enough security against amplified boomerang attacks.

2.6 Rectangle Attack

The rectangle attack shows that it is possible to use all the possible β and γ simultaneously, and presents additional improvements over the amplified boomerang attack. These improvements increase the probability of a quartet to be a right quartet and N plaintext pairs with input difference α are expected to produce $N^2 2^{-n}\hat{p}^2\hat{q}^2$ right quartets, where $\hat{p}$ and $\hat{q}$ are as defined as:

$$\hat{p} = \sqrt{\sum_{\beta} \mathrm{Pr}^2[\alpha \to \beta]}, \qquad \hat{q} = \sqrt{\sum_{\gamma} \mathrm{Pr}^2[\gamma \to \delta]}. \tag{1}$$

By using the above observation, the existence of 10-round distinguisher is strongly implied, because the current characteristic probability of 10-round CLEFIA for the boomerang attack is slightly smaller than the threshold 2^{-128} (see Table 2). Since it is expected that key-recovery attacks can be mounted for CLEFIA with up to 10 or a few more rounds, the full-round CLEFIA has enough security against rectangle attacks.

2.7 Truncated Differential Cryptanalysis

Truncated differential cryptanalysis is a general technique for the analysis of blockciphers, which was proposed by Knudsen [22]. Truncated differentials are differentials where only a part of the difference can be predicted. Due to the strong byte oriented design of CLEFIA, it is natural that truncated differential cryptanalysis using '0' and '1' to represent each byte data depending on existence of difference. This approach was adopted to evaluate many blockciphers including E2 and Camellia [18, 29, 30, 36]. So far, it is still an open problem to find systematic ways to evaluate immunity of Feistel-type blockciphers with SP-type F-functions against the truncated differential attack. This makes the evaluation of CLEFIA difficult.

However, we can learn from the results of E2 and Camellia because, the both algorithms and CLEFIA are Feistel-type structure. The big difference with regard to truncated differential attack between two algorithms is that E2 has SPS-type F-functions but Camellia has SP-type F-functions. Best known results for them are as follows : E2 has a known 7-round truncated differential, and Camellia without FL/FL^{-1} has a 9-round truncated differential [18, 29, 30, 36]. The lack of the second S-layer in Camellia can be considered to be a reason for producing the difference of estimated immunity against truncated differential attack.

We call modified CLEFIA whose F-function are SPS-type CLEFIA+S. Moreover we call modified CLEFIA+S whose F-functions do not use diffusion switching mechanism CLEFIA+S-D, in which only a single diffusion matrix is repeatedly used. We confirmed by a computer simulation that CLEFIA+S-D with 10 rounds (or more) does not have any useful truncated differentials. Suppose that there is a 9-round truncated differential for CLEFIA+S-D, we expect that full-round CLEFIA-D is expected to be strong against truncated differential cryptanalysis from the observation of the difference between E2 and Camellia. Moreover, if the DSM is enabled, the immunity is expected to be stronger than the above.

Since these are partial analysis of truncated differential attack obtained so far, we consider a more convincing evaluation method is required for the full-spec CLEFIA.

2.8 Truncated Linear Cryptanalysis

Truncated linear cryptanalysis is a general technique for the analysis of blockciphers, which was proposed by the designers of Camellia in the course of the evaluation of the cipher [1]. Due to the duality between differential and linear cryptanalysis, the security against truncated linear cryptanalysis by using a similar algorithm to truncated differential cryptanalysis can be evaluated [27]. Consequently, we believe that full-round CLEFIA even without DSM is strong against truncated differential cryptanalysis. Moreover, if the DSM is enabled, the immunity is expected to be stronger than the above.

2.9 Impossible Differential Cryptanalysis

The impossible differential is the differential which holds with probability zero, i.e., the differential which never happens. Using such impossible differentials, it is possible to eliminate wrong key candidates and thus find the correct key [4].

We found 9-round impossible differential characteristics for CLEFIA.

- $(0, \alpha, 0, 0) \overset{9r}{\not\to} (0, \alpha, 0, 0)$
- $(0, 0, 0, \alpha) \overset{9r}{\not\to} (0, 0, 0, \alpha)$

where $\alpha \in \{0,1\}^{32}$ is a non-zero value. Figure 3 shows the first 9-round impossible differential characteristic of the generalized Feistel structure employed by CLEFIA. In the figure, '+' denotes a non-zero difference and '*' denotes an unknown difference. The second characteristic is obtained by rotating the positions of all difference in Figure 3.

Table 3 shows the summary of the complexity required for the impossible differential attacks using these 9-round impossible differential characteristics. Table 3 shows that it is expected that full-round CLEFIA has enough security against impossible differential attacks. Details of these attacks are explained in the following sections.

Table 3: Summary of Impossible Differential Cryptanalysis

# of rounds	key length	key whitening	# of chosen plaintexts	time complexity
10	128, 192, 256	w/	$2^{101.7}$	2^{102}
11	192, 256	w/	$2^{103.5}$	2^{188}
12	256	w/o	$2^{103.8}$	2^{252}

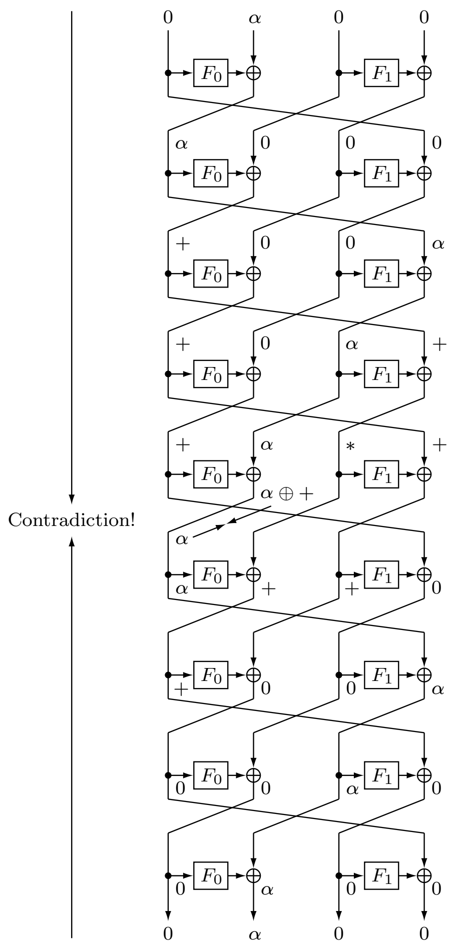

Figure 3: 9-round Impossible Differential Characteristic

2.9.1 Key Recovery Attack on 10-round CLEFIA

First, we describe an attack on the 10-round CLEFIA.

It is possible to derive the 32-bit round key RK_{18} of the 10th round by using the 9-round impossible differential $(0, \alpha, 0, 0) \overset{9r}{\not\rightarrow} (0, \alpha, 0, 0)$. It is also possible to derive the 32-bit round key RK_{19} of the 10th round by using the 9-round impossible differential $(0, 0, 0, \alpha) \overset{9r}{\not\rightarrow} (0, 0, 0, \alpha)$.

Key recovery of RK_{18} An attacker chooses a structure of 2^{32} plaintexts where the first, third, and fourth 32-bit words are held fixed and the second takes on over all 2^{32} possibilities, and then collects the pairs satisfying $\Delta C^{(10)} = (\alpha, \beta, 0, 0)$, where $\beta \in \{0, 1\}^{32}$ is a non-zero value. Note that $C^{(10)} = (C_0^{(10)}, C_1^{(10)}, C_2^{(10)}, C_3^{(10)})$ is the output of 10-round CLEFIA

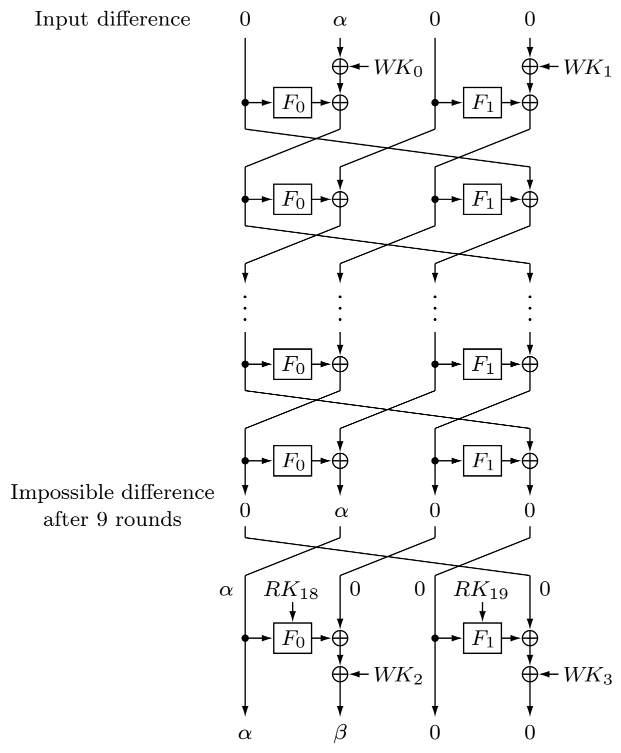

Figure 4: Impossible Differential Cryptanalysis on 10-round CLEFIA

including the key whitening.

The probability that a plaintext pair satisfies the condition above is $(\frac{1}{2^{32}})^3 \cdot (\frac{2^{32}-1}{2^{32}})^1 \simeq 2^{-96}$, because there are three 32-bit words with a fixed difference and one 32-bit word with non-zero difference in $\Delta C^{(10)}$.

Round key RK_{18} can be derived as follows:

1. Guess an element in the key space of RK_{18}
2. Using the collected ciphertexts and the guessed key value, calculate the output difference of the round function F_0 of the 10th round, denoted by ΔF_0.
3. Check whether the following condition is satisfied:

$$\Delta F_0 \oplus \Delta C_1^{(10)} = 0$$

4. If the condition above is satisfied, the guessed key value is not correct. Eliminate it from the key space.

The probability that an element in the key space survives the check with such a pair is $1 - 2^{-32}$. Therefore, let N be the number of the pairs required for narrowing down to one correct key, we have

$$2^{32}(1-2^{-32})^N = 1.$$

N is about $2^{36.7}$. Hence, the number of required chosen plaintext pairs is $2^{96} \cdot 2^{36.7} = 2^{132.7}$. Since the attacker can collect ${}_{2^{32}}C_2 = 2^{63}$ pairs from such one structure, he needs $2^{132.7-63} = 2^{69.7}$ structures. It follows that RK_{18} can be derived by using $2^{69.7} \cdot 2^{32} = 2^{101.7}$ chosen plaintexts.

The required time complexity is as follows:

- Computing ciphertexts: $2^{101.7}$ encryptions
- Finding RK_{18}: $2^{32} + 2^{32} \cdot (1-2^{-32}) + \cdots + 2^{32} \cdot (1-2^{-32})^{2^{36.7}-1}$ $\simeq 2^{64}$ F-function computations $< 2^{60}$ encryptions

Therefore, the required time complexity is at most 2^{102} encryptions.

In the same way as deriving RK_{18}, RK_{19} can be derived by using the other 9-round impossible differential.

2.9.2 Key Recovery Attack on 11-round CLEFIA

Next, we describe an attack on the 11-round CLEFIA. Here we consider the key whitenings, too.

In 192-bit and 256-bit key cases, it is possible to derive three round keys RK_{18}, RK_{20} and RK_{21} and a whitening key WK_3 by using the 9-round impossible differential $(0,\alpha,0,0) \overset{9r}{\not\rightarrow} (0,\alpha,0,0)$, with less complexity than exhaustive key search. Similarly, it is possible to derive three round keys RK_{19}, RK_{20} and RK_{21} and whitening key WK_2 by using the 9-round impossible differential $(0,0,0,\alpha) \overset{9r}{\not\rightarrow} (0,0,0,\alpha)$.

Key recovery of $\boldsymbol{RK_{18}}$, $\boldsymbol{RK_{20}}$, $\boldsymbol{RK_{21}}$, $\boldsymbol{WK_3}$ An attacker chooses a structure of 2^{32} plaintexts where the first, third, and fourth 32-bit words are held fixed and the second takes on over all 2^{32} possibilities, and then collects the pairs satisfying $\Delta C^{(11)} = (\beta,\gamma,0,\alpha)$, where $\beta \in \{0,1\}^{32}$ and $\gamma \in \{0,1\}^{32}$ are non-zero values.

The probability that a plaintext pair satisfies the condition above is $(\frac{1}{2^{32}})^2 \cdot (\frac{2^{32}-1}{2^{32}})^2 \simeq 2^{-64}$, because there are two 32-bit words with a fixed difference and two 32-bit words with non-zero difference in $\Delta C^{(11)}$.

Round keys and whitening key $(RK_{18}, RK_{20}, RK_{21}, WK_3) \in \mathcal{K}$ can be derived as follows:

1. Guess an element in the key space $\mathcal{K}$.
2. Using the collected ciphertexts and the guessed key value, calculate the output difference of the 9th round.

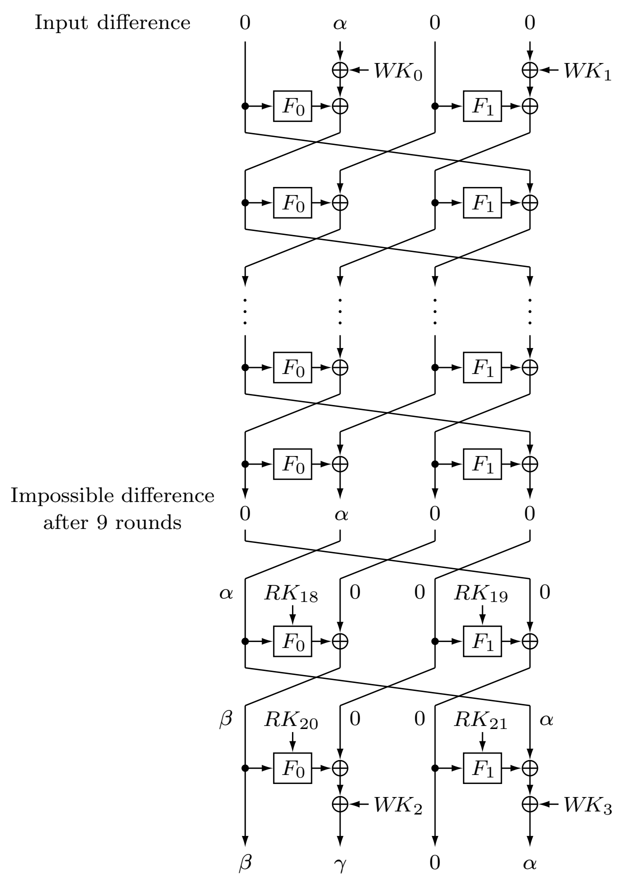

Figure 5: Impossible Differential Cryptanalysis on 11-round CLEFIA

3. Check whether the following conditions are satisfied:

$$\begin{aligned} \Delta C_2^{(9)} &= 0 \\ \Delta C_3^{(9)} &= 0 \end{aligned}$$

4. If the conditions above are satisfied, the guessed key value is not correct. Eliminate it from the key space.

The probability that an element in the key space survives the check with such a pair is $1-2^{-64}$. Therefore, let N be the number of the pairs required for narrowing down to one correct key, we have

$$2^{128}(1-2^{-64})^N = 1.$$

N is about $2^{70.5}$. Hence, the number of required chosen plaintext pairs is $2^{64} \cdot 2^{70.5} = 2^{134.5}$. Since the attacker can collect ${}_{2^{32}}C_2 = 2^{63}$ pairs from such one structure, he needs $2^{134.5-63} = 2^{71.5}$ structures. It follows that RK_{18}, RK_{20}, RK_{21}, and WK_3 can be derived by using $2^{71.5} \cdot 2^{32} = 2^{103.5}$ chosen plaintexts.

The required time complexity is as follows:

- Computing ciphertexts: $2^{103.5}$ encryptions
- Finding RK_{18}, RK_{20}, RK_{21}, WK_3:
  $2^{128} + 2^{128} \cdot (1 - 2^{-64}) + \cdots + 2^{128} \cdot (1 - 2^{-64})^{2^{70.1}-1} \simeq 2^{192}$ F-function computations $< 2^{188}$ encryptions

Therefore, the required time complexity is about 2^{188} encryptions. This complexity is less than exhaustive key search in 192-bit and 256-bit key cases.

2.9.3 Key Recovery Attack on 12-round CLEFIA

Next, we describe an attack on the 12-round CLEFIA. Here we don't consider the key whitenings.

In 256-bit key case, it is possible to derive five round keys RK_{18}, RK_{20}, RK_{21}, RK_{22}, and RK_{23} by using the 9-round impossible differential $(0, \alpha, 0, 0) \overset{9r}{\not\to} (0, \alpha, 0, 0)$, with less complexity than exhaustive key search. Similarly, it is possible to derive five round keys RK_{19}, RK_{20}, RK_{21}, RK_{22}, and RK_{23} by using the 9-round impossible differential $(0, 0, 0, \alpha) \overset{9r}{\not\to} (0, 0, 0, \alpha)$.

Key recovery of RK_{18}, RK_{20}, RK_{21}, RK_{22}, RK_{23} An attacker chooses a structure of 2^{32} plaintexts where the first, third, and fourth 32-bit words are held fixed and the second takes on over all 2^{32} possibilities, and then collects the pairs satisfying $\Delta C^{(12)} = (\gamma, \epsilon, \alpha, \delta)$, where $\beta \in \{0,1\}^{32}$, $\gamma \in \{0,1\}^{32}$, and $\epsilon \in \{0,1\}^{32}$ are non-zero values.

The probability that a plaintext pair satisfies the condition above is $(\frac{1}{2^{32}}) \cdot (\frac{2^{32}-1}{2^{32}})^3 \simeq 2^{-32}$, because there are a 32-bit word with a fixed difference and three 32-bit words with non-zero difference in $\Delta C^{(12)}$.

Round keys $(RK_{18}, RK_{20}, RK_{21}, RK_{22}, RK_{23}) \in \mathcal{K}$ can be derived as follows:

1. Guess an element in the key space $\mathcal{K}$.
2. Using the collected ciphertexts and the guessed key value, calculate the output difference of the 9th round.
3. Check whether the following conditions are satisfied:

$$\begin{aligned} \Delta C_0^{(9)} &= 0 \\ \Delta C_2^{(9)} &= 0 \\ \Delta C_3^{(9)} &= 0 \end{aligned}$$

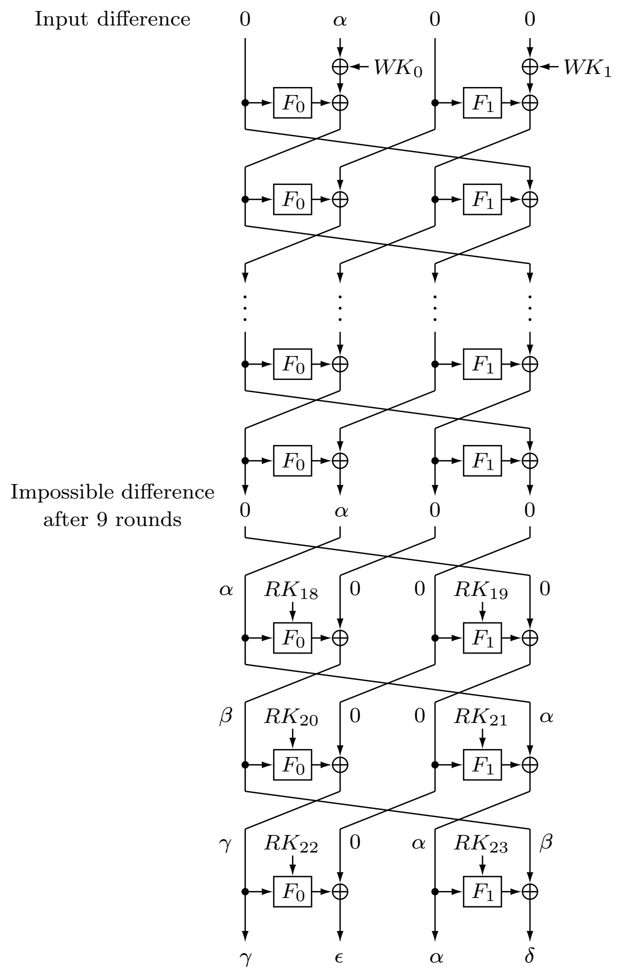

Figure 6: Impossible Differential Cryptanalysis on 12-round CLEFIA

4. If the conditions above are satisfied, the guessed key value is not correct. Eliminate it from the key space.

The probability that an element in the key space survives the check with such a pair is $1-2^{-96}$. Therefore, let N be the number of the pairs required for narrowing down to one correct key, we have

$$2^{160}(1-2^{-96})^N = 1.$$

N is about $2^{102.8}$. Hence, the number of required chosen plaintext pairs is $2^{32} \cdot 2^{102.8} = 2^{134.8}$. Since the attacker can collect ${}_{2^{32}}C_2 = 2^{63}$ pairs from such one structure, he needs $2^{134.8-63} = 2^{71.8}$ structures. It follows that RK_{18}, RK_{20}, RK_{21}, RK_{22}, RK_{23} can be derived by using $2^{71.8} \cdot 2^{32} = 2^{103.8}$ chosen plaintexts.

The required time complexity is as follows:

- Computing ciphertexts: $2^{103.8}$ encryptions
- Finding RK_{18}, RK_{20}, RK_{21}, RK_{22}, RK_{23} :
  $2^{160} + 2^{160} \cdot (1 - 2^{-96}) + \cdots + 2^{160} \cdot (1 - 2^{-96})^{2^{70.1}-1} \simeq 2^{256}$ F-function computations $< 2^{252}$ encryptions

Therefore, the required time complexity is about 2^{252} encryptions. This complexity is less than exhaustive key search in 256-bit key case.

2.10 Saturation Cryptanalysis

Saturation cryptanalysis, also known as multiset cryptanalysis, is a relatively new type of analytic method, which was first proposed by Daemen *et al.* [10] as a dedicated attack called "square attack" applied to the blockcipher Square.

2.10.1 Distinguisher based on Byte Saturation

Typically, the saturation cryptanalysis makes use of byte-oriented structure of blockciphers. The best attack against AES/Rijndael is known as this type of attack. Since CLEFIA also has strong byte oriented structure, we first consider the byte-based saturation cryptanalysis.

Let $X_i \in \{0,1\}^8$ be a set of 8-bit values. Now we categorize status of the set of X_i into four groups depending on conditions defined as follows.

- `Const (C)` : if $\forall i, j \quad X_i = X_j$,
- `All (A)` : if $\forall i, j \quad i \neq j \Leftrightarrow X_i \neq X_j$,
- `Balance (B)` : if $\bigoplus_i X_i = 0$,
- `Unknown (U)` : unknown.

Then consider the situation that there are 256 plaintexts in which all bytes are `Const` except only one `All` byte. Using the conditions, saturation relationship between input and output for 5-round CLEFIA can be written as follows.

- `((C C C C) (C C C A) (C C C C) (C C C C))`
  $\xrightarrow{5r}$ `((U U U U) (U U U U) (B B B B) (U U U U))`
- `((C C C C) (C C C C) (C C C C) (C C C A))`
  $\xrightarrow{5r}$ `((B B B B) (U U U U) (U U U U) (U U U U))`

Moreover, if `(C C C A)` is replaced by one of `(C C A C)`, `(C A C C)`, `(A C C C)`, the above output conditions are valid. As a result, there are eight saturation paths in total. The first saturation path is depicted in Figure 7. These paths make 5-round CLEFIA distinguishable from a random permutation since `Balance` is found at the output.

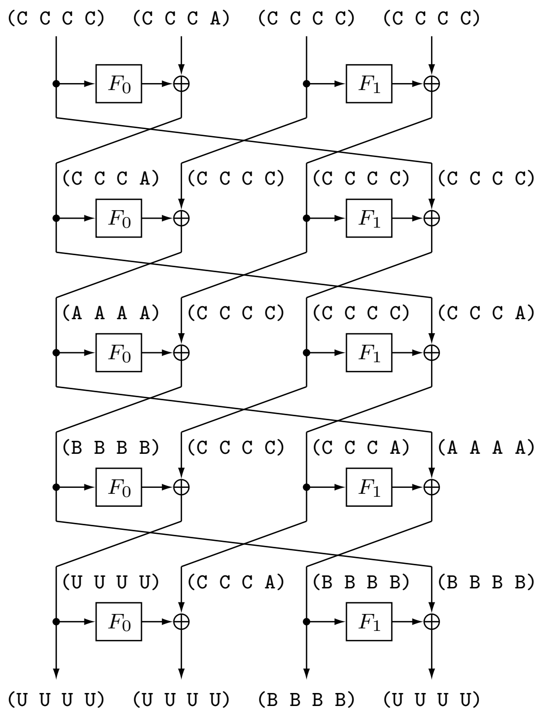
(C C C C) (C C C A) (C C C C) (C C C C)
F_0
F_1
(C C C A) (C C C C) (C C C C) (C C C C)
(A A A A) (C C C C) (C C C C) (C C C A)
(B B B B) (C C C C) (C C C A) (A A A A)
(U U U U) (C C C A) (B B B B) (B B B B)
(U U U U) (U U U U) (B B B B) (U U U U)

Figure 7: Saturation Characteristic

Then we assume that an attacker would execute key recovery attacks using the above saturation characteristics.

2.10.2 Key Recovery Attack on 7-round CLEFIA

We assume that there are additional 2 rounds after the 5-round saturation characteristic (see Figure 8), and RK_{10} and RK_{13} are the keys to be recovered. This attack uses the fact that only a part of ciphertext $C_0^{(7)}, C_2^{(7)}, C_3^{(7)}$ and round keys RK_{10}, RK_{13} are required to decrypt the 32-bit data which is saturated as `(B B B B)`.

In this setting, round keys $(RK_{10}, RK_{13}) \in \mathcal{K}$ can be derived as follows:

1. Guess an element $k_{guess10}$ for RK_{10} and $k_{guess13}$ for RK_{13} .

2. For each guessed key value,

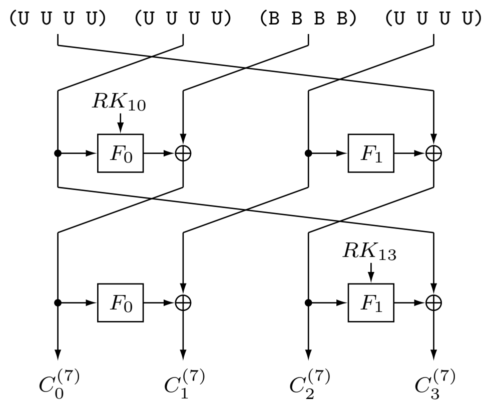

Figure 8: Key Recovery Attack

- For each ciphertext, compute

$$Z_i = F_0(k_{guess13}, C_2^{(7)}) \oplus C_3^{(7)} \ ,$$

then compute

$$Y_i = F_1(k_{guess10}, Z_i) \oplus C_0^{(7)} \ .$$

Compute the exclusive-or of all Y_i, $Y = \bigoplus_i Y_i$.

3. If $Y = 0$, then $k_{guess10}, k_{guess13}$ is a candidate for RK_{10}, RK_{13}, respectively. If $Y \neq 0$, then the guess is not the correct value for RK_{10}, RK_{13}, respectively.

The probability that a key candidate in the key space survives the above discarding step is expected to be 2^{-32}. Therefore, three sets of 256 plaintexts is enough for narrowing down to one correct key value. The time complexity is $2^{64} \times 2^8 \times 2^8$ calculations of F-function which is about 2^{80} F-function executions. Consequently, this attack is applicable to 128-bit, 192-bit and 256-bit keys.

Attacks for additional round may be possible, but we expect that a number of additional rounds is limited as far as the same saturation characteristic is used. However, if we extend the size of saturated word from 8-bit to 32-bit, we can attack longer rounds of CLEFIA. It is explained in the next section.

2.10.3 Distinguisher based on 32-bit Word Saturation

We consider the 32-bit word oriented saturation attack as follows.

Let $X_i \in \{0,1\}^{32}$ be a set of 32-bit values, then we categorize status of the set of X_i into four groups in the same fashion.

- `Const (C)` : if $\forall i, j \;\; X_i = X_j$,
- `All (A)` : if $\forall i, j \;\; i \neq j \Leftrightarrow X_i \neq X_j$,
- `Balance (B)` : if $\bigoplus_i X_i = 0$,
- `Unknown (U)` : unknown.

Using these extended conditions, the saturation relationship between input and output for 6-round CLEFIA can be written as follows.

- `(C A C C)` $\overset{6r}{\rightarrow}$ `(B U U U)`
- `(C C C A)` $\overset{6r}{\rightarrow}$ `(U U B U)`

The first saturation path is depicted in Figure 9. These paths make 6-round CLEFIA distinguishable from a random permutation since `Balance` is found at the output.

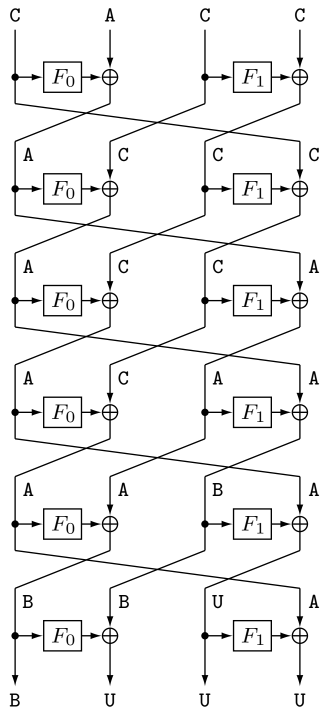

Figure 9: Saturation Characteristic (32-bit)

These 6-round distinguishers can be extended into 8-round distinguishers. We first explain how to extend to 7-round distinguishers. Let $\mathtt{A}_{(64)}$ be an `All` state of 64-bit words, and it is divided into two segments as $\mathtt{A}_{(64)} = \mathtt{A}_{0(64)} \mid \mathtt{A}_{1(64)}$. Using this, we can get the 7-round distinguisher as:

- $\texttt{(C C A}_{0(64)}\ \texttt{A}_{1(64)}\texttt{)} \xrightarrow{7r} \texttt{(B U U U)}$
- $\texttt{(A}_{0(64)}\ \texttt{A}_{1(64)}\ \texttt{C C)} \xrightarrow{7r} \texttt{(U U B U)}$

These distinguishers require 2^{64} plaintexts.

This is explained as follows. After the first round $\texttt{(C C A}_{0(64)}\ \texttt{A}_{1(64)}\texttt{)}$ becomes $\texttt{(C A}_{0(64)}\ \texttt{A'}_{1(64)}\ \texttt{C)}$ where the concatenated segment $\texttt{A}_{0(64)}\ \texttt{|}\ \texttt{A'}_{1(64)}$ is also `All`. It can be viewed that $\texttt{(C A}_{0(64)}\ \texttt{A'}_{1(64)}\ \texttt{C)}$ contains 2^{32} structures `(C A C C)` where the second rightmost constant takes all possible 2^{32} values. Therefore `Balance` is kept at the output which is directly suggested by the above 6-round distinguishers.

Extension to 8-round distinguisher is obtained in a similar way. Let $\texttt{A}_{(96)}$ be an `All` state of 96-bit words, and it is divide into three segments as $\texttt{A}_{(96)} = \texttt{A}_{0(96)} \mid \texttt{A}_{1(96)} \mid \texttt{A}_{2(96)}$. Using this, we can get the 8-round distinguishers as:

- $\texttt{(A}_{0(96)}\ \texttt{C A}_{1(96)}\ \texttt{A}_{2(96)}\texttt{)} \xrightarrow{8r} \texttt{(B U U U)}$
- $\texttt{(A}_{0(96)}\ \texttt{A}_{1(96)}\ \texttt{A}_{2(96)}\ \texttt{C)} \xrightarrow{8r} \texttt{(U U B U)}$

These distinguishers require 2^{96} plaintexts.

In the following, a 9-round attack and a 10-round attack are described.

2.10.4 Key Recovery Attack on 9-round CLEFIA

We assume that there is an additional round after the 8-round distinguishers, and RK_{17} is the key to be recovered.

1. Input a set of 2^{96} plaintexts which has the following format:
   $\texttt{(A}_{0(96)}\ \texttt{C A}_{1(96)}\ \texttt{A}_{2(96)}\texttt{)}$
2. For the output word $C_2^{(9)}$, count the frequencies of the values. Then make a list, $LIST$, of the 32-bit values with odd frequencies.
3. For the output word $C_3^{(9)}$, compute the exclusive-or of the all 2^{96} values, denoted Y.
4. For all candidates $l_i \in LIST$ and each guessed key value k_{guess} for RK_{17}, compute

   $$Z = \bigoplus_i F_1(k_{guess}, l_i) \ .$$

   - If $Z = Y$, then k_{guess} is a candidate for RK_{17}. If $Z \neq Y$, then k_{guess} is not the correct value for RK_{17}.

The probability that a key candidate in the key space survives the above discarding step is expected to be 2^{-32}. Therefore, three sets of 2^{32} plaintexts are enough for narrowing down to one correct key value. The time complexity is about 2^{31} calculations of F-function for $LIST$. Consequently, this attack is applicable to 128-bit, 192-bit and 256-bit keys.

2.10.5 Key Recovery Attack on 10-round CLEFIA

We assume that there are two additional rounds after the 8-round distinguishers, and RK_{17} and RK_{18} are the keys to be recovered. Basically, the scenario is the same as that of the 7-round attack on CLEFIA based on byte-oriented saturation described in Section 2.10.2.

In this setting, round keys (RK_{17}, RK_{18}) can be derived as follows:

1. Guess an element $k_{guess17}$ for RK_{17} and $k_{guess18}$ for RK_{18} .
2. For each guessed key value,
   - For each ciphertext, compute
     $$Z_i = F_0(k_{guess18}, C_0^{(10)}) \oplus C_1^{(10)} \quad ,$$
     then compute
     $$Y_i = F_1(k_{guess17}, Z_i) \oplus C_2^{(10)} \quad .$$
     Compute the exclusive-or of all Y_i, $Y = \bigoplus_i Y_i$.
3. If $Y = 0$, then $k_{guess17}, k_{guess18}$ is a candidate for RK_{17}, RK_{18}, respectively. If $Y \neq 0$, then the guess is not the correct value for RK_{17}, RK_{18}, respectively.

The probability that a key candidate in the key space survives the above discarding step is expected to be 2^{-32}. Therefore, more than two sets of 2^{32} plaintexts are required for narrowing down to one correct key value. The time complexity is about 2^{128} calculations of F-function because at most 2^{32} F_0 calculations and 2^{32} F_1 calculations are required for each guessed key (64 bits). Consequently, this attack is applicable to 128-bit, 192-bit and 256-bit keys.

Although we have shown several versions of key recovery attacks for reduced-round CLEFIA, the attackable numbers of rounds which will be extended in the future is expected within a few more rounds. Therefore, we believe full-round CLEFIA holds strong immunity against saturation cryptanalysis.

2.11 Gilbert-Minier Collision Attack

Gilbert-Minier Collision attack is a kind of saturation attack, which is proposed by Gilbert and Minier [14]. In their original paper, this attack utilized a special type of 4-round distinguisher customized for Rijndael. It seems to be difficult to apply the same distinguishing function to CLEFIA. Using this technique, 7-round Rijndael can be attacked as opposed to 6-round attack by the normal saturation attack. We also expect that similar technique can be applied to CLEFIA, but expected gaining is within a few rounds as well. Therefore, we believe full-round CLEFIA holds strong immunity against Gilbert-Minier Collision attack.

2.12 Higher Order Differential Cryptanalysis

This type of attack was developed in [16, 22, 23] and works well for blockciphers for which the nonlinear components can be represented as Boolean polynomials of low degree. The attack is based on the following fact: if the intermediate bits of the blockcipher are represented by Boolean polynomials of degree d, then the $(d+1)$-st order differential of the polynomial is zero.

For CLEFIA S-boxes S_0 and S_1, their degrees are 6 and 7, respectively. In more detail, the S-box S_0 consists of four smaller S-boxes, SS_0, SS_1, SS_2, and SS_3. Let $SS_i : \{0,1\}^4 \rightarrow \{0,1\}^4$ where $SS_i(x_0, x_1, x_2, x_3) = (y_0, y_1, y_2, y_3)$, and if we write y_j as a Boolean polynomial in (x_0, x_1, x_2, x_3), then we verified

$$\deg(y_j) = 3$$

holds for all $0 \leq j \leq 3$. Furthermore, let $S_0 : \{0,1\}^8 \rightarrow \{0,1\}^8$ where $S_0(x_0, x_1, \ldots, x_7) = (y_0, y_1, \ldots, y_7)$, and if we write y_j as a Boolean polynomial in $(x_0, x_1, \ldots, x_7)$, then we verified

$$\deg(y_j) = 6$$

holds for all $0 \leq j \leq 7$ by deriving concrete Boolean expressions.

Next, the S-box S_1 is based on the inversion function over GF(2^8), and it has the highest possible degree of 8-bit S-boxes. That is, let $S_1 : \{0,1\}^8 \rightarrow \{0,1\}^8$ where $S_1(x_0, x_1, \ldots, x_7) = (y_0, y_1, \ldots, y_7)$ and if we write y_j as a Boolean polynomial in $(x_0, x_1, \ldots, x_7)$, then we verified

$$\deg(y_j) = 7$$

for all $0 \leq j \leq 7$ by deriving concrete Boolean expressions.

Therefore, it is expected that the degree of an intermediate bit increases exponentially as the data passes through the S-boxes, whose degree is at least 6.

We expect that after passing three S-boxes, it is not possible to collect data for taking the $(d+1)$-st order differential since $6^3 > 128$. We believe that the higher order differential cryptanalysis has limited applications on CLEFIA, and the full round CLEFIA is strong enough against this attack.

2.13 Interpolation Cryptanalysis

This type of attack was proposed in [16] and it works for blockciphers for which the nonlinear components have a simple mathematical description. The principle of interpolation attack is that, if the ciphertext is represented as a polynomial or rational expression of the plaintext whose number of unknown coefficients is N, then the polynomial or rational expression can be constructed using N pairs of plaintexts and ciphertexts. If the attacker constructs the polynomial or rational expression, then it is possible to encrypt any plaintext into the corresponding ciphertext or decrypt any ciphertext into the corresponding plaintext without knowing the key. Since N determines the complexity and the number of pairs required for the attack, it is important to make N as large as possible. If N is so large that it is impractical for the attackers to collect N plaintext-ciphertext pairs, the blockcipher is secure against interpolation attack.

For CLEFIA S-boxes S_0 and S_1, we evaluated the number of terms to represent them as polynomials in GF(2^8). Let $S_0 : \mathrm{GF}(2^8) \rightarrow \mathrm{GF}(2^8)$ where $S_0(x) = y$, and if we write y as a polynomial in x, then we verified that the minimum number of terms is 244, where the minimum is taken over all irreducible polynomials.

Next, for S-box $S_1 : \mathrm{GF}(2^8) \rightarrow \mathrm{GF}(2^8)$ where $S_1(x) = y$, we verified that the minimum number of terms is 252, where the minimum is taken over all irreducible polynomials.

In both cases, the number of terms is close to the maximum value, 255, for a permutation over GF(2^8). Furthermore the use of two S-boxes S_0 and S_1 is likely to destroy any mathematical structure from the individual S-box in few rounds.

We believe it is very unlikely that the interpolation attack will be of any threat to CLEFIA.

2.14 XSL Cryptanalysis

A pure algebraic construction for the S-boxes have many interesting nonlinear properties. However, they may lead to the possible expression of a blockcipher as a system of sparse, over-defined low-degree multivariate polynomial equations over GF(2) or GF(2^8), and this fact may lead to attacks, as argued by Courtois and Pieprzyk in [9].

In what follows, we estimate the complexity of an XSL attack against modified version of CLEFIA, called CLEFIA-I, by following the very same methodology than [9]. We consider the first XSL attack as in [17], where the goal of the attack is to derive the round keys. Thus, in this scenario, we do not consider the key scheduling part.

CLEFIA-I is obtained from CLEFIA by simply replacing all 4-bit S-boxes, SS_0, SS_1, SS_2, and SS_3, by an identity function I, i.e., $I : \{0,1\}^4 \rightarrow$

$\{0,1\}^4$, where $I(x) = x$. Notice that attacking CLEFIA-I is much easier than attacking CLEFIA since the replaced 4-bit S-boxes do not contribute to increasing the security against XSL attacks.

According to Courtois and Pieprzyk [9], the complexity of the XSL attack can be estimated to

$$T^{\omega} \text{ with } T \approx (t-\rho)^P \binom{S_{\text{inv}}}{P}, \tag{2}$$

where:

- T is the total number of terms,
- ω is the complexity exponent of a Gaussian reduction,
- t is the number of monomials to represent the S-box S_1,
- ρ is the number of equations,
- S_{inv} is the number of S-boxes considered during the attack, and
- The integer, P, is the parameter of the attack.

Now if the S-box on s bits is an affine transformation of the inverse function in $\mathrm{GF}(2^s)$, then it will give $\rho = 3s - 1$ bi-affine equations true with probability 1, and one additional equation true with probability $1 - 2^{-s}$ [9]. Based on the similar observation than in [9], we have $t = 81$ and $\rho = 23$.

Next S_{inv} is the total number of S-boxes S_1 considered during an attack. Then for $r = 18$ we have

$$S_{\text{inv}} = 2 \times 2 \times 18 \times 2 = 144,$$

since each F_i is built from two S-boxes, there are two F-functions in one round, there are 18 rounds, and we need 2 plaintext-ciphertext pairs. Note that we need 2 known plaintext-ciphertext pairs to uniquely determine the round keys, since round keys involve both K and L. Similarly, we have $S_{\text{inv}} = 2 \times 2 \times 22 \times 4 = 352$ for $r = 22$, and $S_{\text{inv}} = 2 \times 2 \times 26 \times 4 = 416$ for $r = 26$, since we need at least 4 known plaintext-ciphertext pairs to uniquely determine the round keys, as round keys involve K_L, K_R, L_L, and L_R.

There are conditions on the parameter of the attack, P [9]. According to [17], P is given by

$$P = \frac{t-\rho}{s + \frac{t'}{S_{\text{inv}}}},$$

where $t' = 25$ in our case. This gives $P = 8$ for $r = 18$, 22, and 26.

Courtois and Pieprzyk [9] assume that $\omega = 2.376$, which is the best known value obtained by Coppersmith and Winograd [8]. According to [9], the constant factor in this algorithm is unknown to the authors of [8], and

Table 4: Estimations of Complexity of XSL Attacks against CLEFIA-I.

	$\omega = 2.376$	$\omega = 3$
$r = 18$	2^{216}	2^{273}
$r = 22$	2^{242}	2^{306}
$r = 26$	2^{247}	2^{312}

is expected to be very big. Accordingly, it is disputed whether such an algorithm can be applied efficiently in practice. For this reason, we will consider both $\omega = 2.376$ and $\omega = 3$ in our estimations.

Given above values and based on Eq.(2), the total number of terms can be estimated as $T = 81^8\binom{144}{8} > 2^{50+41} = 2^{91}$ for CLEFIA-I with $r = 18$, which gives the complexity $T^{2.376} = 2^{216}$ and $T^3 = 2^{273}$. For CLEFIA-I with $r = 22$, we have $T = 81^8\binom{352}{8} > 2^{50+52} = 2^{102}$, and thus $T^{2.376} = 2^{242}$ and $T^3 = 2^{306}$. Finally, for CLEFIA-I with $r = 26$, we have $T = 81^8\binom{416}{8} > 2^{50+54} = 2^{104}$, $T^{2.376} = 2^{247}$, and $T^3 = 2^{312}$.

A summary of our estimations is given in Table 4. At the light of the previous discussion, we should interpret these figures with an extreme care: on the one hand, the real complexity of XSL attacks is by no means clear at the time of writing and is the subject of much controversy [25, 31].

Furthermore, we are eliminating S_0, which is the highly impractical and most pessimistic hypotheses that S_0 has no contribution on the strength against XSL attacks. We believe the actual attack against original CLEFIA must be much harder than this estimation.

So far, we have considered the XSL attack on GF(2), and the XSL technique on GF(2^8) may lead to an efficient key recovery attack [9]. At the time of this writing, it is not possible to check the effectiveness of this approach as pointed out in [25], and we believe that the XSL estimates do not have the accuracy needed to substantiate claims of the existence of an efficient key recovery attack based on the XSL technique. Still, we believe two S-boxes, S_0 and S_1, make the applicability of the XSL technique substantially harder than the single S-box case, and prevent possible attacks based on the XSL attacks.

2.15 χ^2 Cryptanalysis

The χ^2 cryptanalysis is a kind of statistical attack for the analysis of block-ciphers. This attack was originally proposed by Vaudenay [37], and was applied to RC6 by Gilbert et al. [13] and Knudsen and Meier [21], independently. In [21], Knudsen and Meier attacked up to 15-round RC6 with general keys and 17-round RC6 with weak keys. The vulnerability of RC6 against χ^2 cryptanalysis consists in the use of data dependent rotations in the design of RC6. We consider there is no correlation useful for χ^2 attacks in the design of CLEFIA. Therefore, we believe the χ^2 cryptanalysis does not work on the full-round CLEFIA.

3 Cryptanalysis II — Key Scheduling Part

In this section, security aspects of the key scheduling part of CLEFIA are described.

3.1 Slide Attack

Slide attack is a general technique for the analysis of a key scheduling part of blockciphers, which was proposed by Biryukov and Wagner [7]. So far, it is known that there is a good countermeasure against slide attack using round constants independent of each round. In CLEFIA there are round constants shown in Specification [34]. Therefore CLEFIA is expected enough immunity against the slide attack.

3.2 Related-Cipher Attack

Related-cipher attack is a general technique for the analysis of a key scheduling part of blockciphers, which was proposed by Wu [39]. Consider a blockcipher whose numbers of rounds vary depending on the lengths of key, and whose round key set for all key length are identical expect the round keys for additional rounds. In this case, these different round blockciphers are called 'related', and the longer-round cipher will be easily crypt-analyzed by using shorter-round encryption results if round keys of them are the same. Due to the similarity between the key scheduling algorithms of 192-bit key and 256-bit key of CLEFIA, there is a risk for this attack. In order to avoid the related cipher attack, CLEFIA uses different sets of round constants for each key length [34]. This is the same case as the slide attack. Therefore CLEFIA is expected to have enough immunity against the related cipher attack.

3.3 Related-Key Cryptanalysis

Biham proposed related-key cryptanalysis [3]. This attack considers the information that can be extracted from two encryptions using related keys. The concept was used in [20] to present the idea of related-key differentials. These differentials study the development of differences in two encryptions under two related keys.

A related-key differential is a triplet of a plaintext difference ΔP, a ciphertext difference ΔC, and a key difference ΔK, such that

$$\Pr_{P,K}[E_K(P) \oplus E_{K \oplus \Delta K}(P \oplus \Delta P) = \Delta C]$$

is high enough (or zero).

As for CLEFIA with 128-bit keys, L is generated by using $GFN_{4,12}$. As in Table 1, $GFN_{4,12}$ has at least 28 active S-boxes, and we have $DCP_{max} \leq$

$2^{28\times(-4.67)} = 2^{-130.76}$. Therefore, for any ΔK and ΔL, a differential probability of $(\Delta K \rightarrow \Delta L)$ is expected to be less than 2^{-128}, i.e., no useful differential $(\Delta K \rightarrow \Delta L)$ exists. This implies the probability of any related-key differential $(\Delta P, \Delta C, \Delta K)$ is less than 2^{-128}, if all the information on ΔL is needed, since all the bits in L are used in 2 consecutive rounds. Other types of distinguishers may use $(\Delta K \rightarrow \Delta L)$, where some of the words in ΔL are unknown. We consider these types of distinguishers have limited effect since the unknown word propagates to all words in at least 3 rounds. Also we are not aware of any related-key differential with probability zero.

For 192 and 256-bit keys, (L_L, L_R) is generated by applying $GFN_{8,10}$, where $GFN_{8,10}$ is a 10-round generalized Feistel structure with 8 data lines. The round keys for $GFN_{8,10}$ are fixed constants determined by the key length. From Table 3 in [35], it has at least 29 differential active S-boxes, which implies there are no differential characteristics with probability more than 2^{-128}. That is, for any $(\Delta K_L, \Delta K_R)$ and $(\Delta L_L, \Delta L_R)$, a differential probability of $((\Delta K_L, \Delta K_R) \rightarrow (\Delta L_L, \Delta L_R))$ is expected to be less than 2^{-128}. Therefore, the probability of any related-key differential $(\Delta P, \Delta C, (\Delta K_L, \Delta K_R))$ is less than 2^{-128}, if all the values of $(\Delta L_L, \Delta L_R)$ are needed, since all the bits in L_L and L_R are used as round keys in at least 6 consecutive rounds.

Other types of distinguishers may use $((\Delta K_L, \Delta L_R) \rightarrow (\Delta L_L, \Delta L_R))$, where some of the words in $(\Delta L_L, \Delta L_R)$ are unknown, but this is not effective as in the case for 128-bit keys.

Therefore, we believe full-round CLEFIA holds strong immunity against related-key cryptanalysis.

3.4 Related-Key Boomerang Cryptanalysis

The main idea behind the attack is to use two short related-key differentials with high probabilities instead of one long related-key differential with a low probability.

Let n be the block size in bits and k be the key length in bits. As in the case for the boomerang attack, we assume that CLEFIA $E : \{0,1\}^n \times \{0,1\}^k \rightarrow \{0,1\}^n$ can be described as a cascade $E = E_1 \circ E_0$, such that for E_0 there exists a related-key differential $\alpha \rightarrow \beta$ under a key difference ΔK_0 with probability p, and for E_1 there exists a related-key differential $\gamma \rightarrow \delta$ under a key difference ΔK_1 with probability q.

The related-key boomerang process involves four different unknown (but related) keys: K_a, $K_b = K_a \oplus \Delta K_0$, $K_c = K_a \oplus \Delta K_1$, and $K_d = K_a \oplus \Delta K_0 \oplus \Delta K_1$. The attack is performed by the following algorithm:

- Choose a plaintext P_a at random and compute $P_b = P_a \oplus \alpha$.
- Ask for the encryption of P_x under K_x, i.e., $C_a = E_{K_a}(P_a)$ and $C_b = E_{K_b}(P_b)$.

- Compute $C_c = C_a \oplus \delta$ and $C_d = C_b \oplus \delta$.
- Ask for the decryption of C_x under K_x, i.e., $P_c = E^{-1}_{K_c}(C_c)$ and $P_d = E^{-1}_{K_d}(C_d)$.
- Test whether $P_c \oplus P_d = \alpha$.

It is easy to see that for a random permutation the probability that the last condition is satisfied is 2^{-n}. For E the probability that this condition is satisfied is p^2q^2 just like for a regular boomerang attack, and we need $(pq)^2 > 2^{-128}$, i.e., $pq > 2^{-64}$, in order to attack CLEFIA.

As we have seen in the previous section, since we employ $\mathit{GFN}_{4,12}$ for 128-bit key and $\mathit{GFN}_{8,10}$ for 192/256-bit keys in CLEFIA, it is very hard to achieve the condition $pq > 2^{-64}$. Indeed, we are not aware of E_0 and E_1 with more than a few rounds satisfying this condition.

Therefore, we believe it is very unlikely that the attack will be of any threat to CLEFIA.

3.5 Related-Key Rectangle Cryptanalysis

The transformation of the related-key boomerang attack into a related-key rectangle attack is similar to the transformation of the boomerang attack into the rectangle attack. Assume that E can be decomposed as before, where α, δ, $\hat{p}$, $\hat{q}$, K_a, K_b, K_c, and K_d have the same meaning. Then, the related-key rectangle distinguisher is as follows:

- Choose N plaintext pairs (P_a, P_b), where $P_b = P_a \oplus \alpha$, at random and ask for the encryption of P_a under K_a and of P_b under K_b.
- Choose N plaintext pairs (P_c, P_d), where $P_d = P_c \oplus \alpha$, at random and ask for the encryption of P_c under K_c and of P_d under K_d.
- Search for quartets of plaintexts (P_a, P_b, P_c, P_d) and the corresponding ciphertexts (C_a, C_b, C_c, C_d), satisfying $C_a \oplus C_c = C_b \oplus C_d = \delta$.

Then starting with N plaintext pairs with input difference α to be encrypted under K_a and K_b, we expect $N^2 2^{-n}(\hat{p}\hat{q})^2$ right quartets.

The attack requires $(\hat{p}\hat{q})^2 > 2^{-128}$ in order to apply against CLEFIA, however, we have not found E_0 and E_1 more than a few rounds satisfying this condition. Therefore, we believe CLEFIA is strong enough against this attack.

4 Software Implementations

This section describes the software performance results of CLEFIA. CLEFIA with 128-bit key achieves 1.48 Gbps on 2.4 GHz AMD Athlon 64 processor.

4.1 Optimization Techniques

CLEFIA can be implemented efficiently in software on various platforms including 32-bit and 64-bit processors.[1]

4.1.1 Encryption: 32-bit Processor Case

We present how to implement the encryption part of CLEFIA with 128-bit key efficiently. We don't refer to CLEFIA with 192/256-bit key because the same techniques are applicable to them except the number of rounds.

Let (x_0, x_1, x_2, x_3) be an input of F-function F_0 without the key addition layer and (y_0, y_1, y_2, y_3) be an output of F_0. Similarly, let (x_4, x_5, x_6, y_7) and (y_4, y_5, y_6, y_7) be an input and an output of F-function F_1 without the key addition layer, respectively. The relationship of the input and the output is described as follows:

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} \texttt{01} & \texttt{02} & \texttt{04} & \texttt{06} \\ \texttt{02} & \texttt{01} & \texttt{06} & \texttt{04} \\ \texttt{04} & \texttt{06} & \texttt{01} & \texttt{02} \\ \texttt{06} & \texttt{04} & \texttt{02} & \texttt{01} \end{pmatrix} \begin{pmatrix} S_0(x_0) \\ S_1(x_1) \\ S_0(x_2) \\ S_1(x_3) \end{pmatrix}$$

$$\begin{pmatrix} y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} \texttt{01} & \texttt{08} & \texttt{02} & \texttt{0A} \\ \texttt{08} & \texttt{01} & \texttt{0A} & \texttt{02} \\ \texttt{02} & \texttt{0A} & \texttt{01} & \texttt{08} \\ \texttt{0A} & \texttt{02} & \texttt{08} & \texttt{01} \end{pmatrix} \begin{pmatrix} S_1(x_4) \\ S_0(x_5) \\ S_1(x_6) \\ S_0(x_7) \end{pmatrix}$$

The elements of the above matrices are represented in hexadecimal form. We describe optimization techniques on this notation.

We explain the following two types of implementation techniques on 32-bit processors.

Type 1

If a 32-bit processor has a sufficiently large primary cache, we can use four tables of an 8-bit input and a 32-bit output per F-function for fast implementation. This implementation requires the following 8 tables:

[1]It is supposed to be implemented efficiently on 8-bit processors because of its strong byte-oriented structure, but we have not precisely evaluated performance on 8-bit processors yet.

$$\begin{array}{rcl}
T_{00}(x) &=& (\quad S_0(x),\ \{\texttt{02}\}\times S_0(x),\ \{\texttt{04}\}\times S_0(x),\ \{\texttt{06}\}\times S_0(x)\,) \\
T_{01}(x) &=& (\,\{\texttt{02}\}\times S_1(x),\ S_1(x),\ \{\texttt{06}\}\times S_1(x),\ \{\texttt{04}\}\times S_1(x)\,) \\
T_{02}(x) &=& (\,\{\texttt{04}\}\times S_0(x),\ \{\texttt{06}\}\times S_0(x),\ S_0(x),\ \{\texttt{02}\}\times S_0(x)\,) \\
T_{03}(x) &=& (\,\{\texttt{06}\}\times S_1(x),\ \{\texttt{04}\}\times S_1(x),\ \{\texttt{02}\}\times S_1(x),\ S_1(x)\,) \\
T_{10}(x) &=& (\quad S_1(x),\ \{\texttt{08}\}\times S_1(x),\ \{\texttt{02}\}\times S_1(x),\ \{\texttt{0A}\}\times S_1(x)\,) \\
T_{11}(x) &=& (\,\{\texttt{08}\}\times S_0(x),\ S_0(x),\ \{\texttt{0A}\}\times S_0(x),\ \{\texttt{02}\}\times S_0(x)\,) \\
T_{12}(x) &=& (\,\{\texttt{02}\}\times S_1(x),\ \{\texttt{0A}\}\times S_1(x),\ S_1(x),\ \{\texttt{08}\}\times S_1(x)\,) \\
T_{13}(x) &=& (\,\{\texttt{0A}\}\times S_0(x),\ \{\texttt{02}\}\times S_0(x),\ \{\texttt{08}\}\times S_0(x),\ S_0(x)\,)
\end{array}$$

Next, we compute the following equations:

$$\begin{array}{rcl}
(y_0, y_1, y_2, y_3) &=& T_{00}(x_0) \oplus T_{01}(x_1) \oplus T_{02}(x_2) \oplus T_{03}(x_3) \\
(y_4, y_5, y_6, y_7) &=& T_{10}(x_4) \oplus T_{11}(x_5) \oplus T_{12}(x_6) \oplus T_{13}(x_7)
\end{array}$$

The required operations for this implementation are estimated as follows:

Size of table (KB):	8
Operation per round (18 rounds in total)	
# of table lookups:	8
# of XORs in F-function:	6
# of XORs out of F-function:	2
# of XORs for round key addition:	2
# of XORs for key whitening:	4

Type 2

We show another implementation technique on a 32-bit processor which can reduce the size of table to half of the above implementation by introducing rotation operations. This implementation is preferable when the processor has a smaller primary cache than 8 KB and a rotation operation on the processor is relatively fast.

In the implementation, the tables $T_{02}(x)$, $T_{03}(x)$, $T_{12}(x)$ and $T_{13}(x)$ in Type 1 implementation are not required to prepare. The tables are generated from the other tables as follows:

$$\begin{array}{rcl}
T_{02}(x) &=& T_{00}(x) \lll 16 \\
T_{03}(x) &=& T_{01}(x) \lll 16 \\
T_{12}(x) &=& T_{10}(x) \lll 16 \\
T_{13}(x) &=& T_{11}(x) \lll 16
\end{array}$$

This implementation requires the following operations.

Size of table (KB):	4
Operation per round (18 rounds in total)	
# of table lookups:	8
# of rotation:	4
# of XORs in F-function:	6
# of XORs out of F-function:	2
# of XORs for round key addition:	2
# of XORs for key whitening:	4

Note that the size of table is reduced to half, from 8KB to 4KB, compared to Type 1 implementation.

4.1.2 Encryption: 64-bit Processor Case

Here we explain the following four types of implementation techniques on 64-bit processors. A round function of encryption of CLEFIA with 128-bit key can be transformed equivalently as shown in Figure 10.

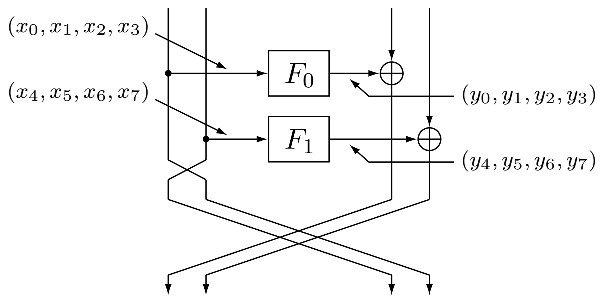

Figure 10: A round function of encryption using implementations on 64-bit processors

Type 1

If a 64-bit processor has a primary cache whose size is equal to or more than 16KB, we can use eight tables of an 8-bit input and a 64-bit output for a round function as follows.

$$\begin{aligned}
T_{00}(x) &= (\ S_0(x),\ \{\texttt{02}\} \times S_0(x),\ \{\texttt{04}\} \times S_0(x),\ \{\texttt{06}\} \times S_0(x),\ 0,\ 0,\ 0,\ 0\ ) \\
T_{01}(x) &= (\ \{\texttt{02}\} \times S_1(x),\ S_1(x),\ \{\texttt{06}\} \times S_1(x),\ \{\texttt{04}\} \times S_1(x),\ 0,\ 0,\ 0,\ 0\ ) \\
T_{02}(x) &= (\ \{\texttt{04}\} \times S_0(x),\ \{\texttt{06}\} \times S_0(x),\ S_0(x),\ \{\texttt{02}\} \times S_0(x),\ 0,\ 0,\ 0,\ 0\ ) \\
T_{03}(x) &= (\ \{\texttt{06}\} \times S_1(x),\ \{\texttt{04}\} \times S_1(x),\ \{\texttt{02}\} \times S_1(x),\ S_1(x),\ 0,\ 0,\ 0,\ 0\ )
\end{aligned}$$

$$\begin{aligned}
T_{10}(x) &= (\ 0,\ 0,\ 0,\ 0,\ S_1(x),\ \{\texttt{08}\} \times S_1(x),\ \{\texttt{02}\} \times S_1(x),\ \{\texttt{0A}\} \times S_1(x)\ ) \\
T_{11}(x) &= (\ 0,\ 0,\ 0,\ 0,\ \{\texttt{08}\} \times S_0(x),\ S_0(x),\ \{\texttt{0A}\} \times S_0(x),\ \{\texttt{02}\} \times S_0(x)\ ) \\
T_{12}(x) &= (\ 0,\ 0,\ 0,\ 0,\ \{\texttt{02}\} \times S_1(x),\ \{\texttt{0A}\} \times S_1(x),\ S_1(x),\ \{\texttt{08}\} \times S_1(x)\ ) \\
T_{13}(x) &= (\ 0,\ 0,\ 0,\ 0,\ \{\texttt{0A}\} \times S_0(x),\ \{\texttt{02}\} \times S_0(x),\ \{\texttt{08}\} \times S_0(x),\ S_0(x)\ )
\end{aligned}$$

Using these tables, we compute the following equation:

$$\begin{aligned}(y_0, y_1, y_2, y_3, y_4, y_5, y_6, y_7) &= T_{00}(x_0) \oplus T_{01}(x_1) \oplus T_{02}(x_2) \oplus T_{03}(x_3) \oplus \\ &\quad T_{10}(x_4) \oplus T_{11}(x_5) \oplus T_{12}(x_6) \oplus T_{13}(x_7)\end{aligned}$$

This implementation requires the following operations.

Size of table (KB):	16
Operation per round (18 rounds in total)	
# of table lookups:	8
# of XORs in F-function:	7
# of XORs out of F-function:	1
# of XORs for round key addition:	1
# of swap (rotation):	1
# of XORs for key whitening:	2

Note that the rotation operations are required in order to swap $(x_0, x_1, x_2 x_3)$ and (x_4, x_5, x_6, x_7) as shown in Figure 10 .

Type 2

If a 64-bit processor has a sufficiently large primary cache, e.g. 32KB, we can avoid a swap operation per round by adding the following tables to Type 1 implementation.

$$\begin{aligned}
T_{04}(x) &= (\,0,\ 0,\ 0,\ 0,\ S_0(x),\ \{\mathtt{02}\} \times S_0(x),\ \{\mathtt{04}\} \times S_0(x),\ \{\mathtt{06}\} \times S_0(x)\,)\\
T_{05}(x) &= (\,0,\ 0,\ 0,\ 0,\ \{\mathtt{02}\} \times S_1(x),\ S_1(x),\ \{\mathtt{06}\} \times S_1(x),\ \{\mathtt{04}\} \times S_1(x)\,)\\
T_{06}(x) &= (\,0,\ 0,\ 0,\ 0,\ \{\mathtt{04}\} \times S_0(x),\ \{\mathtt{06}\} \times S_0(x),\ S_0(x),\ \{\mathtt{02}\} \times S_0(x)\,)\\
T_{07}(x) &= (\,0,\ 0,\ 0,\ 0,\ \{\mathtt{06}\} \times S_1(x),\ \{\mathtt{04}\} \times S_1(x),\ \{\mathtt{02}\} \times S_1(x),\ S_1(x)\,)\\
T_{14}(x) &= (\,S_1(x),\ \{\mathtt{08}\} \times S_1(x),\ \{\mathtt{02}\} \times S_1(x),\ \{\mathtt{0A}\} \times S_1(x),\ 0,\ 0,\ 0,\ 0\,)\\
T_{15}(x) &= (\,\{\mathtt{08}\} \times S_0(x),\ S_0(x),\ \{\mathtt{0A}\} \times S_0(x),\ \{\mathtt{02}\} \times S_0(x),\ 0,\ 0,\ 0,\ 0\,)\\
T_{16}(x) &= (\,\{\mathtt{02}\} \times S_1(x),\ \{\mathtt{0A}\} \times S_1(x),\ S_1(x),\ \{\mathtt{08}\} \times S_1(x),\ 0,\ 0,\ 0,\ 0\,)\\
T_{17}(x) &= (\,\{\mathtt{0A}\} \times S_0(x),\ \{\mathtt{02}\} \times S_0(x),\ \{\mathtt{08}\} \times S_0(x),\ S_0(x),\ 0,\ 0,\ 0,\ 0\,)
\end{aligned}$$

We appropriately select one of the following equations round by round.

$$\begin{aligned}(y_0, y_1, y_2, y_3, y_4, y_5, y_6, y_7) &= T_{00}(x_0) \oplus T_{01}(x_1) \oplus T_{02}(x_2) \oplus T_{03}(x_3) \oplus \\ &\quad T_{10}(x_4) \oplus T_{11}(x_5) \oplus T_{12}(x_6) \oplus T_{13}(x_7) \\ (y_0, y_1, y_2, y_3, y_4, y_5, y_6, y_7) &= T_{04}(x_0) \oplus T_{05}(x_1) \oplus T_{06}(x_2) \oplus T_{07}(x_3) \oplus \\ &\quad T_{14}(x_4) \oplus T_{15}(x_5) \oplus T_{16}(x_6) \oplus T_{17}(x_7)\end{aligned}$$

This implementation requires the following operations.

Size of table (KB):	32
Operation per round (18 rounds in total)	
# of table lookups:	8
# of XORs in F-function:	7
# of XORs out of F-function:	1
# of XORs for round key addition:	1
# of XORs for key whitening:	2

The advantage of this implementation over Type 1 implementation is that we require no swap operation.

Type 3

We show another implementation on a 64-bit processor which can reduce the size of table to half of Type 1 implementation by merging tables T_{0i} and T_{1i} into the following table T_{2i} for $0 \le i \le 3$.

$$\begin{aligned}
T_{20}(x) &= (\; S_0(x),\; S_1(x),\; \{\texttt{02}\}\times S_0(x),\; \{\texttt{08}\}\times S_1(x),\\
&\quad \{\texttt{04}\}\times S_0(x),\; \{\texttt{02}\}\times S_1(x),\; \{\texttt{06}\}\times S_0(x),\; \{\texttt{0A}\}\times S_1(x))\\
T_{21}(x) &= (\{\texttt{02}\}\times S_1(x),\; \{\texttt{08}\}\times S_0(x),\; S_1(x),\; S_0(x),\\
&\quad \{\texttt{06}\}\times S_1(x),\; \{\texttt{0A}\}\times S_0(x),\; \{\texttt{04}\}\times S_1(x),\; \{\texttt{02}\}\times S_0(x))\\
T_{22}(x) &= (\{\texttt{04}\}\times S_0(x),\; \{\texttt{02}\}\times S_1(x),\; \{\texttt{06}\}\times S_0(x),\; \{\texttt{0A}\}\times S_1(x),\\
&\quad S_0(x),\; S_1(x),\; \{\texttt{02}\}\times S_0(x),\; \{\texttt{08}\}\times S_1(x))\\
T_{23}(x) &= (\{\texttt{06}\}\times S_1(x),\; \{\texttt{0A}\}\times S_0(x),\; \{\texttt{04}\}\times S_1(x),\; \{\texttt{02}\}\times S_0(x),\\
&\quad \{\texttt{02}\}\times S_1(x),\; \{\texttt{08}\}\times S_0(x),\; S_1(x),\; S_0(x))
\end{aligned}$$

We compute the following equation using mask operations.

$$\begin{aligned}
& (y_0, y_1, y_2, y_3, y_4, y_5, y_6, y_7)\\
=\; & (T_{20}(x_0) \oplus T_{21}(x_1) \oplus T_{22}(x_2) \oplus T_{23}(x_3))\ \&\ \texttt{0xFF00FF00FF00FF00}) \oplus\\
& (T_{20}(x_4) \oplus T_{21}(x_5) \oplus T_{22}(x_6) \oplus T_{23}(x_7))\ \&\ \texttt{0x00FF00FF00FF00FF})
\end{aligned}$$

This implementation requires the following operations.

Size of table (KB):	8
Operation per round (18 rounds in total)	
# of table lookups:	8
# of XORs in F-function:	7
# of XORs out of F-function:	1
# of mask operations (ANDs):	2
# of XORs for round key addition:	1
# of swap (rotation):	1
# of XORs for key whitening:	2

Type 4

The size of table in Type 3 implementation can be reduced by using rotation operations. This implementation is efficient when the processor has only a small primary cache. In the implementation, the tables $T_{22}(x)$ and $T_{23}(x)$ in Type 3 implementation are not required. The outputs of these tables are generated using other tables as follows:

$$\begin{aligned} T_{22}(x) &= T_{20}(x) \lll 32 \\ T_{23}(x) &= T_{21}(x) \lll 32 \end{aligned}$$

This implementation requires the following operations.

Size of table (KB):	4
Operation per round (18 rounds in total)	
# of table lookups:	8
# of XORs in F-function:	7
# of XORs out of F-function:	1
# of mask operations (ANDs):	2
# of rotation:	4
# of XORs for round key addition:	1
# of swap (rotation):	1
# of XORs for key whitening:	2

4.1.3 Decryption

As described in Sections 3 and 4.1 of Algorithm Specification [34], CLEFIA does not have complete involution property. If both of the encryption function and the decryption function can be implemented separately, best performance in speed is expected. If we need to merge them into a single function because of code size limitation or other reasons, we can use the following techniques without large reduction of performance.

- Change address of a look-up table of F-function in even rounds by whether it is used in encryption or decryption step.
- Change round keys in even rounds according to F-function.
- Swap the final output only in decryption step.

4.1.4 Key Scheduling

Key scheduling operation consists of two parts: generating L from a secret key K, and expanding K and L. The former utilizes a round function of encryption, where the same techniques as described in the previous sections are applicable. The latter requires relatively light operations including bit rotation like operations.

Table 5: Results on Software Performance of CLEFIA (assembly language)

	Type of implementation	Key Length	Encryption (cycles/byte)	Decryption (cycles/byte)	Key Setup (cycles)	Table size (KB)
CLEFIA	single-block	128	12.9	13.3	217	8
		192	15.8	16.2	272	
		256	18.3	18.4	328	
	two-block parallel encryption	128	11.1	11.1	217	16
		192	13.3	13.3	272	
		256	15.6	15.6	328	
AES [28]	single-block	128	10.6	N/A	N/A	8

4.2 Performance Results

We present the evaluation results on current software performance of CLEFIA.

Table 5 shows software performance results on Athlon 64 (AMD64) 4000+ 2.4 GHz processor running Windows XP 64-bit Edition. We measured software processing speed of enc/dec and key setup using the `rdtsc` instruction. In the single-block (common) implementation, 12.9 cycles/byte (1.48 Gbps on the processor) is achieved. The two-block parallel implementation [28], which is suitable for CTR mode, achieves 11.1 cycles/byte. Since the total numbers of S-boxes for CLEFIA with 128-bit keys, 144, is less than those for AES with 128-bit keys, 160, we consider that CLEFIA is potentially competitive to AES by reducing data dependency in parallel implementations.

5 Hardware Implementations

This section describes the hardware performance results of CLEFIA. CLEFIA with 128-bit keys achieves about 1.605 Gbps with 5,979 gates using a 0.09 μm CMOS ASIC library. This is in the smallest class as well as the most efficient class among all known 128-bit blockciphers.

5.1 Optimization Techniques in Data Processing Part

We discuss optimization techniques of each component in data processing part including S-boxes S_0, S_1 and diffusion matrices M_0, M_1.

5.1.1 S-box S_0

The 8-bit S-box S_0 consists of three layers: substitution layer 1, linear transformation layer, and substitution layer 2.

Substitution layer 1 can be implemented by parallel location of two 4-bit S-box circuits. Each 4-bit S-box circuit is automatically generated by logic synthesis tool according to each 16 entries $\times$ 4 bits table. Linear transformation layer is a linear $(2, 2)$ multipermutation over $\mathrm{GF}(2^4)$ defined by a primitive polynomial x^4+x+1. This layer requires only 10 XOR (exclusive-OR) logic gates with maximum delay of 2 XOR gates. Substitution layer 2 can be implemented in the same manner with substitution layer 1. Therefore, the total circuit area of S-box S_0 counts the area of four 4-bit S-boxes and 10 XORs; the maximum delay is equivalent to that of two 4-bit S-boxes and 2 XORs.

5.1.2 S-box S_1

The 8-bit S-box S_1 consists of three layers: affine transformation f, inversion over $\mathrm{GF}(2^8)$, and affine transformation g as shown in the left figure of Figure 11. The inversion is performed in $\mathrm{GF}(2^8)$ defined by a primitive polynomial $p(x) = x^8 + x^4 + x^3 + x^2 + 1$. It is well known that significant reduction of gate area is achieved by using inversion over composite fields instead of inversion over $\mathrm{GF}(2^8)$ [32]. For our implementation, we choose the composite field $\mathrm{GF}((2^4)^2)$ defined by the following irreducible polynomials:

$$\left\{\begin{array}{ll} \mathrm{GF}(2^4) & : q_0(x) = x^4 + x + 1 \\ \mathrm{GF}((2^4)^2) & : q_1(x) = x^2 + x + \lambda \quad (\lambda = \omega^3) \end{array}\right. ,$$

where ω is a root of $q_0(x)$. We define the isomorphic mapping ϕ from GF(2^8) to GF($(2^4)^2$) as

$$\begin{aligned}\phi : x_0\alpha^7 + x_1\alpha^6 + x_2\alpha^5 + x_3\alpha^4 + x_4\alpha^3 + x_5\alpha^2 + x_6\alpha + x_7 \\ \mapsto (y_0\omega^3 + y_1\omega^2 + y_2\omega + y_3)\beta + (y_4\omega^3 + y_5\omega^2 + y_6\omega + y_7)\end{aligned}$$

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 0&0&1&0&0&0&0&0 \\ 0&0&1&1&1&0&1&0 \\ 0&1&1&0&0&1&0&0 \\ 1&1&1&1&0&1&0&0 \\ 1&0&1&0&1&1&0&0 \\ 1&0&1&0&1&1&1&0 \\ 0&0&0&0&1&0&1&0 \\ 1&1&1&0&1&1&1&1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix},$$

where α and β are a root of $p(x)$ and $q_1(x)$, respectively. The inverse isomorphic mapping ϕ^{-1} from GF($(2^4)^2$) to GF(2^8) is described as

$$\begin{aligned}\phi^{-1} : (x_0\omega^3 + x_1\omega^2 + x_2\omega + x_3)\beta + (x_4\omega^3 + x_5\omega^2 + x_6\omega + x_7) \\ \mapsto y_0\alpha^7 + y_1\alpha^6 + y_2\alpha^5 + y_3\alpha^4 + y_4\alpha^3 + y_5\alpha^2 + y_6\alpha + y_7\end{aligned}$$

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 1&1&1&1&0&0&1&0 \\ 1&1&0&1&0&1&0&0 \\ 1&0&0&0&0&0&0&0 \\ 1&1&0&0&0&0&1&0 \\ 0&0&0&0&1&1&1&0 \\ 0&1&1&1&0&1&0&0 \\ 0&0&0&0&1&1&0&0 \\ 1&1&0&1&0&0&0&1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix}.$$

Affine transformation f and the isomorphic mapping ϕ is merged to a single matrix operation as

$$\begin{aligned}\phi \circ f : x_0\alpha^7 + x_1\alpha^6 + x_2\alpha^5 + x_3\alpha^4 + x_4\alpha^3 + x_5\alpha^2 + x_6\alpha + x_7 \\ \mapsto (y_0\omega^3 + y_1\omega^2 + y_2\omega + y_3)\beta + (y_4\omega^3 + y_5\omega^2 + y_6\omega + y_7)\end{aligned}$$

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&1&1&0&0 \\ 0&0&0&1&0&0&1&0 \\ 0&0&1&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&1 \\ 1&0&0&1&0&0&0&0 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}.$$

The inverse isomorphic mapping ϕ^{-1} and affine transformation g is also

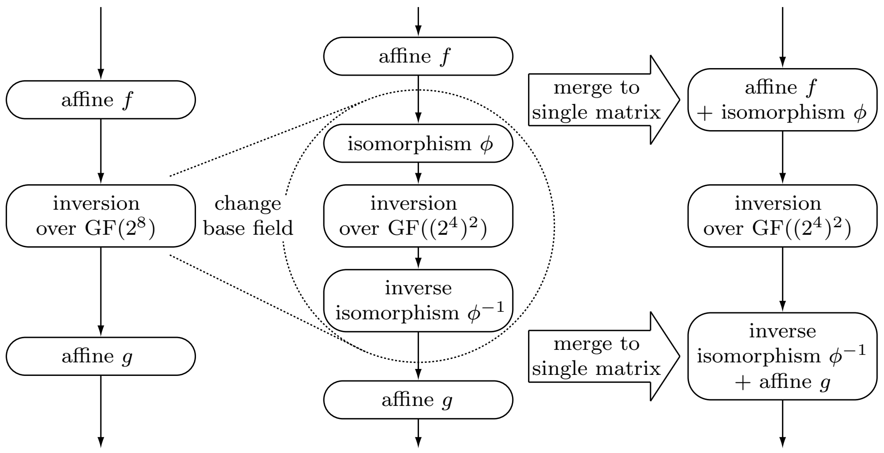
affine f
inversion over GF(2^8)
affine g
change base field
affine f
isomorphism ϕ
inversion over GF($(2^4)^2$)
inverse isomorphism ϕ^{-1}
affine g
merge to single matrix
merge to single matrix
affine f + isomorphism ϕ
inversion over GF($(2^4)^2$)
inverse isomorphism ϕ^{-1} + affine g

Figure 11: Optimized Implementation of S-box S_1

merged to a single matrix operation as

$$g \circ \phi^{-1} : (x_0\omega^3 + x_1\omega^2 + x_2\omega + x_3)\beta + (x_4\omega^3 + x_5\omega^2 + x_6\omega + x_7)$$
$$\mapsto y_0\alpha^7 + y_1\alpha^6 + y_2\alpha^5 + y_3\alpha^4 + y_4\alpha^3 + y_5\alpha^2 + y_6\alpha + y_7$$
$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&1&0&1 \\ 0&0&0&1&1&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&1&0 \\ 0&0&0&0&1&1&0&0 \\ 0&0&1&1&0&0&0&0 \\ 1&0&1&0&0&0&0&0 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

These merged mappings $\phi \circ f$ and $g \circ \phi^{-1}$ require 2 XORs + 2 XNORs + 2 NOTs and 2 XORs + 4 XNORs, respectively.

For an arbitrary element $a_0\beta + a_1$ of GF($(2^4)^2$) where $a_0, a_1 \in$ GF(2^4), the inversion $b_0\beta + b_1 = (a_0\beta + a_1)^{-1}$ ($b_0, b_1 \in$ GF(2^4)) can be computed as follows:

$$\begin{aligned} b_0 &= a_0\Delta^{-1}, \\ b_1 &= (a_0 + a_1)\Delta^{-1}, \\ \Delta &= (a_0 + a_1)a_1 + \lambda a_0^2. \end{aligned}$$

These calculations are performed in GF(2^4). We summarize our optimized implementation of S-box S_1 in the right figure of Figure 11.

5.1.3 Diffusion Matrices M_0 and M_1

Diffusion matrices M_0 and M_1 are multiplied to the outputs of S-boxes as a linear $(4, 4)$ multipermutation over GF(2^8), which is defined by a primitive polynomial $x^8 + x^4 + x^3 + x^2 + 1$.

An addition of two elements in GF(2^8), denoted by $\oplus$, is equivalent to a bitwise XOR operation of their representation as an 8-bit binary string, which costs 8 XOR logic gates. A multiplication in GF(2^8), denoted by $\times$, corresponds to the multiplication of polynomials modulo $x^8+x^4+x^3+x^2+1$. For an element a in GF(2^8), $\{\texttt{02}\} \times a$, $\{\texttt{04}\} \times a$ and $\{\texttt{08}\} \times a$ require 3, 5 and 8 XOR logic gates, respectively.

For an input vector (X_0, X_1, X_2, X_3) and an output vector (Y_0, Y_1, Y_2, Y_3), the multiplication by M_0 is decomposed into three matrices whose non-zero elements are only one of the values $\{\texttt{01}, \texttt{02}, \texttt{04}\}$ shown in the following equations.

$$
\begin{aligned}
\begin{pmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \end{pmatrix}
&= \begin{pmatrix} \texttt{01} & \texttt{02} & \texttt{04} & \texttt{06} \\ \texttt{02} & \texttt{01} & \texttt{06} & \texttt{04} \\ \texttt{04} & \texttt{06} & \texttt{01} & \texttt{02} \\ \texttt{06} & \texttt{04} & \texttt{02} & \texttt{01} \end{pmatrix}
\begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{pmatrix} \\
&= \begin{pmatrix} \texttt{01} & \texttt{00} & \texttt{00} & \texttt{00} \\ \texttt{00} & \texttt{01} & \texttt{00} & \texttt{00} \\ \texttt{00} & \texttt{00} & \texttt{01} & \texttt{00} \\ \texttt{00} & \texttt{00} & \texttt{00} & \texttt{01} \end{pmatrix}
\begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{pmatrix}
+ \begin{pmatrix} \texttt{00} & \texttt{02} & \texttt{00} & \texttt{02} \\ \texttt{02} & \texttt{00} & \texttt{02} & \texttt{00} \\ \texttt{00} & \texttt{02} & \texttt{00} & \texttt{02} \\ \texttt{02} & \texttt{00} & \texttt{02} & \texttt{00} \end{pmatrix}
\begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{pmatrix} \\
&+ \begin{pmatrix} \texttt{00} & \texttt{00} & \texttt{04} & \texttt{04} \\ \texttt{00} & \texttt{00} & \texttt{04} & \texttt{04} \\ \texttt{04} & \texttt{04} & \texttt{00} & \texttt{00} \\ \texttt{04} & \texttt{04} & \texttt{00} & \texttt{00} \end{pmatrix}
\begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{pmatrix}
\end{aligned}
$$

The property of an Hadamard-type matrix M_0 allows intermediate values to be fully shared as follows, and thus contributes to reduction of XOR gates.

$$
\begin{cases} A_0 = X_0 \oplus X_1 \\ A_1 = X_2 \oplus X_3 \end{cases}
\begin{cases} C_0 = \{\texttt{02}\} \times B_0 \\ C_1 = \{\texttt{02}\} \times B_1 \end{cases}
\begin{cases} Y_0 = C_1 \oplus D_1 \oplus X_0 \\ Y_1 = C_0 \oplus D_1 \oplus X_1 \\ Y_2 = C_1 \oplus D_0 \oplus X_2 \\ Y_3 = C_0 \oplus D_0 \oplus X_3 \end{cases}
$$
$$
\begin{cases} B_0 = X_0 \oplus X_2 \\ B_1 = X_1 \oplus X_3 \end{cases}
\begin{cases} D_0 = \{\texttt{04}\} \times A_0 \\ D_1 = \{\texttt{04}\} \times A_1 \end{cases}
$$

The total number and the maximum depth of XOR gates required for multiplication by M_0 are 112 and 4, respectively.

For an input vector (X_0, X_1, X_2, X_3) and an output vector (Z_0, Z_1, Z_2, Z_3), the multiplication by M_1 is decomposed into three matrices whose non-zero elements are only one of the values $\{\texttt{01}, \texttt{02}, \texttt{08}\}$ shown in the following equations.

$$\begin{aligned}
\begin{pmatrix} Z_0 \\ Z_1 \\ Z_2 \\ Z_3 \end{pmatrix} &= \begin{pmatrix} \mathtt{01} & \mathtt{08} & \mathtt{02} & \mathtt{0A} \\ \mathtt{08} & \mathtt{01} & \mathtt{0A} & \mathtt{02} \\ \mathtt{02} & \mathtt{0A} & \mathtt{01} & \mathtt{08} \\ \mathtt{0A} & \mathtt{02} & \mathtt{08} & \mathtt{01} \end{pmatrix} \begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{pmatrix} \\
&= \begin{pmatrix} \mathtt{01} & \mathtt{00} & \mathtt{00} & \mathtt{00} \\ \mathtt{00} & \mathtt{01} & \mathtt{00} & \mathtt{00} \\ \mathtt{00} & \mathtt{00} & \mathtt{01} & \mathtt{00} \\ \mathtt{00} & \mathtt{00} & \mathtt{00} & \mathtt{01} \end{pmatrix} \begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{pmatrix} + \begin{pmatrix} \mathtt{00} & \mathtt{00} & \mathtt{02} & \mathtt{02} \\ \mathtt{00} & \mathtt{00} & \mathtt{02} & \mathtt{02} \\ \mathtt{02} & \mathtt{02} & \mathtt{00} & \mathtt{00} \\ \mathtt{02} & \mathtt{02} & \mathtt{00} & \mathtt{00} \end{pmatrix} \begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{pmatrix} \\
&+ \begin{pmatrix} \mathtt{00} & \mathtt{08} & \mathtt{00} & \mathtt{08} \\ \mathtt{08} & \mathtt{00} & \mathtt{08} & \mathtt{00} \\ \mathtt{00} & \mathtt{08} & \mathtt{00} & \mathtt{08} \\ \mathtt{08} & \mathtt{00} & \mathtt{08} & \mathtt{00} \end{pmatrix} \begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{pmatrix}
\end{aligned}$$

In the same way to the multiplication by M_0, intermediate values can be fully shared as follows:

$$\begin{cases} A_0 = X_0 \oplus X_1 \\ A_1 = X_2 \oplus X_3 \end{cases} \begin{cases} C_0 = \{\mathtt{02}\} \times A_0 \\ C_1 = \{\mathtt{02}\} \times A_1 \end{cases} \begin{cases} Z_0 = C_1 \oplus D_1 \oplus X_0 \\ Z_1 = C_1 \oplus D_0 \oplus X_1 \\ Z_2 = C_0 \oplus D_1 \oplus X_2 \\ Z_3 = C_0 \oplus D_0 \oplus X_3 \end{cases}$$
$$\begin{cases} B_0 = X_0 \oplus X_2 \\ B_1 = X_1 \oplus X_3 \end{cases} \begin{cases} D_0 = \{\mathtt{08}\} \times B_0 \\ D_1 = \{\mathtt{08}\} \times B_1 \end{cases}$$

The number of XOR gates required for multiplication by M_1 counts 118 with these maximum depth being 4.

5.2 Performance Results

We present the evaluation results on current hardware performance of CLEFIA. For CLEFIA with 128-bit key, two types of architecture are implemented: loop architecture and compact architecture. Loop architecture is straightforward hardware implementation taking 1 cycle per round, where both F-functions F_0 and F_1 are implemented in parallel. In compact architecture, F-function F_0 and F_1 are merged F_0/F_1 circuit in order to reduce circuit area. F_0/F_1 circuit is used as F_0 in one cycle and F_1 in another cycle, so that it takes 2 cycles per round. The latency of loop and compact architecture for encryption/decryption is 18 and 36, respectively.

For CLEFIA with 192/256-bit key, only loop architecture is implemented. The latency of encryption/decryption for CLEFIA with 192-bit and 256-bit key is 22 and 26, respectively.

The environment of our hardware design and evaluation is as follows:

Language	Verilog-HDL
Design library	0.09 μm CMOS ASIC library
Simulator	VCS version 2005.06
Logic synthesis	Design Compiler version 2006.06

Table 6: Results on Hardware Performance of CLEFIA

	Key Length	Enc/Dec (cycles)	Key Setup (cycles)	Area (gates)	Freq. (MHz)	Speed (Mbps)	Speed/Area (Kbps/gate)
CLEFIA (0.09μm)	128	18	12	5,979	225.83	1,605.94	268.63
				12,009	422.29	3,003.00	250.06
		36	24	4,950	201.28	715.69	144.59
				9,377	389.55	1,385.10	147.71
	192	22	20	8,536	206.56	1,201.85	140.81
				15,718	391.08	2,275.39	144.76
	256	26	20	8,482	206.56	1,016.95	119.89
				15,542	391.08	1,925.33	123.88
AES [33] (0.13μm)	128	11	N/A	12,454	145.35	1,691.35	135.81
				21,337	224.22	2,609.11	122.28
		54	N/A	5,398	131.24	311.09	57.63
				9,227	220.75	523.26	56.71
Camellia [33] (0.13μm)	128	22	N/A	10,993	166.94	971.29	88.36
				16,905	256.41	1,491.84	88.25
		44	N/A	6,511	111.98	325.76	50.03
				12,231	238.10	692.65	56.63

For each implementation, the above and below columns show the results of the synthesized circuits by area and speed optimization, respectively.

One gate is equivalent to a 2-way NAND and the speed is evaluated under the worst-case conditions.

Table 6 represents the evaluation results. For each implementation, two types of circuit are synthesized by specifying either area or speed optimization. We also show, for comparison, the best known results of hardware performance of AES and Camellia [33]. The synthesized circuit of CLEFIA with 128-bit key in loop architecture occupies only 5,979 gates with efficiency of 268.63 Kbps/gate. Moreover, less than 5 Kgates is achieved for CLEFIA with 128-bit keys in compact architecture. Although we take into account the difference of ASIC libraries, these figures indicate that CLEFIA satisfies both low cost and high efficiency in hardware implementation compared to AES and Camellia.

Acknowledgments

We thank Prof. Lars R. Knudsen, Prof. Bart Preneel, Prof. Vincent Rijmen and Prof. Serge Vaudenay, and some anonymous reviewers for their effort to evaluate the earlier version of CLEFIA. In particular, Prof. Rijmen, Prof. Knudsen and Prof. Preneel pointed out that there exist non-trivial 6-round 32-bit-word-based saturation distinguishers and improved saturation attacks using the distinguishers. The saturation attacks shown in Section 2.10 are based on the improved analytic results in their reports.

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." 2000. Available at http://info.isl.ntt.co.jp/crypt/camellia/dl/support.pdf.

[2] 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." in *Proceedings of Selected Areas in Cryptography – SAC '00* (D. R. Stinson and S. E. Tavares, eds.), no. 2012 in LNCS, pp. 41–54, Springer-Verlag, 2001.

[3] E. Biham, "New types of cryptanalytic attacks using related keys." *Journal of Cryptology*, vol. 7, no. 4, pp. 229–246, 1994.

[4] E. Biham, A. Biryukov, and A. Shamir, "Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials." in *Proceedings of Eurocrypt'99* (J. Stern, ed.), no. 1592 in LNCS, pp. 12–23, Springer-Verlag, 1999.

[5] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems." *Journal of Cryptology*, vol. 4, pp. 3–72, 1991.

[6] E. Biham and A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*. Springer-Verlag, 1993.

[7] A. Biryukov and D. Wagner, "Slide attack." in *Proceedings of Fast Software Encryption – FSE'99* (L. R. Knudsen, ed.), no. 1636 in LNCS, pp. 245–259, Springer-Verlag, 1999.

[8] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions." *Journal of Symbolic Computation*, vol. 9, no. 3, pp. 251–280, 1990.

[9] N. Courtois and J. Pieprzyk, "Cryptanalysis of block ciphers with overdefined systems of equations." in *Proceedings of Asiacrypt'02* (Y. Zheng, ed.), no. 2501 in LNCS, pp. 267–287, Springer-Verlag, 2002.

[10] J. Daemen, L. R. Knudsen, and V. Rijmen, "The block cipher SQUARE." in *Proceedings of Fast Software Encryption – FSE'97* (E. Biham, ed.), no. 1267 in LNCS, pp. 149–165, Springer-Verlag, 1997.

[11] J. Daemen and V. Rijmen, "Statistics of correlation and differentials in block ciphers." in *IACR ePrint archive 2005/212*, 2005.

[12] J. Daemen and V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard (Information Security and Cryptography)*. Springer, 2002.

[13] H. Gilbert, H. Handshuh, A. Joux, and S. Vaudenay, “A stastical attack on RC6.” in *Proceedings of Fast Software Encryption – FSE’00* (B. Schneier, ed.), no. 1978 in LNCS, pp. 64–74, Springer-Verlag, 2001.

[14] H. Gilbert and M. Minier, “A collision attack on 7 rounds of Rijndael.” in *Proceedings of 3rd AES candidate conference*, pp. 230–241, 2001.

[15] S. Hong, S. Lee, J. Lim, J. Sung, D. H. Cheon, and I. Cho, “Provable security against differential and linear cryptanalysis for the SPN structure.” in *Proceedings of Fast Software Encryption – FSE’00* (B. Schneier, ed.), no. 1978 in LNCS, pp. 273–283, Springer-Verlag, 2001.

[16] T. Jakobsen and L. R. Knudsen, “The interpolation attack on block ciphers.” in *Proceedings of Fast Software Encryption – FSE’97* (E. Biham, ed.), no. 1267 in LNCS, pp. 28–40, Springer-Verlag, 1997.

[17] P. Junod and S. Vaudenay, “FOX : A new family of block ciphers.” in *Proceedings of Selected Areas in Cryptography – SAC’04* (H. Handschuh and M. A. Hasan, eds.), no. 3357 in LNCS, pp. 114–129, Springer-Verlag, 2004.

[18] M. Kanda, “Practical security evaluation against differential and linear cryptanalyses for Feistel ciphers with SPN round function.” in *Proceedings of Selected Areas in Cryptography – SAC’00* (D. R. Stinson and S. E. Tavares, eds.), no. 2012 in LNCS, pp. 324–338, Springer-Verlag, 2001.

[19] J. Kelsey, T. Kohno, and B. Schneier, “Amplified boomerang attacks against reduced-round MARS and serpent.” in *Proceedings of Fast Software Encryption – FSE’00* (B. Schneier, ed.), no. 1978 in LNCS, pp. 75–93, Springer-Verlag, 2001.

[20] J. Kelsey, B. Schneier, and D. Wagner, “Related-key cryptanalysis of 3-Way, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA.” in *Proceedings of Information and Communication Security ’97*, no. 1334 in LNCS, pp. 233–246, Springer-Verlag, 1997.

[21] L. R. Knudsen and W. Meier, “Correlations in RC6 with a reduced number of rounds.” in *Proceedings of Fast Software Encryption – FSE’00* (B. Schneier, ed.), no. 1978 in LNCS, pp. 94–108, Springer-Verlag, 2001.

[22] L. R. Knudsen, “Truncated and higher order differentials.” in *Fast Software Encryption: Second International Workshop* (B. Preneel, ed.), no. 1008 in LNCS, pp. 196–211, Springer-Verlag, 1994.

[23] X. Lai, “Higher order derivatives and differential cryptanalysis.” in *Proceedings of symposium on communication, coding and cryptography, in honor of J. L. Massey on the occation of his 60th birthday*, 1994.

[24] S. K. Langford and M. E. Hellman, “Differential-linear cryptanalysis.” in *Proceedings of Crypto’94* (Y. Desmedt, ed.), no. 839 in LNCS, pp. 17–25, Springer-Verlag, 1994.

[25] C. Lim and K. Khoo, “An analysis of XSL applied to BES.” in *Pre-proceedings of Fast Software Encryption – FSE’07* (A. Biryukov, ed.), pp. 253–265, 2007.

[26] M. Matsui, “Linear cryptanalysis of the data encryption standard.” in *Proceedings of Eurocrypt’93* (T. Helleseth, ed.), no. 765 in LNCS, pp. 386–397, Springer-Verlag, 1994.

[27] M. Matsui, “On correlation between the order of s-boxes and the strength of DES.” in *Proceedings of Eurocrypt’94* (A. D. Santis, ed.), no. 950 in LNCS, pp. 366–375, Springer-Verlag, 1995.

[28] M. Matsui, “How far can we go on the x64 processors?.” in *Proceedings of Fast Software Encryption – FSE’06* (M. Robshaw, ed.), no. 4047 in LNCS, pp. 341–358, Springer-Verlag, 2006.

[29] M. Matsui and T. Tokita, “Cryptanalysis of reduced version of the block cipher E2.” in *Proceedings of Fast Software Encryption – FSE’99* (L. R. Knudsen, ed.), no. 1636 in LNCS, pp. 71–80, Springer-Verlag, 1999.

[30] S. Moriai, M. Sugita, K. Aoki, and M. Kanda, “Security of E2 against truncated differential cryptanalysis.” in *Proceedings of Selected Areas in Cryptography – SAC’99* (H. M. Heys and C. M. Adams, eds.), no. 1758 in LNCS, pp. 106–117, Springer-Verlag, 2000.

[31] S. Murphy and M. Robshaw, “Comments on the security of the AES and the XSL technique.” *Electronic Letters*, vol. 39, no. 1, pp. 36–38, 2003.

[32] A. Rudra, P. K. Dubey, C. S. Jutla, V. Kumar, J. R. Rao, and P. Rohatgi, “Efficient Rijndael encryption implementation with composite field arithmetic.” in *Proceedings of Cryptographic Hardware and Embedded Systems - CHES 2001* (Ç. Koç, D. Naccache, and C. Paar, eds.), no. 2162 in LNCS, pp. 171–184, Springer-Verlag, 2001.

[33] A. Satoh and S. Morioka, “Hardware-focused performance comparison for the standard block ciphers AES,Camellia, and Triple-DES.” in *Proceedings of ISC 2003* (C. Boyd and W. Mao, eds.), no. 2851 in LNCS, pp. 252–266, Springer-Verlag, 2003.

[34] "The 128-bit blockcipher CLEFIA : Algorithm specification." On-line document, 2007. Sony Corporation.

[35] "The 128-bit blockcipher CLEFIA : Design rationale." On-line document, 2007. Sony Corporation.

[36] M. Sugita, K. Kobara, and H. Imai, "Security of reduced version of the block cipher Camellia against truncated and impossible differential cryptanalysis." in *Proceedings of Asiacrypt'01* (C. Boyd, ed.), no. 2248 in LNCS, pp. 193–207, Springer-Verlag, 2001.

[37] S. Vaudenay, "An experimental on DES statistical cryptanalysis." in *3rd ACM conference on computer and communications security*, pp. 139–147, ACM Press, 1996.

[38] D. Wagner, "The boomerang attack." in *Proceedings of Fast Software Encryption – FSE'99* (L. R. Knudsen, ed.), no. 1636 in LNCS, pp. 156–170, Springer-Verlag, 1999.

[39] H. Wu, "Related-cipher attacks." in *Proceedings of Information and Communications Security – ICICS 2002* (R. H. Deng, S. Qing, F. Bao, and J. Zhou, eds.), no. 2513 in LNCS, pp. 447–455, Springer-Verlag, 2002.