The 128-Bit Blockcipher CLEFIA (Extended Abstract)

Taizo Shirai[1], Kyoji Shibutani[1], Toru Akishita[1], Shiho Moriai[1], and Tetsu Iwata[2]

[1] Sony Corporation
1-7-1 Konan, Minato-ku, Tokyo 108-0075, Japan
{taizo.shirai,kyoji.shibutani,toru.akishita,shiho.moriai}@jp.sony.com
[2] Nagoya University
Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan
iwata@cse.nagoya-u.ac.jp

Abstract. We propose a new 128-bit blockcipher CLEFIA supporting key lengths of 128, 192 and 256 bits, which is compatible with AES. CLEFIA achieves enough immunity against known attacks and flexibility for efficient implementation in both hardware and software by adopting several novel and state-of-the-art design techniques. CLEFIA achieves a good performance profile both in hardware and software. In hardware using a 0.09 μm CMOS ASIC library, about 1.60 Gbps with less than 6 Kgates, and in software, about 13 cycles/byte, 1.48 Gbps on 2.4 GHz AMD Athlon 64 is achieved. CLEFIA is a highly efficient blockcipher, especially in hardware.

Keywords: blockcipher, generalized Feistel structure, DSM, CLEFIA.

1 Introduction

A lot of secure and high performance blockciphers have been designed benefited from advancing research which started since the development of DES [11]. For example, IDEA, MISTY1, AES, and Camellia are fruits of such research activities [1, 10, 15, 19]. New design and evaluation techniques are evolved day by day; topics on algebraic immunity and related-key attacks are paid attention recently [8, 6]. Moreover, light-weight ciphers suitable for a very limited resource environment are still active research fields. FOX and HIGHT are examples of such newly developed blockciphers [12, 13].

We think it is good timing to show a new blockcipher design based on current state-of-the-art techniques. In this paper, we propose a new 128-bit blockcipher CLEFIA supporting key lengths of 128, 192 and 256 bits, which are compatible with AES. CLEFIA achieves enough immunity against known cryptanalyses and flexibility for very efficient implementation in hardware and software. The fundamental structure of CLEFIA is a generalized Feistel structure consisting of 4 data lines, in which there are two 32-bit F-functions per one round.

A. Biryukov (Ed.): FSE 2007, LNCS 4593, pp. 181–195, 2007.

One of novel design approaches of CLEFIA is that these F-functions employ the Diffusion Switching Mechanism (DSM) [22, 23]: they use different diffusion matrices, and two different S-boxes are used to obtain stronger immunity against a certain class of attacks. Consequently, the required number of rounds can be reduced. Moreover, the two S-boxes are based on different algebraic structures, which is expected to increase algebraic immunity. Other novel ideas include the secure and compact key scheduling design and the *DoubleSwap* function used in it. The key scheduling part uses a generalized Feistel structure, and it is possible to share it with the data processing part. The *DoubleSwap* function can be compactly implemented to enable efficient round key generation in encryption and decryption.

CLEFIA achieves about 1.60 Gbps with less than 6 Kgates in hardware using a 0.09 μm CMOS ASIC library, and about 13 cycles/byte, 1.48 Gbps on 2.4 GHz AMD Athlon 64 processor in software. We consider CLEFIA is a well-balanced blockcipher in security and performance, and the performance is advantageous among other blockciphers especially in hardware.

This paper is organized as follows: in Sect. 2, notations are first introduced. In Sect. 3, we give the specification of CLEFIA. Then design rationale is shown in Sect. 4. Sect. 5 describes the evaluation results on both of security and performance aspects. Finally Sect. 6 concludes the paper.

2 Notations

This section describes mathematical notations, conventions and symbols used throughout this paper.

`0x`	: A prefix for a binary string in a hexadecimal form
$a_{(b)}$	: b denotes the bit length of a
$a\|b$ or $(a\|b)$	: Concatenation
$(a,\ b)$ or $(a\ b)$	: Vector style representation of $a\|b$
$a \leftarrow b$	: Updating a value of a by a value of b
${}^t a$	: Transposition of a vector or a matrix a
$a \oplus b$	: Bitwise exclusive-OR. Addition in GF(2^n)
$a \cdot b$	: Multiplication in GF(2^n)
$\overline{a}$	: Logical negation
$a \lll b$	: b-bit left cyclic shift operation
$\mathbf{w_b}(a)$	: For an $8n$-bit string $a = a_0\|a_1\|\ldots\|a_{n-1}$, $a_i \in \{0,1\}^8$, $\mathbf{w_b}(a)$ denotes the number of non-zero a_is.

3 Specification

This section describes the specification of CLEFIA. We first define a function $GFN_{d,r}$ which is a fundamental structure for CLEFIA, followed by definitions of a data processing part and a key scheduling part.

3.1 Definition of $GFN_{d,r}$

CLEFIA uses a 4-branch and an 8-branch Type-2 generalized Feistel network [24]. We denote d-branch r-round generalized Feistel network employed in CLEFIAas $GFN_{d,r}$. $GFN_{d,r}$ employs two different 32-bit F-functions F_0 and F_1 whose input/output are defined as follows.

$$F_0, F_1 : \begin{cases} \{0,1\}^{32} \times \{0,1\}^{32} \rightarrow \{0,1\}^{32} \\ \quad (RK_{(32)}, x_{(32)}) \mapsto y_{(32)} \end{cases}$$

For d 32-bit input X_i and output Y_i ($0 \leq i < d$), and $dr/2$ 32-bit round keys RK_i ($0 \leq i < dr/2$), $GFN_{d,r}$ ($d = 4, 8$) are defined as follows.

$$GFN_{4,r} : \begin{cases} \{\{0,1\}^{32}\}^{2r} \times \{\{0,1\}^{32}\}^4 \rightarrow \{\{0,1\}^{32}\}^4 \\ (RK_{0(32)}, \ldots, RK_{2r-1(32)}, X_{0(32)}, \ldots, X_{3(32)}) \mapsto Y_{0(32)}, \ldots, Y_{3(32)} \end{cases}$$

Step 1. $T_0 \mid T_1 \mid T_2 \mid T_3 \leftarrow X_0 \mid X_1 \mid X_2 \mid X_3$
Step 2. For $i = 0$ to $r-1$ do the following:
  Step 2.1 $T_1 \leftarrow T_1 \oplus F_0(RK_{2i}, T_0)$, $T_3 \leftarrow T_3 \oplus F_1(RK_{2i+1}, T_2)$
  Step 2.2 $T_0 \mid T_1 \mid T_2 \mid T_3 \leftarrow T_1 \mid T_2 \mid T_3 \mid T_0$
Step 3. $Y_0 \mid Y_1 \mid Y_2 \mid Y_3 \leftarrow T_3 \mid T_0 \mid T_1 \mid T_2$

$$GFN_{8,r} : \begin{cases} \{\{0,1\}^{32}\}^{4r} \times \{\{0,1\}^{32}\}^8 \rightarrow \{\{0,1\}^{32}\}^8 \\ (RK_{0(32)}, \ldots, RK_{4r-1(32)}, X_{0(32)}, \ldots, X_{7(32)}) \mapsto Y_{0(32)}, \ldots, Y_{7(32)} \end{cases}$$

Step 1. $T_0 \mid T_1 \mid \ldots \mid T_7 \leftarrow X_0 \mid X_1 \mid \ldots \mid X_7$
Step 2. For $i = 0$ to $r-1$ do the following:
  Step 2.1 $T_1 \leftarrow T_1 \oplus F_0(RK_{4i}, T_0)$, $T_3 \leftarrow T_3 \oplus F_1(RK_{4i+1}, T_2)$
  $T_5 \leftarrow T_5 \oplus F_0(RK_{4i+2}, T_4)$, $T_7 \leftarrow T_7 \oplus F_1(RK_{4i+3}, T_6)$
  Step 2.2 $T_0 \mid T_1 \mid \ldots \mid T_6 \mid T_7 \leftarrow T_1 \mid T_2 \mid \ldots \mid T_7 \mid T_0$
Step 3. $Y_0 \mid Y_1 \mid \ldots \mid Y_6 \mid Y_7 \leftarrow T_7 \mid T_0 \mid \ldots \mid T_5 \mid T_6$

The inverse function $GFN_{d,r}^{-1}$ are realized by changing the order of RK_i and the direction of word rotation at Step 2.2 and Step 3.

3.2 Data Processing Part

The data processing part of CLEFIA consists of ENC_r for encryption and DEC_r for decryption. ENC_r and DEC_r use a 4-branch generalized Feistel structure $GFN_{4,r}$. Let $P, C \in \{0,1\}^{128}$ be a plaintext and a ciphertext, and let $P_i, C_i \in \{0,1\}^{32}$ ($0 \leq i < 4$) be divided plaintext and ciphertext where $P = P_0|P_1|P_2|P_3$ and $C = C_0|C_1|C_2|C_3$, and let $WK_0, WK_1, WK_2, WK_3 \in \{0,1\}^{32}$ be whitening keys and $RK_i \in \{0,1\}^{32}$ ($0 \leq i < 2r$) be round keys provided by the key scheduling part. Then, r-round encryption function ENC_r is defined as follows:

$$ENC_r : \begin{cases} \{\{0,1\}^{32}\}^4 \times \{\{0,1\}^{32}\}^{2r} \times \{\{0,1\}^{32}\}^4 \rightarrow \{\{0,1\}^{32}\}^4 \\ (WK_{0(32)}, \ldots, WK_{3(32)}, RK_{0(32)}, \ldots, RK_{2r-1(32)}, P_{0(32)}, \ldots, P_{3(32)}) \\ \qquad \mapsto C_{0(32)}, \ldots, C_{3(32)} \end{cases}$$

Step 1. $T_0 \mid T_1 \mid T_2 \mid T_3 \leftarrow P_0 \mid (P_1 \oplus WK_0) \mid P_2 \mid (P_3 \oplus WK_1)$
Step 2. $T_0 \mid T_1 \mid T_2 \mid T_3 \leftarrow GFN_{4,r}(RK_0, \ldots, RK_{2r-1}, T_0, T_1, T_2, T_3)$
Step 3. $C_0 \mid C_1 \mid C_2 \mid C_3 \leftarrow T_0 \mid (T_1 \oplus WK_2) \mid T_2 \mid (T_3 \oplus WK_3)$

The decryption function DEC_r is the inverse function of ENC_r which is defined by using $GFN^{-1}_{d,r}$. Fig. 1 in Appendix C illustrates the ENC_r function. The number of rounds, r, is 18, 22 and 26 for 128-bit, 192-bit and 256-bit keys, respectively.

3.3 Key Scheduling Part

The key scheduling part of CLEFIA supports 128, 192 and 256-bit keys and outputs whitening keys WK_i $(0 \leq i < 4)$ and round keys RK_j $(0 \leq j < 2r)$ for the data processing part. We first define the *DoubleSwap* function which is used in the key scheduling part.

Definition 1 (The *DoubleSwap* Function Σ).
The DoubleSwap function $\Sigma : \{0,1\}^{128} \rightarrow \{0,1\}^{128}$ *is defined as follows:*

$$X_{(128)} \mapsto Y_{(128)}$$
$$Y = X[7-63] \mid X[121-127] \mid X[0-6] \mid X[64-120] ,$$

where $X[a-b]$ *denotes a bit string cut from the a-th bit to the b-th bit of* X. *0-th bit is the most significant bit (See Fig. 2 in Appendix C).*

Let K be a k-bit key, where k is 128, 192 or 256. The key scheduling part is divided into the following two sub-parts. (1) Generating an intermediate key L from K, and (2) Expanding K and L to generate WK_i and RK_j. The key scheduling is explained according to the sub-parts.

Key Scheduling for a 128-bit Key. The 128-bit intermediate key L is generated by applying $GFN_{4,12}$ which takes twenty-four 32-bit constant values $CON^{(128)}_i$ $(0 \leq i < 24)$ as round keys and $K = K_0|K_1|K_2|K_3$ as an input. Then K and L are used to generate WK_i $(0 \leq i < 4)$ and RK_j $(0 \leq j < 36)$ in the following steps. In the latter part, thirty-six 32-bit constant values $CON^{(128)}_i$ $(24 \leq i < 60)$ are used. The generation steps of CON are explained in Sect 3.5.

(Generating L from K)
Step 1. $L \leftarrow GFN_{4,12}(CON^{(128)}_0, \ldots, CON^{(128)}_{23}, K_0, \ldots, K_3)$

(Expanding K and L)
Step 2. $WK_0|WK_1|WK_2|WK_3 \leftarrow K$
Step 3. For $i = 0$ to 8 do the following:
  $T \leftarrow L \oplus (CON^{(128)}_{24+4i} \mid CON^{(128)}_{24+4i+1} \mid CON^{(128)}_{24+4i+2} \mid CON^{(128)}_{24+4i+3})$
  $L \leftarrow \Sigma(L)$
  if i is odd: $T \leftarrow T \oplus K$
  $RK_{4i}|RK_{4i+1}|RK_{4i+2}|RK_{4i+3} \leftarrow T$

Key Scheduling for a 192-bit Key. Two 128-bit values K_L, K_R are generated from a 192-bit key $K = K_0|K_1|K_2|K_3|K_4|K_5,\ K_i \in \{0,1\}^{32}$. Then two 128-bit values L_L, L_R are generated by applying $GFN_{8,10}$ which takes $CON_i^{(192)}$ $(0 \le i < 40)$ as round keys and $K_L|K_R$ as a 256-bit input. Then K_L, K_R and L_L, L_R are used to generate WK_i $(0 \le i < 4)$ and RK_j $(0 \le j < 44)$ in the following steps. In the latter part, forty-four 32-bit constant values $CON_i^{(192)}$ $(40 \le i < 84)$ are used.

The following steps show the 192-bit/256-bit key scheduling. For the 192-bit key scheduling, the value of k is set as 192.

(Generating L_L, L_R from K_L, K_R for a k-bit key)
Step 1. Set $k = 192$ or $k = 256$
Step 2. If $k = 192$: $K_L \leftarrow K_0|K_1|K_2|K_3,\ K_R \leftarrow K_4|K_5|\overline{K_0}|\overline{K_1}$
else if $k = 256$: $K_L \leftarrow K_0|K_1|K_2|K_3,\ K_R \leftarrow K_4|K_5|K_6|K_7$
Step 3. Let $K_L = K_{L0}|K_{L1}|K_{L2}|K_{L3},\quad K_R = K_{R0}|K_{R1}|K_{R2}|K_{R3}$
$L_L|L_R \leftarrow GFN_{8,10}(CON_0^{(k)}, \ldots, CON_{39}^{(k)}, K_{L0}, \ldots, K_{L3}, K_{R0}, \ldots, K_{R3})$

(Expanding K_L, K_R and L_L, L_R for a k-bit key)
Step 4. $WK_0|WK_1|WK_2|WK_3 \leftarrow K_L \oplus K_R$
Step 5. For $i = 0$ to 10 (if $k = 192$), or 12 (if $k = 256$) do the following:
If (i mod 4) = 0 or 1:
$T \leftarrow L_L \oplus (CON_{40+4i}^{(k)} \mid CON_{40+4i+1}^{(k)} \mid CON_{40+4i+2}^{(k)} \mid CON_{40+4i+3}^{(k)})$
$L_L \leftarrow \Sigma(L_L)$
if i is odd: $T \leftarrow T \oplus K_R$
else:
$T \leftarrow L_R \oplus (CON_{40+4i}^{(k)} \mid CON_{40+4i+1}^{(k)} \mid CON_{40+4i+2}^{(k)} \mid CON_{40+4i+3}^{(k)})$
$L_R \leftarrow \Sigma(L_R)$
if i is odd: $T \leftarrow T \oplus K_L$
$RK_{4i}|RK_{4i+1}|RK_{4i+2}|RK_{4i+3} \leftarrow T$

Key Scheduling for a 256-bit Key. For a 256-bit key, the value of k is set as 256, and the steps are almost the same as in the 192-bit key case. The difference is that we use $CON_i^{(256)}$ $(0 \le i < 40)$ as round keys to generate L_L and L_R, and then to generate RK_j $(0 \le j < 52)$, we use fifty-two 32-bit constant values $CON_i^{(256)}$ $(40 \le i < 92)$.

3.4 F-Functions

F-functions $F_0, F_1 : (RK_{(32)}, x_{(32)}) \mapsto y_{(32)}$ are defined as follows:

[F-function F_0]	[F-function F_1]
Step 1. $T \leftarrow RK \oplus x$	*Step 1.* $T \leftarrow RK \oplus x$
Step 2. Let $T = T_0\|T_1\|T_2\|T_3,\ T_i \in \{0,1\}^8$	*Step 2.* Let $T = T_0\|T_1\|T_2\|T_3,\ T_i \in \{0,1\}^8$
$T_0 \leftarrow S_0(T_0),\ T_1 \leftarrow S_1(T_1)$	$T_0 \leftarrow S_1(T_0),\ T_1 \leftarrow S_0(T_1)$
$T_2 \leftarrow S_0(T_2),\ T_3 \leftarrow S_1(T_3)$	$T_2 \leftarrow S_1(T_2),\ T_3 \leftarrow S_0(T_3)$
Step 3. Let $y = y_0\|y_1\|y_2\|y_3,\ y_i \in \{0,1\}^8$	*Step 3.* Let $y = y_0\|y_1\|y_2\|y_3,\ y_i \in \{0,1\}^8$
${}^t(y_0, y_1, y_2, y_3) = M_0\ {}^t(T_0, T_1, T_2, T_3)$	${}^t(y_0, y_1, y_2, y_3) = M_1\ {}^t(T_0, T_1, T_2, T_3)$

S_0 and S_1 are nonlinear 8-bit S-boxes. The orders of these S-boxes are different in F_0 and F_1. Tables in Appendix B show the input/output values of each S-box. In these tables all values are expressed in hexadecimal form, suffixes '0x' are omitted. For an 8-bit input of an S-box, the upper 4-bit indicates a row and the lower 4-bit indicates a column .

Two matrices M_0 and M_1 in Step 3 are defined as follows.

$$M_0 = \begin{pmatrix} \texttt{0x01} & \texttt{0x02} & \texttt{0x04} & \texttt{0x06} \\ \texttt{0x02} & \texttt{0x01} & \texttt{0x06} & \texttt{0x04} \\ \texttt{0x04} & \texttt{0x06} & \texttt{0x01} & \texttt{0x02} \\ \texttt{0x06} & \texttt{0x04} & \texttt{0x02} & \texttt{0x01} \end{pmatrix}, \qquad M_1 = \begin{pmatrix} \texttt{0x01} & \texttt{0x08} & \texttt{0x02} & \texttt{0x0a} \\ \texttt{0x08} & \texttt{0x01} & \texttt{0x0a} & \texttt{0x02} \\ \texttt{0x02} & \texttt{0x0a} & \texttt{0x01} & \texttt{0x08} \\ \texttt{0x0a} & \texttt{0x02} & \texttt{0x08} & \texttt{0x01} \end{pmatrix}.$$

These are 4×4 Hadamard-type matrices with elements $h_{ij} = a_{i \oplus j}$ for a certain set $\{a_0, a_1, a_2, a_3\}$[1].

The multiplications between matrices and vectors are performed in GF(2^8) defined by the lexicographically first primitive polynomial $z^8 + z^4 + z^3 + z^2 + 1$. Fig. 3 in Appendix C illustrates the construction of F_0 and F_1.

3.5 Constant Values

32-bit constant values $CON_i^{(k)}$ are used in the key scheduling algorithm. We need 60, 84 and 92 constant values for 128, 192 and 256-bit keys, respectively. Let $\mathbf{P}_{(16)} = \texttt{0xb7e1}$ $(= (e-2) \cdot 2^{16})$ and $\mathbf{Q}_{(16)} = \texttt{0x243f}$ $(= (\pi - 3) \cdot 2^{16})$, where e is the base of the natural logarithm (2.71828...) and π is the circle ratio (3.14159...). $CON_i^{(k)}$, for $k = 128, 192, 256$, are generated by the following way (See Table 1 for the repetition numbers $l^{(k)}$ and the initial values $IV^{(k)}$).

Step 1. $T \leftarrow IV^{(k)}$
Step 2. For $i = 0$ to $l^{(k)} - 1$ do the following:
  Step 2.1. $CON_{2i}^{(k)} \leftarrow (T \oplus \mathbf{P}) \mid (\overline{T} \lll 1)$
  Step 2.2. $CON_{2i+1}^{(k)} \leftarrow (\overline{T} \oplus \mathbf{Q}) \mid (T \lll 8)$
  Step 2.3. $T \leftarrow T \cdot \texttt{0x0002}^{-1}$

In Step 2.3, multiplications are performed in GF(2^{16}) defined by a primitive polynomial $z^{16} + z^{15} + z^{13} + z^{11} + z^5 + z^4 + 1$ (=0x1a831)[2].

Table 1. Required Numbers of Constant Values

k	# of CON	$l^{(k)}$	$IV^{(k)}$	
128	60	30	0x428a	$(= (\sqrt[3]{2} - 1) \cdot 2^{16})$
192	84	42	0x7137	$(= (\sqrt[3]{3} - 1) \cdot 2^{16})$
256	92	46	0xb5c0	$(= (\sqrt[3]{5} - 1) \cdot 2^{16})$

[1] An Hadamard-type matrix is used in the blockcipher Anubis [2].

[2] The lower 16-bit value is defined as 0xa831=$(\sqrt[3]{101} - 4) \cdot 2^{16}$. '101' is the smallest prime number satisfying the primitive polynomial condition in this form.

4 Design Rationale

CLEFIA is designed to realize good balance on three fundamental directions for practical ciphers: (1) security, (2) speed, and (3) cost for implementations. This section describes the design rationale of several aspects of CLEFIA.

Structure. CLEFIA employs a 4-branch generalized Feistel structure which is considered as an extension of a 2-branch traditional Feistel structure. There are many types of generalized Feistel structures. We choose one instance which is known as "Generalized type-2 transformations" defined by Zheng *et al.* [24]. The type-2 structure has two F-functions in one round for the four data lines case. The type-2 structure has the following features:

- F-functions are smaller than that of the traditional Feistel structure
- Plural F-functions can be processed simultaneously
- Tends to require more rounds than the traditional Feistel structure

The first feature is a great advantage for software and hardware implementations, and the second one is suitable for efficient implementation especially in hardware. We conclude that the advantages of the type-2 structure surpass the disadvantage of the third one for our blockcipher design. Moreover, the new design technique, which is explained in the next, enables to reduce the number of rounds effectively.

Diffusion Switching Mechanism. CLEFIA employs two different diffusion matrices to enhance the immunity against differential attacks and linear attacks by using the Diffusion Switching Mechanism (DSM). This design technique was originally developed for the traditional Feistel structures [22,23]. We customized this technique suitable for $GFN_{d,r}$, which is one of the unique propositions of this cipher. Due to the DSM, we can prevent difference cancellations and linear mask cancellations in the neighborhood rounds in the cipher. As a result the guaranteed number of active S-boxes is increased.

Let the branch number of a function P be $\mathcal{B}(P) = \min_{a \neq 0}\{\mathbf{w_b}(a) + \mathbf{w_b}(P(a))\}$. The two matrices M_0 and M_1 used in CLEFIA satisfy the following branch number conditions of the DSM.

$$\mathcal{B}(M_0) = \mathcal{B}(M_1) = 5, \quad \mathcal{B}(M_0|M_1) = \mathcal{B}({}^tM_0^{-1}|{}^tM_1^{-1}) = 5 \ .$$

Table 2 shows the guaranteed numbers of active S-boxes of CLEFIA. The guaranteed data are obtained from computer simulations using a exhaustive-type search algorithm. Now we focus on the columns indexed by '$GFN_{4,r}$'. The columns of 'D' and 'L' in the table show the guaranteed number of differential and linear active S-boxes, respectively. The 'DSM' denotes that the DSM is used, and the 'w/o DSM' denotes that DSM is not used, where only one matrix with branch number 5 is employed. From this table we can confirm the effects of the DSM when $r \geq 3$, and these guaranteed numbers increase about $20\% - 40\%$ than the structure without DSM. Consequently, the numbers of rounds can be reduced, which implies that the performance is improved.

Table 2. Guaranteed Numbers of Active S-boxes

	$GFN_{4,r}$			$GFN_{8,r}$
r	D & L w/o DSM	D DSM	L DSM	D DSM
1	0	0	0	0
2	1	1	1	1
3	2	2	5	2
4	6	6	6	6
5	8	8	10	8
6	12	12	15	12
7	12	14	16	14
8	13	18	18	21
9	14	20	20	24
10	18	22	23	29
11	20	24	26	34
12	24	28	30	39
13	24	30	32	44
14	25	34	34	48
15	26	36	36	50
16	30	38	39	53
17	32	40	42	56
18	36	44	46	59
19	36	46	48	62
20	37	50	50	66
21	38	52	52	71
22	42	55	55	76
23	44	56	58	81
24	48	59	62	86
25	48	62	64	91
26	49	65	66	94

Table 3. Tables of SS_i $(0 \leq i < 4)$

x	0 1 2 3 4 5 6 7 8 9 a b c d e f	x	0 1 2 3 4 5 6 7 8 9 a b c d e f
$SS_0(x)$	e 6 c a 8 7 2 f b 1 4 0 5 9 d 3	$SS_1(x)$	6 4 0 d 2 b a 3 9 c e f 8 7 5 1
$SS_2(x)$	b 8 5 e a 6 4 c f 7 2 3 1 0 d 9	$SS_3(x)$	a 2 6 d 3 4 5 e 0 7 8 9 b f c 1

Choices of two S-boxes. CLEFIA employs two different types of 8-bit S-boxes: one is based on four 4-bit random S-boxes, and the other is based on the inverse function over GF(2^8) which has the best possible maximum differential probability DP_{max} and linear probability LP_{max}. The both S-boxes are selected to be implemented efficiently especially in hardware. The two 8-bit S-boxes S_0 and S_1 are defined as:

$$S_0, S_1 : \begin{cases} \{0,1\}^8 \rightarrow \{0,1\}^8 \\ \quad x_{(8)} \mapsto y_{(8)} \end{cases}$$

S_0 is generated by combining four 4-bit S-boxes SS_0, SS_1, SS_2 and SS_3 in the following way. The values of these S-boxes are defined as Table 3.

Step 1. $t_0 \leftarrow SS_0(x_0), \quad t_1 \leftarrow SS_1(x_1)$, where $x = x_0 \vert x_1,\ x_i \in \{0,1\}^4$
Step 2. $u_0 \leftarrow t_0 \oplus \texttt{0x2} \cdot t_1, \quad u_1 \leftarrow \texttt{0x2} \cdot t_0 \oplus t_1$
Step 3. $y_0 \leftarrow SS_2(u_0), \quad y_1 \leftarrow SS_3(u_1)$, where $y = y_0 \vert y_1,\ y_i \in \{0,1\}^4$

The multiplication in $\texttt{0x2} \cdot t_i$ is performed in GF(2^4) defined by the lexicographically first primitive polynomial $z^4 + z + 1$.

S_1 is defined as follows:

$$y = \begin{cases} g(f(x)^{-1}) & \text{if} \quad f(x) \neq 0 \\ g(0) & \text{if} \quad f(x) = 0 \end{cases} .$$

The inverse function is performed in GF(2^8) defined by a primitive polynomial $z^8 + z^4 + z^3 + z^2 + 1$. $f(\cdot)$ and $g(\cdot)$ are affine transformations over GF(2), which are defined as follows.

$f : x_{(8)} \mapsto y_{(8)}$	$g : x_{(8)} \mapsto y_{(8)}$
$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 0&0&0&1&1&0&0&0 \\ 0&1&0&1&0&0&0&1 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&1&1&0 \\ 0&1&1&0&0&1&0&1 \\ 0&1&0&1&1&1&0&0 \\ 0&1&1&0&0&0&0&0 \\ 1&0&0&0&0&0&0&1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 0&0&0&0&1&0&1&0 \\ 0&1&0&0&0&0&0&1 \\ 0&1&0&1&1&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&1&1&0&0&0&0 \\ 0&0&0&0&0&0&1&0 \\ 1&0&0&1&0&0&0&0 \\ 0&1&0&0&0&1&0&0 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$

Here, $x = x_0|x_1|x_2|x_3|x_4|x_5|x_6|x_7$ and $y = y_0|y_1|y_2|y_3|y_4|y_5|y_6|y_7$, $x_i, y_i \in \{0, 1\}$. The constants in f and g can be represented as `0x1e` and `0x69`, respectively. If we apply isomorphic mapping ϕ from GF(2^8) to GF($(2^4)^2$) defined by an irreducible polynomial $z^2 + z + \omega^3$, where ω is a root of $z^4 + z + 1 = 0$, the merged transformations $\phi \circ f$ and $g \circ \phi^{-1}$ require only few XOR operations.

For security parameters, DP_{max} of S_0 is $2^{-4.67}$ and its LP_{max} is $2^{-4.38}$, the minimum Boolean degree is 6, and the minimum number of terms over GF(2^8) is 244. For S_1, DP_{max} and LP_{max} are both $2^{-6.00}$, the minimum Boolean degree is 7 and it has at least 252 terms over GF(2^8).

Designs for Efficient Implementations. CLEFIA can be implemented efficiently both in hardware and software. In Table 4, we summarize the design aspects for efficient implementations.

Table 4. Design Aspects for Efficient Implementations

GFN	· Small size F-functions (32-bit in/out) · No need for the inverse F-functions
SP-type F-function	· Enabling the fast table implementation in software
DSM	· Reducing the numbers of rounds
S-boxes	· Very small footprint of S_0 and S_1 in hardware
Matrices	· Using elements with low hamming weights only
Key Schedule	· Sharing the structure with the data processing part · Requiring only a 128-bit register for a 128-bit key · Small footprint of *DoubleSwap*

5 Evaluations

5.1 Security

As a result of our security evaluation, full-round CLEFIA is considered as a secure blockcipher against known attacks. Here, we mention the cryptanalytic results of several attacks which are considered effective for reduced-round CLEFIA.

Differential Cryptanalysis [7]. For differential attack, we adopt an approach to count the number of active S-boxes of differential characteristics. This method was adopted by AES, Camellia and other blockciphers [10, 1]. We found the guaranteed number of differential active S-boxes of CLEFIA by computer search as shown in Table 2. Using 28 active S-boxes for 12-round CLEFIA and $DP^{S_0}_{max} = 2^{-4.67}$, it is shown that $DCP^{12r}_{max} \leq 2^{28\times(-4.67)} = 2^{-130.76}$. This means there is no useful 12-round differential characteristic for an attacker. Moreover, since S_1 has lower DP_{max}, the actual upper-bound of DCP is expected to be lower than the above estimation. As a result, we believe that full-round CLEFIA is secure against differential cryptanalysis.

Linear Cryptanalysis [17]. We also apply active S-boxes based approach for the evaluation of linear cryptanalysis. Since $LP^{S_0}_{max} = 2^{-4.38}$, combining 30 active S-boxes for 12-round CLEFIA, $LCP^{12r}_{max} \leq 2^{30\times(-4.38)} = 2^{-131.40}$. We conclude that it is difficult for an attacker to find 12-round linear-hulls which can be used to distinguish CLEFIA from a random permutation. As a result, full-round CLEFIA is secure enough against linear cryptanalysis.

Impossible Differential Cryptanalysis [4]. We consider that impossible differential attack is one of the most powerful attacks against CLEFIA. The following two impossible differential paths are found.

$$(0,\alpha,0,0) \overset{9r}{\not\rightarrow} (0,\alpha,0,0) \text{ and } (0,0,0,\alpha) \overset{9r}{\not\rightarrow} (0,0,0,\alpha) \quad p = 1$$

where $\alpha \in \{0,1\}^{32}$ is any non-zero difference. These paths are confirmed by the check algorithm proposed by Kim *et al.* [14]. Using the above distinguisher, we can mount actual key-recovery attacks for each key length. Table 5 shows the summary of the complexity required for the impossible differential attacks. According to Table 5, it is expected that full-round CLEFIA has enough security margin against this attack.

Table 5. Summary of Impossible Differential Cryptanalysis

# of rounds	key length	key whitening	# of chosen plaintexts	time complexity
10	128, 192, 256	yes	$2^{101.7}$	2^{102}
11	192, 256	yes	$2^{103.5}$	2^{188}
12	256	no	$2^{103.8}$	2^{252}

Saturation Cryptanalysis [9]. We also consider that saturation attack is one of the most powerful attacks against CLEFIA. In this analysis, we consider a 32-bit word based saturation attack. Let $X_i \in \{0,1\}^{32}$ $(0 \leq i < 2^{32})$ be 2^{32} 32-bit variables. Now we classify X_i into four states depending on the conditions satisfied.

$$\boxed{\begin{array}{llll} \text{Const } (C) & : \forall i,j \;\; X_i = X_j \;\;, & \text{All } (A) & : \forall i,j \;\; i \neq j \Leftrightarrow X_i \neq X_j \;\;, \\ \text{Balance } (B) & : \bigoplus_{i=0}^{2^{32}-1} X_i = 0 \;\;, & \text{Unknown } (U) & : \text{unknown} \;\;. \end{array}}$$

Using the above notation, the following 6-round distinguishers are found.

$$(C, A, C, C) \xrightarrow{6r} (B, U, U, U) \;\; and \;\; (C, C, C, A) \xrightarrow{6r} (U, U, B, U) \;\; p = 1$$

These distinguishers can be extended to an 8-round distinguisher by adding two more rounds before the above 6 round path. Let $A_{(96)}$ be an "All" state of 96-bit word, and we divide it into 3 segments as $A_{(96)} = A_0|A_1|A_2$. Then the 8-round distinguishers are given as follows.

$$(A_0, C, A_1, A_2) \xrightarrow{8r} (B, U, U, U) \;\; and \;\; (A_0, A_1, A_2, C) \xrightarrow{8r} (U, U, B, U) \;\; p = 1$$

Using the above 8-round distinguisher, it turns out that 10-round 128-bit key CLEFIA can be attacked with complexity slightly less than 2^{128} F-function calculations. From the above observations, we conclude that full-round CLEFIA has enough security margin against this attack.

Algebraic Attack [8]. Let CLEFIA-I be a modified version of CLEFIA by replacing all 4-bit S-boxes by the identity function I, $I : \{0, 1\}^4 \rightarrow \{0, 1\}^4$, where $I(x) = x$. Based on the estimation method by Courtois and Pieprzyk the total number of terms can be estimated as follows [8]. $T = 81^8\binom{144}{8} > 2^{50+41} = 2^{91}$ 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 $T^3 = 2^{312}$. Although we give the results of the estimation, we should interpret these estimations with an extreme care: the real complexity of the XSL attacks is by no means clear at the time of writing and is the subject of much controversy [20, 16].

Related-Key Attack [3]. As for CLEFIA with a 128-bit key, L is generated by $L = GFN_{4,12}(CON^{(128)}, K)$. As in Table 2, $GFN_{4,12}$ has at least 28 active S-boxes, and we have $DCP_{max} \leq 2^{-130.76}$. 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.

Also for 192 and 256-bit keys, (L_L, L_R) is generated by applying $GFN_{8,10}$ to K_L, K_R. From Table 2, it has at least 29 differential active S-boxes, which implies there are no differential characteristics with probability more than 2^{-128}.

The above observations imply 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 for differential. Because all the bits in L are used in 2 or 6 consecutive rounds. As a result, we believe that CLEFIA holds strong immunity against related-key cryptanalysis. We also expect that CLEFIA holds enough immunity against other related-key type attacks including related-key boomerang and related-key rectangle attacks [5].

Security against Other Attacks. Due to the page limitation, the details of the security evaluation against known general attacks are omitted. Immunity against some of the known attacks can be estimated by the evaluation results of similar type attacks already mentioned in this section. We consider any attack does not threat full-round CLEFIA.

Table 6. Results on Hardware Performance of CLEFIA

	Key Length	Enc/Dec (cycles)	Key Setup (cycles)	Optimi-zation	Area (gates)	Freq. (MHz)	Speed (Mbps)	Speed/Area (Kbps/gate)
CLEFIA (0.09 μm)	128	18	12	area	5,979	225.83	1,605.94	268.63
				speed	12,009	422.29	3,003.00	250.06
		36	24	area	4,950	201.28	715.69	144.59
				speed	9,377	389.55	1,385.10	147.71
	192	22	20	area	8,536	206.56	1,201.85	140.81
	256	26	20	area	8,482	206.56	1,016.95	119.89
AES [21] (0.13 μm)	128	11	N/A	area	12,454	145.35	1,691.35	135.81
		54	N/A	area	5,398	131.24	311.09	57.63

Table 7. 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 [18]	single-block	128	10.6	N/A	N/A	8

5.2 Performance

Table 6 shows evaluation results of hardware performance of CLEFIA using a 0.09μm CMOS ASIC library, where one gate is equivalent to a 2-way NAND and the speed is evaluated under the worst-case condition. The Verilog-HDL models were synthesized by specifying area or speed optimization to a logic synthesis tool. The synthesized circuit of CLEFIA with 128-bit key by area optimization occupies only 5,979 gates at a throughput of about 1.60 Gbps. Although we take into account the difference of ASIC libraries, these figures indicate high efficiency of CLEFIA in hardware implementation compared to the best known results of hardware performance of AES [21].

Table 7 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 cycle/byte (1.48 Gbps on the processor) is achieved. The two-block parallel encryption is suitable for CTR mode, because two blocks can be processed simultaneously [18].

6 Conclusion

We proposed a 128-bit blockcipher CLEFIA, which supports 128-bit, 192-bit, and 256-bit keys. CLEFIA employs several new design approaches, including the DSM technique. As a result, enough immunity against known attacks is achieved. Moreover, the design of CLEFIA allows very efficient implementation in a variety of environments. Some results of highly efficient implementation are exemplified.

Acknowledgment

The authors thank Lars Knudsen, Bart Preneel, Vincent Rijmen, and Serge Vaudenay for their helpful comments. The authors also thank the anonymous referees for their valuable comments.

References

1. Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., Tokita, T.: Camellia: A 128-bit block cipher suitable for multiple platforms. In: Stinson, D.R., Tavares, S. (eds.) SAC 2000. LNCS, vol. 2012, pp. 41–54. Springer, Heidelberg (2001)
2. Barreto, P.S.L.M., Rijmen, V.: The Anubis block cipher. NESSIE, September 2000 (primitive submitted), Available at `http://www.cryptonessie.org/`
3. Biham, E.: New types of cryptanalytic attacks using related keys. Journal of Cryptology 7(4), 229–246 (1994)
4. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 12–23. Springer, Heidelberg (1999)
5. Biham, E., Dunkelman, O., Keller, N.: Related-key boomerang and rectangle attacks. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 507–525. Springer, Heidelberg (2005)
6. Biham, E., Dunkelman, O., Keller, N.: Related-key impossible differential attacks on 8-round AES-192. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 21–33. Springer, Heidelberg (2006)
7. Biham, E., Shamir, A.: Differential Cryptanalysis of the Data Encryption Standard. Springer, Heidelberg (1993)
8. Courtois, N., Pieprzyk, J.: Cryptanalysis of block ciphers with overdefined systems of equations. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 267–287. Springer, Heidelberg (2002)
9. Daemen, J., Knudsen, L.R., Rijmen, V.: The block cipher SQUARE. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 149–165. Springer, Heidelberg (1997)
10. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard (Information Security and Cryptography). Springer, Heidelberg (2002)
11. Data Encryption Standard: Federal Information Processing Standard (FIPS), Publication 46, National Bureau of Standards, U.S. Department of Commerce, Washington, DC (January 1977)
12. Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B., Lee, C., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J., Chee, S.: Hight: A new block cipher suitable for low-resource device. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 46–59. Springer, Heidelberg (2006)

13. Junod, P., Vaudenay, S.: FOX: A new family of block ciphers. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 114–129. Springer, Heidelberg (2004)
14. Kim, J., Hong, S., Sung, J., Lee, C., Lee, S.: Impossible differential cryptanalysis for block cipher structure. In: Johansson, T., Maitra, S. (eds.) INDOCRYPT 2003. LNCS, vol. 2904, pp. 82–96. Springer, Heidelberg (2003)
15. Lai, X., Massey, J.L., Murphy, S.: Markov ciphers and differential cryptanalysis. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 17–38. Springer, Heidelberg (1991)
16. Lim, C., Khoo, K.: An Analysis of XSL Applied to BES. In: Pre-proceedings of Fast Software Encryption '07, FSE'07, pp. 253–265 (2007)
17. Matsui, M.: Linear cryptanalysis of the data encryption standard. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
18. Matsui, M.: How far can we go on the x64 processors? In: Robshaw, M. (ed.) FSE 2006. LNCS, vol. 4047, pp. 341–358. Springer, Heidelberg (2006)
19. Matsui, M.: New block encryption algorithm MISTY. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 54–68. Springer, Heidelberg (1997)
20. Murphy, S., Robshaw, M.: Comments on the security of the AES and the XSL technique. Electronic Letters 39(1), 36–38 (2003)
21. Satoh, A., Morioka, S.: Hardware-focused performance comparison for the standard block ciphers AES, Camellia, and Triple-DES. In: Boyd, C., Mao, W. (eds.) ISC 2003. LNCS, vol. 2851, pp. 252–266. Springer, Heidelberg (2003)
22. Shirai, T., Preneel, B.: On Feistel ciphers using optimal diffusion mappings across multiple rounds. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 1–15. Springer, Heidelberg (2004)
23. Shirai, T., Shibutani, K.: On Feistel structures using a diffusion switching mechanism. In: Robshaw, M. (ed.) FSE 2006. LNCS, vol. 4047, pp. 41–56. Springer, Heidelberg (2006)
24. Zheng, Y., Matsumoto, T., Imai, H.: On the construction of block ciphers provably secure and not relying on any unproved hypotheses. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 461–480. Springer, Heidelberg (1990)

A Test Vectors

We give test vectors of CLEFIA for each key length. The data are expressed in hexadecimal form.

128-bit key:	
key	`ffeeddcc bbaa9988 77665544 33221100`
plaintext	`00010203 04050607 08090a0b 0c0d0e0f`
ciphertext	`de2bf2fd 9b74aacd f1298555 459494fd`
192-bit key:	
key	`ffeeddcc bbaa9988 77665544 33221100 f0e0d0c0 b0a09080`
plaintext	`00010203 04050607 08090a0b 0c0d0e0f`
ciphertext	`e2482f64 9f028dc4 80dda184 fde181ad`
256-bit key:	
key	`ffeeddcc bbaa9988 77665544 33221100`
	`f0e0d0c0 b0a09080 70605040 30201000`
plaintext	`00010203 04050607 08090a0b 0c0d0e0f`
ciphertext	`a1397814 289de80c 10da46d1 fa48b38a`

B Tables of S_0 and S_1

	S_0																S_1															
	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.a	.b	.c	.d	.e	.f	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.a	.b	.c	.d	.e	.f
0.	57	49	d1	c6	2f	33	74	fb	95	6d	82	ea	0e	b0	a8	1c	6c	da	c3	e9	4e	9d	0a	3d	b8	36	b4	38	13	34	0c	d9
1.	28	d0	4b	92	5c	ee	85	b1	c4	0a	76	3d	63	f9	17	af	bf	74	94	8f	b7	9c	e5	dc	9e	07	49	4f	98	2c	b0	93
2.	bf	a1	19	65	f7	7a	32	20	06	ce	e4	83	9d	5b	4c	d8	12	eb	cd	b3	92	e7	41	60	e3	21	27	3b	e6	19	d2	0e
3.	42	5d	2e	e8	d4	9b	0f	13	3c	89	67	c0	71	aa	b6	f5	91	11	c7	3f	2a	8e	a1	bc	2b	c8	c5	0f	5b	f3	87	8b
4.	a4	be	fd	8c	12	00	97	da	78	e1	cf	6b	39	43	55	26	fb	f5	de	20	c6	a7	84	ce	d8	65	51	c9	a4	ef	43	53
5.	30	98	cc	dd	eb	54	b3	8f	4e	16	fa	22	a5	77	09	61	25	5d	9b	31	e8	3e	0d	d7	80	ff	69	8a	ba	0b	73	5c
6.	d6	2a	53	37	45	c1	6c	ae	ef	70	08	99	8b	1d	f2	b4	6e	54	15	62	f6	35	30	52	a3	16	d3	28	32	fa	aa	5e
7.	e9	c7	9f	4a	31	25	fe	7c	d3	a2	bd	56	14	88	60	0b	cf	ea	ed	78	33	58	09	7b	63	c0	c1	46	1e	df	a9	99
8.	cd	e2	34	50	9e	dc	11	05	2b	b7	a9	48	ff	66	8a	73	55	04	c4	86	39	77	82	ec	40	18	90	97	59	dd	83	1f
9.	03	75	86	f1	6a	a7	40	c2	b9	2c	db	1f	58	94	3e	ed	9a	37	06	24	64	7c	a5	56	48	08	85	d0	61	26	ca	6f
a.	fc	1b	a0	04	b8	8d	e6	59	62	93	35	7e	ca	21	df	47	7e	6a	b6	71	a0	70	05	d1	45	8c	23	1c	f0	ee	89	ad
b.	15	f3	ba	7f	a6	69	c8	4d	87	3b	9c	01	e0	de	24	52	7a	4b	c2	2f	db	5a	4d	76	67	17	2d	f4	cb	b1	4a	a8
c.	7b	0c	68	1e	80	b2	5a	e7	ad	d5	23	f4	46	3f	91	c9	b5	22	47	3a	d5	10	4c	72	cc	00	f9	e0	fd	e2	fe	ae
d.	6e	84	72	bb	0d	18	d9	96	f0	5f	41	ac	27	c5	e3	3a	f8	5f	ab	f1	1b	42	81	d6	be	44	29	a6	57	b9	af	f2
e.	81	6f	07	a3	79	f6	2d	38	1a	44	5e	b5	d2	ec	cb	90	d4	75	66	bb	68	9f	50	02	01	3c	7f	8d	1a	88	bd	ac
f.	9a	36	e5	29	c3	4f	ab	64	51	f8	10	d7	bc	02	7d	8e	f7	e4	79	96	a2	fc	6d	b2	6b	03	e1	2e	7d	14	95	1d

C Figures

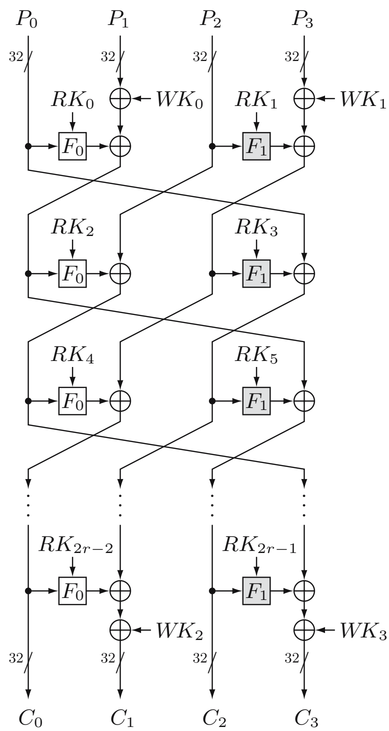

Fig. 1. *ENC*$_r$

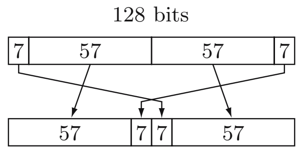

Fig. 2. *DoubleSwap*

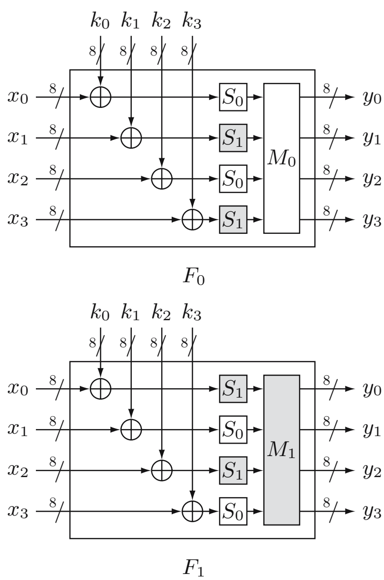

Fig. 3. F-functions