

Independent Submission  
Request for Comments: 6114  
Category: Informational  
ISSN: 2070-1721

M. Katagi  
S. Moriai  
Sony Corporation  
March 2011

## The 128-Bit Blockcipher CLEFIA

### Abstract

This document describes the specification of the blockcipher CLEFIA. CLEFIA is a 128-bit blockcipher, with key lengths of 128, 192, and 256 bits, which is compatible with the interface of the Advanced Encryption Standard (AES). The algorithm of CLEFIA was published in 2007, and its security has been scrutinized in the public community. CLEFIA is one of the new-generation lightweight blockcipher algorithms designed after AES. Among them, CLEFIA offers high performance in software and hardware as well as lightweight implementation in hardware. CLEFIA will be of benefit to the Internet, which will be connected to more distributed and constrained devices.

### Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This is a contribution to the RFC Series, independently of any other RFC stream. The RFC Editor has chosen to publish this document at its discretion and makes no statement about its value for implementation or deployment. Documents approved for publication by the RFC Editor are not a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at <http://www.rfc-editor.org/info/rfc6114>.

## Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

## Table of Contents

1. Introduction .....	3
2. Notations .....	3
3. CLEFIA Algorithm .....	4
4. CLEFIA Building Blocks .....	4
4.1. GFN_{d,r} .....	4
4.2. F-Functions .....	6
4.3. S-Boxes .....	7
4.4. Diffusion Matrices .....	9
5. Data Processing Part .....	9
5.1. Encryption/Decryption .....	9
5.2. The Numbers of Rounds .....	10
6. Key Scheduling Part .....	10
6.1. DoubleSwap Function .....	10
6.2. Overall Structure .....	11
6.3. Key Scheduling for a 128-Bit Key .....	11
6.4. Key Scheduling for a 192-Bit Key .....	11
6.5. Key Scheduling for a 256-Bit Key .....	12
6.6. Constant Values .....	13
7. Security Considerations .....	18
8. Informative References .....	18
Appendix A. Test Vectors .....	19
Appendix B. Test Vectors (Intermediate Values) .....	19

## 1. Introduction

Due to the widespread use of the Internet, devices with limited capabilities, e.g., wireless sensors, are connected to the network. In order to realize enough security for the network, cryptographic technologies suitable for such constrained devices are very important. This recent technology is called "lightweight cryptography", and the demand for lightweight cryptography is increasing.

In order to satisfy these needs, a 128-bit blockcipher, CLEFIA, was designed based on state-of-the-art techniques [FSE07]. CLEFIA is a 128-bit blockcipher, with key lengths of 128, 192, and 256 bits, which is compatible with the interface of AES [FIPS-197]. Since the cipher algorithm was published in 2007, its security has been scrutinized in the public community, but no security weaknesses have been reported so far.

CLEFIA is a lightweight blockcipher, since it can be implemented within 3 Kbytes using a 0.13-um standard Complementary Metal Oxide Semiconductor (CMOS) Application-Specific Integrated Circuit (ASIC) library. Many of the lightweight cryptographic algorithms sacrifice security and/or speed; however, CLEFIA provides high-level security of 128, 192, and 256 bits and high performance in software and hardware. CLEFIA will be of benefit to the Internet, which will be connected to more distributed and resource-constrained devices.

CLEFIA is proposed in ISO/IEC 29192-2 [ISO29192-2] and the CRYPTREC project for the revision of the e-Government recommended ciphers list in Japan [CRYPTREC].

Further information about CLEFIA, including reference implementation, test vectors, and security and performance evaluation, is available from <http://www.sony.net/clefia/>.

## 2. Notations

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

0x	: A prefix for a binary string in hexadecimal form
a b or (a b)	: Concatenation of a and b
(a,b) or (a b)	: Vector style representation of a b
a <- b	: Updating a value of a by a value of b
trans(a)	: Transposition of a vector or a matrix a
a XOR b	: Bitwise exclusive-OR operation

$\sim a$  : Logical negation  
 $a \lll b$  :  $b$ -bit left cyclic shift operation  
 $a^b$  :  $a$  raised to the power of  $b$   
 $a * b$  : Multiplication in  $GF(2^n)$  over a defined polynomial

### 3. CLEFIA Algorithm

The CLEFIA algorithm consists of two parts: a data processing part and a key scheduling part. The data processing part of CLEFIA consists of functions ENCr for encryption and DECr for decryption. The encryption/decryption process is as follows:

- Step 1. Key scheduling
- Step 2. Encrypting/decrypting each block of data using ENCr/DECr

The process of the key scheduling is described in Section 6, and the definitions of ENCr and DECr are explained in Section 5. CLEFIA supports 128-bit, 192-bit, and 256-bit keys, and the key scheduling and ENCr/DECr should be appropriately selected for its key length.

### 4. CLEFIA Building Blocks

#### 4.1. $GFN_{\{d,r\}}$

We first define the function  $GFN_{\{d,r\}}$ , which is a fundamental structure for CLEFIA, and then define a data processing part and a key scheduling part.

CLEFIA uses a 4-branch and an 8-branch generalized Feistel network. The 4-branch generalized Feistel network is used in the data processing part and the key scheduling for a 128-bit key. The 8-branch generalized Feistel network is applied in the key scheduling for a 192-bit/256-bit key. We denote the  $d$ -branch  $r$ -round generalized Feistel network employed in CLEFIA as  $GFN_{\{d,r\}}$ .

For  $d$  pairs of 32-bit inputs  $X_i$  and outputs  $Y_i$  ( $0 \leq i < d$ ), and  $dr/2$  32-bit round keys  $RK_{\{i\}}$  ( $0 \leq i < dr/2$ ),  $GFN_{\{d,r\}}$  ( $d = 4, 8$ ) is defined as follows.

$\text{GFN}_{\{4,r\}}(\text{RK}_{\{0\}}, \dots, \text{RK}_{\{2r-1\}}, X_0, X_1, X_2, X_3)$

input : 32-bit round keys  $\text{RK}_{\{0\}}, \dots, \text{RK}_{\{2r-1\}},$   
 32-bit data  $X_0, X_1, X_2, X_3,$

output: 32-bit data  $Y_0, Y_1, Y_2, Y_3$

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.  $T_1 \leftarrow T_1 \text{ XOR } F_0(\text{RK}_{\{2i\}}, T_0),$   
 $T_3 \leftarrow T_3 \text{ XOR } F_1(\text{RK}_{\{2i + 1\}}, T_2)$

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.  $Y_0 \mid Y_1 \mid Y_2 \mid Y_3 \leftarrow T_3 \mid T_0 \mid T_1 \mid T_2$

$\text{GFN}_{\{8,r\}}(\text{RK}_{\{0\}}, \dots, \text{RK}_{\{4r-1\}}, X_0, X_1, \dots, X_7)$

input : 32-bit round keys  $\text{RK}_{\{0\}}, \dots, \text{RK}_{\{4r-1\}},$   
 32-bit data  $X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7,$

output: 32-bit data  $Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Y_7$

Step 1.  $T_0 \mid T_1 \mid \dots \mid T_7 \leftarrow X_0 \mid X_1 \mid \dots \mid X_7$

Step 2. For  $i = 0$  to  $r - 1$  do the following:

Step 2.1.  $T_1 \leftarrow T_1 \text{ XOR } F_0(\text{RK}_{\{4i\}}, T_0),$   
 $T_3 \leftarrow T_3 \text{ XOR } F_1(\text{RK}_{\{4i + 1\}}, T_2),$   
 $T_5 \leftarrow T_5 \text{ XOR } F_0(\text{RK}_{\{4i + 2\}}, T_4),$   
 $T_7 \leftarrow T_7 \text{ XOR } F_1(\text{RK}_{\{4i + 3\}}, T_6)$

Step 2.2.  $T_0 \mid T_1 \mid \dots \mid T_6 \mid T_7 \leftarrow T_1 \mid T_2 \mid \dots \mid T_7 \mid T_0$

Step 3.  $Y_0 \mid Y_1 \mid \dots \mid Y_6 \mid Y_7 \leftarrow T_7 \mid T_0 \mid \dots \mid T_5 \mid T_6$

The inverse function  $\text{GFNINV}_{\{4,r\}}$  is obtained by changing the order of  $\text{RK}_{\{i\}}$  and the direction of word rotation at Step 2.2 and Step 3 in  $\text{GFN}_{\{4,r\}}$ .

GFNINV\_{4,r}(RK\_{0}, ..., RK\_{2r-1}, X0, X1, X2, X3)

input : 32-bit round keys RK\_{0}, ..., RK\_{2r-1},  
32-bit data X0, X1, X2, X3,

output: 32-bit data Y0, Y1, Y2, Y3

Step 1. T0 | T1 | T2 | T3 <- X0 | X1 | X2 | X3

Step 2. For i = 0 to r - 1 do the following:

Step 2.1. T1 <- T1 XOR F0(RK\_{2(r - i) - 2}, T0),  
T3 <- T3 XOR F1(RK\_{2(r - i) - 1}, T2)

Step 2.2. T0 | T1 | T2 | T3 <- T3 | T0 | T1 | T2

Step 3. Y0 | Y1 | Y2 | Y3 <- T1 | T2 | T3 | T0

#### 4.2. F-Functions

Two F-functions F0 and F1 used in GFN\_{d,r} are defined as follows:

F0(RK, x)

input : 32-bit round key RK, 32-bit data x,

output: 32-bit data y

Step 1. T <- RK XOR x

Step 2. Let T = T0 | T1 | T2 | T3, where Ti is 8-bit data,  
T0 <- S0(T0),  
T1 <- S1(T1),  
T2 <- S0(T2),  
T3 <- S1(T3)

Step 3. Let y = y0 | y1 | y2 | y3, where yi is 8-bit data,  
y <- M0 trans((T0, T1, T2, T3))

F1(RK, x)

input : 32-bit round key RK, 32-bit data x,

output: 32-bit data y

Step 1.  $T \leftarrow RK \text{ XOR } x$

Step 2. Let  $T = T0 \mid T1 \mid T2 \mid T3$ , where  $Ti$  is 8-bit data,

$T0 \leftarrow S1(T0),$

$T1 \leftarrow S0(T1),$

$T2 \leftarrow S1(T2),$

$T3 \leftarrow S0(T3)$

Step 3. Let  $y = y0 \mid y1 \mid y2 \mid y3$ , where  $yi$  is 8-bit data,

$y \leftarrow M1 \text{ trans}((T0, T1, T2, T3))$

S0 and S1 are nonlinear 8-bit S-boxes, and M0 and M1 are 4x4 diffusion matrices described in the following section. In each F-function, two S-boxes are used in the different order, and a different matrix is used.

#### 4.3. S-Boxes

CLEFIA employs two different types of 8-bit S-boxes: S0 is based on four 4-bit S-boxes, and S1 is based on the inverse function over  $GF(2^8)$  [CLEFIA1].

Tables 1 and 2 show the output values of S0 and S1, respectively. In these tables, all values are expressed in hexadecimal form. For an 8-bit input of an S-box, the upper 4 bits indicate a row and the lower 4 bits indicate a column. For example, if a value 0xab is input, 0x7e is output by S0 because it is on the cross line of the row indexed by "a." and the column indexed by ".b".

Table 1: S-Box S0

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

Table 2: S-Box S1

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



#### 4.4. Diffusion Matrices

The multiplications of a diffusion matrix  $M_0$  or  $M_1$ , and a vector  $T$  in Section 4.2, are obtained as follows.

$y = M_0 \text{ trans}((T_0, T_1, T_2, T_3))$ :

```

y0 =      T0 XOR (0x02 * T1) XOR (0x04 * T2) XOR (0x06 * T3),
y1 = (0x02 * T0) XOR      T1 XOR (0x06 * T2) XOR (0x04 * T3),
y2 = (0x04 * T0) XOR (0x06 * T1) XOR      T2 XOR (0x02 * T3),
y3 = (0x06 * T0) XOR (0x04 * T1) XOR (0x02 * T2) XOR      T3

```

$y = M_1 \text{ trans}((T_0, T_1, T_2, T_3))$ :

```

y0 =      T0 XOR (0x08 * T1) XOR (0x02 * T2) XOR (0x0a * T3),
y1 = (0x08 * T0) XOR      T1 XOR (0x0a * T2) XOR (0x02 * T3),
y2 = (0x02 * T0) XOR (0x0a * T1) XOR      T2 XOR (0x08 * T3),
y3 = (0x0a * T0) XOR (0x02 * T1) XOR (0x08 * T2) XOR      T3

```

In the above equations,  $*$  denotes a multiplication in  $GF(2^8)$  defined by the lexicographically first primitive polynomial  $z^8 + z^4 + z^3 + z^2 + 1$ . The constants 0x02, 0x04, 0x06, 0x08, and 0x0a are represented in hexadecimal form of finite field polynomials. For example, 0x02 identifies the finite field element  $z$ . 8-bit data  $T_i$  is also interpreted as a finite field element.

The mathematical background of two diffusion matrices and their choices are explained in [CLEFIA2].

### 5. Data Processing Part

#### 5.1. Encryption/Decryption

The data processing part of CLEFIA consists of ENCr for encryption and DECr for decryption. ENCr and DECr are based on the 4-branch generalized Feistel structure  $GFN_{\{4,r\}}$ . Let  $P, C$  be 128-bit plaintext and ciphertext, and let  $P_i, C_i$  ( $0 \leq i < 4$ ) be divided 32-bit plaintexts and ciphertexts where  $P = P_0 \parallel P_1 \parallel P_2 \parallel P_3$  and  $C = C_0 \parallel C_1 \parallel C_2 \parallel C_3$ , and let  $WK_0, WK_1, WK_2, WK_3$  be 32-bit whitening keys and  $RK_{\{i\}}$  ( $0 \leq i < 2r$ ) be 32-bit round keys provided by the key scheduling part. Then,  $r$ -round encryption function ENCr is defined as follows:

Step 1.  $T_0 \mid T_1 \mid T_2 \mid T_3 \leftarrow P_0 \mid (P_1 \text{ XOR } WK_0) \mid P_2 \mid (P_3 \text{ XOR } WK_1)$

Step 2.  $T_0 \mid T_1 \mid T_2 \mid T_3$   
 $\leftarrow GFN_{\{4,r\}}(RK_{\{0\}}, \dots, RK_{\{2r-1\}}, T_0, T_1, T_2, T_3)$

Step 3.  $C_0 \mid C_1 \mid C_2 \mid C_3 \leftarrow T_0 \mid (T_1 \text{ XOR } WK_2) \mid T_2 \mid (T_3 \text{ XOR } WK_3)$

The decryption function  $DEC_r$  is defined as follows:

Step 1.  $T_0 \mid T_1 \mid T_2 \mid T_3 \leftarrow C_0 \mid (C_1 \text{ XOR } WK_2) \mid C_2 \mid (C_3 \text{ XOR } WK_3)$

Step 2.  $T_0 \mid T_1 \mid T_2 \mid T_3$   
 $\leftarrow GFNINV_{\{4,r\}}(RK_{\{0\}}, \dots, RK_{\{2r-1\}}, T_0, T_1, T_2, T_3)$

Step 3.  $P_0 \mid P_1 \mid P_2 \mid P_3 \leftarrow T_0 \mid (T_1 \text{ XOR } WK_0) \mid T_2 \mid (T_3 \text{ XOR } WK_1)$

## 5.2. The Numbers of Rounds

The number of rounds,  $r$ , is 18, 22, and 26 for 128-bit, 192-bit, and 256-bit keys, respectively. The total number of  $RK_{\{i\}}$  depends on the key length. The data processing part requires 36, 44, and 52 round keys for 128-bit, 192-bit, and 256-bit keys, respectively.

## 6. Key Scheduling Part

The key scheduling part of CLEFIA supports 128-bit, 192-bit, 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.

### 6.1. DoubleSwap Function

We first define the DoubleSwap function, which is used in the key scheduling part.

The DoubleSwap Function  $\text{Sigma}(X)$ :

For 128-bit data  $X$ ,

$Y = \text{Sigma}(X)$   
 $= X[7-63] \mid X[121-127] \mid X[0-6] \mid X[64-120],$

where  $X[a-b]$  denotes a bit string cut from the  $a$ -th bit to the  $b$ -th bit of  $X$ . Bit 0 is the most significant bit.

## 6.2. Overall Structure

The key scheduling part of CLEFIA provides whitening keys and round keys for the data processing part. Let  $K$  be the key and  $L$  be an intermediate key, and the key scheduling part consists of the following two steps.

1. Generating  $L$  from  $K$ .
2. Expanding  $K$  and  $L$  (Generating  $WK_i$  and  $RK_{\{j\}}$ ).

To generate  $L$  from  $K$ , the key schedule for a 128-bit key uses a 128-bit permutation  $GFN_{\{4,12\}}$ , while the key schedules for 192/256-bit keys use a 256-bit permutation  $GFN_{\{8,10\}}$ .

## 6.3. 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  $K = K_0 \mid K_1 \mid K_2 \mid K_3$  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  $CON_{128}[i]$  are explained in Section 6.6.

(Generating  $L$  from  $K$ )

```
Step 1.  $L \leftarrow GFN_{\{4,12\}}(CON_{128}[0], \dots, CON_{128}[23], K_0, \dots, K_3)$ 
```

(Expanding  $K$  and  $L$ )

```
Step 2.  $WK_0 \mid WK_1 \mid WK_2 \mid WK_3 \leftarrow K$ 
```

```
Step 3. For  $i = 0$  to 8 do the following:
```

```
     $T \leftarrow L \text{ XOR } (CON_{128}[24 + 4i] \mid CON_{128}[24 + 4i + 1]$   

        $\mid CON_{128}[24 + 4i + 2] \mid CON_{128}[24 + 4i + 3])$ 
```

```
     $L \leftarrow \text{Sigma}(L)$ 
```

```
    if  $i$  is odd:  $T \leftarrow T \text{ XOR } K$ 
```

```
     $RK_{\{4i\}} \mid RK_{\{4i + 1\}} \mid RK_{\{4i + 2\}} \mid RK_{\{4i + 3\}} \leftarrow T$ 
```

## 6.4. Key Scheduling for a 192-Bit Key

Two 128-bit values  $KL$  and  $KR$  are generated from a 192-bit key  $K = K_0 \mid K_1 \mid K_2 \mid K_3 \mid K_4 \mid K_5$ , where  $K_i$  is 32-bit data. Then, two 128-bit values  $LL$  and  $LR$  are generated by applying  $GFN_{\{8,10\}}$ , which takes  $CON_{192}[i]$  ( $0 \leq i < 40$ ) as round keys and  $KL \mid KR$  as a 256-bit input.

Then,  $KL, KR$  and  $LL, LR$  are used to generate  $WK_i$  ( $0 \leq i < 4$ ) and  $RK_{\{j\}}$  ( $0 \leq j < 44$ ) in the following steps. In the latter part, forty-four 32-bit constant values  $CON_{192}[i]$  ( $40 \leq i < 84$ ) 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.

#### 6.5. Key Scheduling for a 256-Bit Key

The key scheduling for a 256-bit key is almost the same as that for a 192-bit key, except for constant values, the required number of  $RK_i$ , and the initialization of  $KR$ .

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  $CON_{256}[i]$  ( $0 \leq i < 40$ ) as round keys to generate  $LL$  and  $LR$ , and then to generate  $RK_{\{j\}}$  ( $0 \leq j < 52$ ), we use fifty-two 32-bit constant values  $CON_{256}[i]$  ( $40 \leq i < 92$ ).

(Generating  $LL, LR$  from  $KL, KR$  for a  $k$ -bit key)

Step 1. Set  $k = 192$  or  $k = 256$

Step 2. If  $k = 192$  :

$KL \leftarrow K0 \mid K1 \mid K2 \mid K3, KR \leftarrow K4 \mid K5 \mid \sim K0 \mid \sim K1$

else if  $k = 256$  :

$KL \leftarrow K0 \mid K1 \mid K2 \mid K3, KR \leftarrow K4 \mid K5 \mid K6 \mid K7$

Step 3. Let  $KL = KL0 \mid KL1 \mid KL2 \mid KL3$

$KR = KR0 \mid KR1 \mid KR2 \mid KR3$

$LL \mid LR \leftarrow$

$GFN_{\{8,10\}}(CON_k[0], \dots, CON_k[39],$   
 $KL0, \dots, KL3, KR0, \dots, KR3)$

(Expanding  $KL, KR$  and  $LL, LR$  for a  $k$ -bit key)

Step 4.  $WK0 \mid WK1 \mid WK2 \mid WK3 \leftarrow KL \text{ XOR } KR$

Step 5. For  $i = 0$  to 10 (if  $k = 192$ ),  
or 12 (if  $k = 256$ ) do the following:

```

If  $(i \bmod 4) = 0$  or 1:
   $T \leftarrow LL \text{ XOR } (CON\_k[40 + 4i] \mid CON\_k[40 + 4i + 1] \mid$ 
     $\mid CON\_k[40 + 4i + 2] \mid CON\_k[40 + 4i + 3])$ 
   $LL \leftarrow \text{Sigma}(LL)$ 
  if  $i$  is odd:  $T \leftarrow T \text{ XOR } KR$ 
else:
   $T \leftarrow LR \text{ XOR } (CON\_k[40 + 4i] \mid CON\_k[40 + 4i + 1] \mid$ 
     $\mid CON\_k[40 + 4i + 2] \mid CON\_k[40 + 4i + 3])$ 
   $LR \leftarrow \text{Sigma}(LR)$ 
  if  $i$  is odd:  $T \leftarrow T \text{ XOR } KL$ 

 $RK_{\{4i\}} \mid RK_{\{4i + 1\}} \mid RK_{\{4i + 2\}} \mid RK_{\{4i + 3\}} \leftarrow T$ 

```

## 6.6. Constant Values

32-bit constant values  $CON\_k[i]$  are used in the key scheduling algorithm. We need 60, 84, and 92 constant values for 128-bit, 192-bit, and 256-bit keys, respectively. Let  $P(16) = 0xb7e1$  ( $= (e-2)2^{16}$ ) and  $Q(16) = 0x243f$  ( $= (\pi-3)2^{16}$ ), where  $e$  is the base of the natural logarithm (2.71828...) and  $\pi$  is the circle ratio (3.14159...).  $CON\_k[i]$ , for  $k = 128, 192, 256$ , are generated as follows (see Table 3 for the repetition numbers  $l\_k$  and the initial values  $IV\_k$ ).

Step 1.  $T\_k[0] \leftarrow IV\_k$

Step 2. For  $i = 0$  to  $l\_k - 1$  do the following:

Step 2.1.  $CON\_k[2i] \leftarrow (T\_k[i] \text{ XOR } P) \mid (\sim T\_k[i] \lll 1)$

Step 2.2.  $CON\_k[2i + 1] \leftarrow (\sim T\_k[i] \text{ XOR } Q) \mid (T\_k[i] \lll 8)$

Step 2.3.  $T\_k[i + 1] \leftarrow T\_k[i] * (0x0002^{-1})$

In Step 2.3, the multiplications are performed in the field  $GF(2^{16})$  defined by a primitive polynomial  $z^{16} + z^{15} + z^{13} + z^{11} + z^5 + z^4 + 1$  ( $= 0x1a831$ ).  $0x0002^{-1}$  denotes the multiplicative inverse of the finite field element  $z$ . The selection criteria of  $IV$  and the primitive polynomial are shown in [CLEFIA1].

Table 3: Required Numbers of Constant Values

k	# of CON_k[i]	l_k	IV_k
128	60	30	0x428a
192	84	42	0x7137
256	92	46	0xb5c0

Tables 4-6 show the values of  $T_k[i]$  ( $k = 128, 192, 256$ ), and Tables 7-9 show the values of  $CON_k[i]$  ( $k = 128, 192, 256$ ).

Table 4:  $T_{128}[i]$ 

i	0	1	2	3	4	5	6	7
$T_{128}[i]$	428a	2145	c4ba	625d	e536	729b	ed55	a2b2
i	8	9	10	11	12	13	14	15
$T_{128}[i]$	5159	fcba	7e5a	3f2d	cb8e	65c7	e6fb	a765
i	16	17	18	19	20	21	22	23
$T_{128}[i]$	87aa	43d5	f5f2	7af9	e964	74b2	3a59	c934
i	24	25	26	27	28	29		
$T_{128}[i]$	649a	324d	cd3e	669f	e757	a7b3		

Table 5:  $T_{192}[i]$ 

i	0	1	2	3	4	5	6	7
$T_{192}[i]$	7137	ec83	a259	8534	429a	214d	c4be	625f
i	8	9	10	11	12	13	14	15
$T_{192}[i]$	e537	a683	8759	97b4	4bda	25ed	c6ee	6377
i	16	17	18	19	20	21	22	23
$T_{192}[i]$	e5a3	a6c9	877c	43be	21df	c4f7	b663	8f29
i	24	25	26	27	28	29	30	31
$T_{192}[i]$	938c	49c6	24e3	c669	b72c	5b96	2dcb	c2fd
i	32	33	34	35	36	37	38	39
$T_{192}[i]$	b566	5ab3	f941	a8b8	545c	2a2e	1517	de93
i	40	41						
$T_{192}[i]$	bb51	89b0						

Table 6: T\_256[i]

i	0	1	2	3	4	5	6	7
T_256[i]	b5c0	5ae0	2d70	16b8	0b5c	05ae	02d7	d573
i	8	9	10	11	12	13	14	15
T_256[i]	bea1	8b48	45a4	22d2	1169	dcac	6e56	372b
i	16	17	18	19	20	21	22	23
T_256[i]	cf8d	b3de	59ef	f8ef	a86f	802f	940f	9e1f
i	24	25	26	27	28	29	30	31
T_256[i]	9b17	9993	98d1	9870	4c38	261c	130e	0987
i	32	33	34	35	36	37	38	39
T_256[i]	d0db	bc75	8a22	4511	f690	7b48	3da4	1ed2
i	40	41	42	43	44	45		
T_256[i]	0f69	d3ac	69d6	34eb	ce6d	b32e		

Table 7: CON\_128[i] (0 ≤ i &lt; 60)

i	0	1	2	3
CON_128[i]	f56b7aeb	994a8a42	96a4bd75	fa854521
i	4	5	6	7
CON_128[i]	735b768a	1f7abac4	d5bc3b45	b99d5d62
i	8	9	10	11
CON_128[i]	52d73592	3ef636e5	c57a1ac9	a95b9b72
i	12	13	14	15
CON_128[i]	5ab42554	369555ed	1553ba9a	7972b2a2
i	16	17	18	19
CON_128[i]	e6b85d4d	8a995951	4b550696	2774b4fc
i	20	21	22	23
CON_128[i]	c9bb034b	a59a5a7e	88cc81a5	e4ed2d3f
i	24	25	26	27
CON_128[i]	7c6f68e2	104e8ecb	d2263471	be07c765
i	28	29	30	31
CON_128[i]	511a3208	3d3bfbe6	1084b134	7ca565a7
i	32	33	34	35
CON_128[i]	304bf0aa	5c6aaa87	f4347855	9815d543
i	36	37	38	39
CON_128[i]	4213141a	2e32f2f5	cd180a0d	a139f97a
i	40	41	42	43
CON_128[i]	5e852d36	32a464e9	c353169b	af72b274
i	44	45	46	47
CON_128[i]	8db88b4d	e199593a	7ed56d96	12f434c9
i	48	49	50	51
CON_128[i]	d37b36cb	bf5a9a64	85ac9b65	e98d4d32
i	52	53	54	55
CON_128[i]	7adf6582	16fe3ecd	d17e32c1	bd5f9f66
i	56	57	58	59
CON_128[i]	50b63150	3c9757e7	1052b098	7c73b3a7

Table 8: CON\_192[i] (0 ≤ i &lt; 84)

i	0	1	2	3
CON_192[i]	c6d61d91	aaf73771	5b6226f8	374383ec
i	4	5	6	7
CON_192[i]	15b8bb4c	799959a2	32d5f596	5ef43485
i	8	9	10	11
CON_192[i]	f57b7acb	995a9a42	96acbd65	fa8d4d21
i	12	13	14	15
CON_192[i]	735f7682	1f7ebec4	d5be3b41	b99f5f62
i	16	17	18	19
CON_192[i]	52d63590	3ef737e5	1162b2f8	7d4383a6
i	20	21	22	23
CON_192[i]	30b8f14c	5c995987	2055d096	4c74b497
i	24	25	26	27
CON_192[i]	fc3b684b	901ada4b	920cb425	fe2ded25
i	28	29	30	31
CON_192[i]	710f7222	1d2eeec6	d4963911	b8b77763
i	32	33	34	35
CON_192[i]	524234b8	3e63a3e5	1128b26c	7d09c9a6
i	36	37	38	39
CON_192[i]	309df106	5cbc7c87	f45f7883	987ebe43
i	40	41	42	43
CON_192[i]	963ebc41	fa1fdf21	73167610	1f37f7c4
i	44	45	46	47
CON_192[i]	01829338	6da363b6	38c8e1ac	54e9298f
i	48	49	50	51
CON_192[i]	246dd8e6	484c8c93	fe276c73	9206c649
i	52	53	54	55
CON_192[i]	9302b639	ff23e324	7188732c	1da969c6
i	56	57	58	59
CON_192[i]	00cd91a6	6cec2cb7	ec7748d3	8056965b
i	60	61	62	63
CON_192[i]	9a2aa469	f60bcb2d	751c7a04	193dfdc2
i	64	65	66	67
CON_192[i]	02879532	6ea666b5	ed524a99	8173b35a
i	68	69	70	71
CON_192[i]	4ea00d7c	228141f9	1f59ae8e	7378b8a8
i	72	73	74	75
CON_192[i]	e3bd5747	8f9c5c54	9dcfaba3	f1ee2e2a
i	76	77	78	79
CON_192[i]	a2f6d5d1	ced71715	697242d8	055393de
i	80	81	82	83
CON_192[i]	0cb0895c	609151bb	3e51ec9e	5270b089



Table 9: CON\_256[i] (0 ≤ i &lt; 92)

i	0	1	2	3
CON_256[i]	0221947e	6e00c0b5	ed014a3f	8120e05a
i	4	5	6	7
CON_256[i]	9a91a51f	f6b0702d	a159d28f	cd78b816
i	8	9	10	11
CON_256[i]	bcbde947	d09c5c0b	b24ff4a3	de6eae05
i	12	13	14	15
CON_256[i]	b536fa51	d917d702	62925518	0eb373d5
i	16	17	18	19
CON_256[i]	094082bc	6561a1be	3ca9e96e	5088488b
i	20	21	22	23
CON_256[i]	f24574b7	9e64a445	9533ba5b	f912d222
i	24	25	26	27
CON_256[i]	a688dd2d	caa96911	6b4d46a6	076cacdc
i	28	29	30	31
CON_256[i]	d9b72353	b596566e	80ca91a9	eceb2b37
i	32	33	34	35
CON_256[i]	786c60e4	144d8dcf	043f9842	681edeb3
i	36	37	38	39
CON_256[i]	ee0e4c21	822fef59	4f0e0e20	232feff8
i	40	41	42	43
CON_256[i]	1f8eaf20	73af6fa8	37ceffa0	5bef2f80
i	44	45	46	47
CON_256[i]	23eed7e0	4fcf0f94	29fec3c0	45df1f9e
i	48	49	50	51
CON_256[i]	2cf6c9d0	40d7179b	2e72ccd8	42539399
i	52	53	54	55
CON_256[i]	2f30ce5c	4311d198	2f91cf1e	43b07098
i	56	57	58	59
CON_256[i]	fb9d9678f	97f8384c	91fdb3c7	fdcd1c26
i	60	61	62	63
CON_256[i]	a4efd9e3	c8ce0e13	be66ecf1	d2478709
i	64	65	66	67
CON_256[i]	673a5e48	0b1bdbbd0	0b948714	67b575bc
i	68	69	70	71
CON_256[i]	3dc3ebba	51e2228a	f2f075dd	9ed11145
i	72	73	74	75
CON_256[i]	417112de	2d5090f6	cca9096f	a088487b
i	76	77	78	79
CON_256[i]	8a4584b7	e664a43d	a933c25b	c512d21e
i	80	81	82	83
CON_256[i]	b888e12d	d4a9690f	644d58a6	086cacd3
i	84	85	86	87
CON_256[i]	de372c53	b216d669	830a9629	ef2beb34
i	88	89	90	91
CON_256[i]	798c6324	15ad6dce	04cf99a2	68ee2eb3

## 7. Security Considerations

The security of CLEFIA has been scrutinized in the public community, but no security weaknesses have been found for full-round CLEFIA to date, neither by the designers nor by independent cryptographers. Security evaluation by the designers is described in [CLEFIA3], and a list of published cryptanalysis results by external cryptographers is available from [http://www.sony.net/Products/cryptography/clefi/technical/related\\_material.html](http://www.sony.net/Products/cryptography/clefi/technical/related_material.html).

## 8. Informative References

- [CLEFIA1] The 128-bit Blockcipher CLEFIA - Algorithm Specification, Revision 1.0, June 1, 2007, Sony Corporation, <http://www.sony.net/Products/cryptography/clefi/technical/data/clefi-spec-1.0.pdf>.
- [CLEFIA2] The 128-bit blockcipher CLEFIA - Design Rationale, Revision 1.0, June 1, 2007, Sony Corporation, <http://www.sony.net/Products/cryptography/clefi/technical/data/clefi-design-1.0.pdf>.
- [CLEFIA3] The 128-bit blockcipher CLEFIA - Security and Performance Evaluations, Revision 1.0, June 1, 2007, Sony Corporation, <http://www.sony.net/Products/cryptography/clefi/technical/data/clefi-eval-1.0.pdf>.
- [CRYPTREC] Cryptography Research and Evaluation Committees, <http://www.cryptrec.go.jp/>.
- [FIPS-197] National Institute of Standards and Technology, "Advanced Encryption Standard (AES)", FIPS 197, November 2001, <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>.
- [FSE07] Shirai, T., Shibutani, K., Akishita, T., Moriai, S., and T. Iwata, "The 128-bit Blockcipher CLEFIA", proceedings of Fast Software Encryption 2007 - FSE 2007, LNCS 4593, pp. 181-195, Springer-Verlag, 2007.
- [ISO29192-2] ISO/IEC 29192-2, "Information technology - Security techniques - Lightweight cryptography - Part 2: Block ciphers", [http://www.iso.org/iso/iso\\_catalogue/catalogue\\_tc/catalogue\\_detail.htm?csnumber=56552](http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=56552).

## Appendix A. Test Vectors

In this appendix, we give test vectors of CLEFIA for each key length. The data are expressed in hexadecimal form. For the intermediate values of these vectors, refer to Appendix B.

## 128-bit key:

```
key          ffeeddcc bbba9988 77665544 33221100
plaintext    00010203 04050607 08090a0b 0c0d0e0f
ciphertext   de2bf2fd 9b74aacd f1298555 459494fd
```

## 192-bit key:

```
key          ffeeddcc bbba9988 77665544 33221100
             f0e0d0c0 b0a09080
plaintext    00010203 04050607 08090a0b 0c0d0e0f
ciphertext   e2482f64 9f028dc4 80dda184 fde181ad
```

## 256-bit key:

```
key          ffeeddcc bbba9988 77665544 33221100
             f0e0d0c0 b0a09080 70605040 30201000
plaintext    00010203 04050607 08090a0b 0c0d0e0f
ciphertext   a1397814 289de80c 10da46d1 fa48b38a
```

## Appendix B. Test Vectors (Intermediate Values)

## 128-bit key:

```
key          ffeeddcc bbba9988 77665544 33221100
plaintext    00010203 04050607 08090a0b 0c0d0e0f
ciphertext   de2bf2fd 9b74aacd f1298555 459494fd
```

```
L           8f89a61b 9db9d0f3 93e65627 da0d027e
```

```
WK_{0,1,2,3} ffeeddcc bbba9988 77665544 33221100
RK_{0,1,2,3} f3e6cef9 8df75e38 41c06256 640ac51b
RK_{4,5,6,7} 6a27e20a 5a791b90 e8c528dc 00336ea3
RK_{8,9,10,11} 59cd17c4 28565583 312a37cc c08abd77
RK_{12,13,14,15} 7e8e7eec 8be7e949 d3f463d6 a0aad6aa
RK_{16,17,18,19} e75eb039 0d657eb9 018002e2 9117d009
RK_{20,21,22,23} 9f98d11e babee8cf b0369efa d3aaef0d
RK_{24,25,26,27} 3438f93b f9cea4a0 68df9029 b869b4a7
RK_{28,29,30,31} 24d6406d e74bc550 41c28193 16de4795
RK_{32,33,34,35} a34a20f5 33265d14 b19d0554 5142f434
```

plaintext		00010203	04050607	08090a0b	0c0d0e0f
initial whitening key			ffeeddcc		bbaa9988
after whitening		00010203	fbebdbcb	08090a0b	b7a79787
Round 1	input	00010203	fbebdbcb	08090a0b	b7a79787
	F-function	F0		F1	
	input	00010203		08090a0b	
	round key	f3e6cef9		8df75e38	
	after key add	f3e7ccfa		85fe5433	
	after S	290246e1		777de8e8	
	after M	547a3193		abf12070	
Round 2	input	af91ea58	08090a0b	1c56b7f7	00010203
	F-function	F0		F1	
	input	af91ea58		1c56b7f7	
	round key	41c06256		640ac51b	
	after key add	ee51880e		785c72ec	
	after S	cb5d2b0c		63a5edd2	
	after M	f51cebb3		82dfe347	
Round 3	input	fd15e1b8	1c56b7f7	82dee144	af91ea58
	F-function	F0		F1	
	input	fd15e1b8		82dee144	
	round key	6a27e20a		5a791b90	
	after key add	973203b2		d8a7fad4	
	after S	c2c7c6c2		be59e10d	
	after M	d8dfd8de		e15ea81c	
Round 4	input	c4896f29	82dee144	4ecf4244	fd15e1b8
	F-function	F0		F1	
	input	c4896f29		4ecf4244	
	round key	e8c528dc		00336ea3	
	after key add	2c4c47f5		4efc2ce7	
	after S	9da4dafc		43bce638	
	after M	b5b28e96		b65c519a	
Round 5	input	376c6fd2	4ecf4244	4b49b022	c4896f29
	F-function	F0		F1	
	input	376c6fd2		4b49b022	
	round key	59cd17c4		28565583	
	after key add	6ea17816		631fe5a1	
	after S	f26ad3e5		62af9f1b	
	after M	29f08afd		be01d127	

Round 6	input	673fc8b9 4b49b022	7a88be0e 376c6fd2
	F-function	F0	F1
	input	673fc8b9	7a88be0e
	round key	312a37cc	c08abd77
	after key add	5615ff75	ba020379
	after S	b39c8e58	2dd1e9a2
	after M	5999a79e	0429b329
Round 7	input	12d017bc 7a88be0e	3345dcfb 673fc8b9
	F-function	F0	F1
	input	12d017bc	3345dcfb
	round key	7e8e7eec	8be7e949
	after key add	6c5e6950	b8a235b2
	after S	8b737025	67a08eba
	after M	6ed11b09	dfd3cd32
Round 8	input	1459a507 3345dcfb	b8ec058b 12d017bc
	F-function	F0	F1
	input	1459a507	b8ec058b
	round key	d3f463d6	a0aad6aa
	after key add	c7adc6d1	1846d321
	after S	e7ee5a5f	9e97f1a1
	after M	8c9d011c	93684eec
Round 9	input	bfd8dde7 b8ec058b	81b85950 1459a507
	F-function	F0	F1
	input	bfd8dde7	81b85950
	round key	e75eb039	0d657eb9
	after key add	58866dde	8cdd27e9
	after S	4e821daf	59c56044
	after M	e6d6501e	6d5839b4
Round 10	input	5e3a5595 81b85950	79019cb3 bfd8dde7
	F-function	F0	F1
	input	5e3a5595	79019cb3
	round key	018002e2	9117d009
	after key add	5fba5777	e8164cba
	after S	612d8f7b	0185a49c
	after M	3a1b0e97	b9b479c8
Round 11	input	bba357c7 79019cb3	066ca42f 5e3a5595
	F-function	F0	F1
	input	bba357c7	066ca42f
	round key	9f98d11e	babee8cf
	after key add	243b86d9	bcd24ce0
	after S	f70f1144	cb72a481
	after M	28974052	4a6700b1

Round 12	input	5196dce1 066ca42f	145d5524 bba357c7
	F-function	F0	F1
	input	5196dce1	145d5524
	round key	b0369efa	d3aaef0d
	after key add	e1a0421b	c7f7ba29
	after S	6f7efd4f	72642dce
	after M	ffb5db32	907d3820
Round 13	input	f9d97f1d 145d5524	2bde6fe7 5196dce1
	F-function	F0	F1
	input	f9d97f1d	2bde6fe7
	round key	3438f93b	f9cea4a0
	after key add	cde18626	d210cb47
	after S	3f751141	ab28e0da
	after M	0a744c28	1c3e38a3
Round 14	input	1e29190c 2bde6fe7	4da8e442 f9d97f1d
	F-function	F0	F1
	input	1e29190c	4da8e442
	round key	68df9029	b869b4a7
	after key add	76f68925	f5c150e5
	after S	fe6db7e7	fc0c25f6
	after M	aaa2c803	c4315b8d
Round 15	input	817ca7e4 4da8e442	3de82490 1e29190c
	F-function	F0	F1
	input	817ca7e4	3de82490
	round key	24d6406d	e74bc550
	after key add	a5aae789	daa3e1c0
	after S	8d233818	2904757b
	after M	7bd4cced	eac2f0fb
Round 16	input	367c28af 3de82490	f4ebe9f7 817ca7e4
	F-function	F0	F1
	input	367c28af	f4ebe9f7
	round key	41c28193	16de4795
	after key add	77bea93c	e235ae62
	after S	7c4a935b	669b8953
	after M	598e6940	c119609f
Round 17	input	64664dd0 f4ebe9f7	4065c77b 367c28af
	F-function	F0	F1
	input	64664dd0	4065c77b
	round key	a34a20f5	33265d14
	after key add	c72c6d25	73439a6f
	after S	e7e61de7	788c85b4
	after M	2ac01b0a	c755adfa

Round 18	input	de2bf2fd	4065c77b	f1298555	64664dd0
	F-function	F0		F1	
	input	de2bf2fd		f1298555	
	round key	b19d0554		5142f434	
	after key add	6fb6f7a9		a06b7161	
	after S	b44d648c		7e99ea2a	
	after M	ac7738f2		12d0c82d	

output	de2bf2fd	ec12ff89	f1298555	76b685fd
final whitening key		77665544		33221100
after whitening	de2bf2fd	9b74aacd	f1298555	459494fd
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

LL	db05415a	800082db	7cb8186c	d788c5f3
LR	1ca9b2e1	b4606829	c92dd35e	2258a432
WK_{0,1,2,3}	0f0e0d0c	0b0a0908	77777777	77777777
RK_{0,1,2,3}	4d3bfd1b	7a1f5dfa	0fae6e7c	c8bf3237
RK_{4,5,6,7}	73c2eeb8	dd429ec5	e220b3af	c9135e73
RK_{8,9,10,11}	38c46a07	fc2ce4ba	370abf2d	b05e627b
RK_{12,13,14,15}	38351b2f	74bd6e1e	1b7c7dce	92cfc98e
RK_{16,17,18,19}	509b31a6	4c5ad53c	6fc2ba33	e1e5c878
RK_{20,21,22,23}	419a74b9	1dd79e0e	240a33d2	9dabfd09
RK_{24,25,26,27}	6e3ff82a	74ac3ffd	b9696e2e	cc0b3a38
RK_{28,29,30,31}	ed785cbd	9c077c13	04978d83	2ec058ba
RK_{32,33,34,35}	4bbd5f6a	31fe8de8	b76da574	3a6fa8e7
RK_{36,37,38,39}	521213ce	4f1f59d8	c13624f6	ee91f6a4
RK_{40,41,42,43}	17f68fde	f6c360a9	6288bc72	c0ad856b

plaintext	00010203	04050607	08090a0b	0c0d0e0f
initial whitening key		0f0e0d0c		0b0a0908
after whitening	00010203	0b0b0b0b	08090a0b	07070707

Round 1	input	00010203	0b0b0b0b	08090a0b	07070707
	F-function	F0		F1	
	input	00010203		08090a0b	
	round key	4d3bfd1b		7a1f5dfa	
	after key add	4d3aff18		721657f1	
	after S	43c58e9e		ed85d736	
	after M	b5021a3b		c397f62b	

Round	2	input	be091130	08090a0b	c490f12c	00010203
	F-function		F0		F1	
	input		be091130		c490f12c	
	round key		0fae6e7c		c8bf3237	
	after key add		b1a77f4c		0c2fc31b	
	after S		f3d10ba4		13d83a3d	
	after M		9fba69c1		6683cae3	
Round	3	input	97b363ca	c490f12c	6682c8e0	be091130
	F-function		F0		F1	
	input		97b363ca		6682c8e0	
	round key		73c2eeb8		dd429ec5	
	after key add		e4718d72		bbc05625	
	after S		79ea66ed		f47b0d7a	
	after M		61c21ea5		120e06e2	
Round	4	input	a552ef89	6682c8e0	ac0717d2	97b363ca
	F-function		F0		F1	
	input		a552ef89		ac0717d2	
	round key		e220b3af		c9135e73	
	after key add		47725c26		651449a1	
	after S		daeda541		355c651b	
	after M		28a43c63		cblab573	
Round	5	input	4e26f483	ac0717d2	5ca9d6b9	a552ef89
	F-function		F0		F1	
	input		4e26f483		5ca9d6b9	
	round key		38c46a07		fc2ce4ba	
	after key add		76e29e84		a0853203	
	after S		fe663e39		7edcc7c6	
	after M		5ce7dafe		ac7f4e3e	
Round	6	input	f0e0cd2c	5ca9d6b9	092da1b7	4e26f483
	F-function		F0		F1	
	input		f0e0cd2c		092da1b7	
	round key		370abf2d		b05e627b	
	after key add		c7ea7201		b973c3cc	
	after S		e77f9fda		174a3a46	
	after M		b9869270		8fc7e089	
Round	7	input	e52f44c9	092da1b7	c1e1140a	f0e0cd2c
	F-function		F0		F1	
	input		e52f44c9		c1e1140a	
	round key		38351b2f		74bd6ele	
	after key add		ddl5afe6		b55c7a14	
	after S		c5496150		5aa5c15c	
	after M		33d8590f		e62eb913	



Round 8	input	3af5f8b8 c1e1140a	16ce743f e52f44c9
	F-function	F0	F1
	input	3af5f8b8	16ce743f
	round key	1b7c7dce	92cfc98e
	after key add	21898576	8401bdb1
	after S	a118dc09	3949b1f3
	after M	f091202d	04f9e827
Round 9	input	31703427 16ce743f	e1d6acee 3af5f8b8
	F-function	F0	F1
	input	31703427	e1d6acee
	round key	509b31a6	4c5ad53c
	after key add	61eb0581	ad8c79d2
	after S	2a8d3304	eeffc072
	after M	f9639a90	8bebfe3d
Round 10	input	efadeeaf e1d6acee	b11e0685 31703427
	F-function	F0	F1
	input	efadeeaf	b11e0685
	round key	6fc2ba33	e1e5c878
	after key add	806f549c	50fbcefd
	after S	cd5eeb61	25d7fe02
	after M	a100e35b	26a4e16d
Round 11	input	40d64fb5 b11e0685	17d4d54a efadeeaf
	F-function	F0	F1
	input	40d64fb5	17d4d54a
	round key	419a74b9	1dd79e0e
	after key add	014c3b0c	0a034b44
	after S	49a4c013	b4c6c912
	after M	51c0208f	f1a2c339
Round 12	input	e0de260a 17d4d54a	1e0f2d96 40d64fb5
	F-function	F0	F1
	input	e0de260a	1e0f2d96
	round key	240a33d2	9dabfd09
	after key add	c4d415d8	83a4d09f
	after S	801beebe	86b8f8ed
	after M	8a9aef34	3e451646
Round 13	input	9d4e3a7e 1e0f2d96	7e9359f3 e0de260a
	F-function	F0	F1
	input	9d4e3a7e	7e9359f3
	round key	6e3ff82a	74ac3ffd
	after key add	f371c254	0a3f660e
	after S	29ea68e8	b4f530a8
	after M	17524741	4b8c607e

Round 14	input	095d6ad7 7e9359f3	ab524674 9d4e3a7e
	F-function	F0	F1
	input	095d6ad7	ab524674
	round key	b9696e2e	cc0b3a38
	after key add	b03404f9	67597c4c
	after S	152a2f03	52161e39
	after M	f7ee818b	7902f3eb
Round 15	input	897dd878 ab524674	e44cc995 095d6ad7
	F-function	F0	F1
	input	897dd878	e44cc995
	round key	ed785cbd	9c077c13
	after key add	640584c5	784bb586
	after S	459d9e10	636b5a11
	after M	4034defc	0228bdd4
Round 16	input	eb669888 e44cc995	0b75d703 897dd878
	F-function	F0	F1
	input	eb669888	0b75d703
	round key	04978d83	2ec058ba
	after key add	eff1150b	25b58fb9
	after S	90e4ee38	e7691f3b
	after M	4a678609	05b2b4a9
Round 17	input	ae2b4f9c 0b75d703	8ccf6cd1 eb669888
	F-function	F0	F1
	input	ae2b4f9c	8ccf6cd1
	round key	4bbd5f6a	31fe8de8
	after key add	e59610f6	bd31e139
	after S	f6a5286d	b15d7589
	after M	720df49d	bad65e22
Round 18	input	7978239e 8ccf6cd1	51b0c6aa ae2b4f9c
	F-function	F0	F1
	input	7978239e	51b0c6aa
	round key	b76da574	3a6fa8e7
	after key add	ce1586ea	6bdf6e4d
	after S	919c117f	283aaa43
	after M	ef24fe56	08916103
Round 19	input	63eb9287 51b0c6aa	a6ba2e9f 7978239e
	F-function	F0	F1
	input	63eb9287	a6ba2e9f
	round key	521213ce	4f1f59d8
	after key add	31f98149	e9a57747
	after S	5d03e265	3c8d7bda
	after M	b7464b63	e1d086a7

Round 20	input	e6f68dc9	a6ba2e9f	98a8a539	63eb9287
	F-function	F0		F1	
	input	e6f68dc9		98a8a539	
	round key	c13624f6		ee91f6a4	
	after key add	27c0a93f		7639539d	
	after S	20b5938b		09893194	
	after M	3cae819e		b603c454	
Round 21	input	9a14af01	98a8a539	d5e856d3	e6f68dc9
	F-function	F0		F1	
	input	9a14af01		d5e856d3	
	round key	17f68fde		f6c360a9	
	after key add	8de220df		232b367a	
	after S	6666bfff		b383a1bd	
	after M	7ae08a5d		662b2c4d	
Round 22	input	e2482f64	d5e856d3	80dda184	9a14af01
	F-function	F0		F1	
	input	e2482f64		80dda184	
	round key	6288bc72		c0ad856b	
	after key add	80c09316		407024ef	
	after S	cdb5f1e5		fbe99290	
	after M	3d9dac60		108259db	
output		e2482f64	e875fab3	80dda184	8a96f6da
final whitening key			77777777		77777777
after whitening		e2482f64	9f028dc4	80dda184	fde181ad
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
LL	477e8f09	66ee5378	2cc2be04	bf55e28f
LR	d6c10b89	4eeab575	84bd5663	cc933940
WK_{0,1,2,3}	0f0e0d0c	0b0a0908	07060504	03020100
RK_{0,1,2,3}	58f02029	15413cd0	1b0c41a4	e4bacd0f
RK_{4,5,6,7}	6c498393	8846231b	1fc716fc	7c81a45b
RK_{8,9,10,11}	fa37c259	0e3da2ee	aacf9abb	8ec0aad9
RK_{12,13,14,15}	b05bd737	8de1f2d0	8ffee0f6	b70b47ea
RK_{16,17,18,19}	581b3e34	03263f89	2f7100cd	05cee171
RK_{20,21,22,23}	b523d4e9	176d7c44	6d7ba5d7	f797b2f3
RK_{24,25,26,27}	25d80df2	a646bba2	6a3a95e1	3e3a47f0
RK_{28,29,30,31}	b304eb20	44f8824e	c7557cbc	47401e21
RK_{32,33,34,35}	d71ff7e9	aca1fb0c	2deff35d	6ca3a830
RK_{36,37,38,39}	4dd7cfb7	ae71c9f6	4e911fef	90aa95de
RK_{40,41,42,43}	2c664a7a	8cb5cf6b	14c8de1e	43b9caef
RK_{44,45,46,47}	568c5a33	07ef7ddd	608dc860	ac9e50f8
RK_{48,49,50,51}	c0c18358	4f53c80e	33e01cb9	80251e1c
plaintext	00010203	04050607	08090a0b	0c0d0e0f
initial whitening key		0f0e0d0c		0b0a0908
after whitening	00010203	0b0b0b0b	08090a0b	07070707
Round 1	input	00010203	0b0b0b0b	08090a0b 07070707
	F-function	F0		F1
	input	00010203		08090a0b
	round key	58f02029		15413cd0
	after key add	58f1222a		1d4836db
	after S	4ee41927		2c78a1ac
	after M	2db2101b		d87ee718
Round 2	input	26b91b10	08090a0b	df79e01f 00010203
	F-function	F0		F1
	input	26b91b10		df79e01f
	round key	1b0c41a4		e4bacd0f
	after key add	3db55ab4		3bc32d10
	after S	aa5afadb		0f1e1928
	after M	317e029c		c0cc96ba

Round	3	input	39770897 df79e01f c0cd94b9 26b91b10
	F-function	F0	F1
	input	39770897	c0cd94b9
	round key	6c498393	8846231b
	after key add	553e8b04	488bb7a2
	after S	5487484e	d84876a0
	after M	c3a7ac1d	7ae05884
Round	4	input	1cde4c02 c0cd94b9 5c594394 39770897
	F-function	F0	F1
	input	1cde4c02	5c594394
	round key	1fc716fc	7c81a45b
	after key add	03195afe	20d8e7cf
	after S	c607fa95	12f002c9
	after M	5edee0ce	4cfb0e90
Round	5	input	9e137477 5c594394 758c0607 1cde4c02
	F-function	F0	F1
	input	9e137477	758c0607
	round key	fa37c259	0e3da2ee
	after key add	6424b62e	7bb1a4e9
	after S	4592c8d2	46f3a044
	after M	adfd33ae	42450650
Round	6	input	f1a4703a 758c0607 5e9b4a52 9e137477
	F-function	F0	F1
	input	f1a4703a	5e9b4a52
	round key	aacf9abb	8ec0aad9
	after key add	5b6bea81	d05be08b
	after S	22285e04	f822d448
	after M	0fa52ed4	aa7a0a9c
Round	7	input	7a2928d3 5e9b4a52 34697eeb f1a4703a
	F-function	F0	F1
	input	7a2928d3	34697eeb
	round key	b05bd737	8delf2d0
	after key add	ca72ffe4	b9888c3b
	after S	23ed8e68	172b59c0
	after M	8b158630	334e2af2
Round	8	input	d58ecc62 34697eeb c2ea5ac8 7a2928d3
	F-function	F0	F1
	input	d58ecc62	c2ea5ac8
	round key	8ffee0f6	b70b47ea
	after key add	5a702c94	75e11d22
	after S	facf9d64	586f2c19
	after M	72c2027e	a582d5f0

Round 9	input	46ab7c95 c2ea5ac8	dfabfd23 d58ecc62
	F-function	F0	F1
	input	46ab7c95	dfabfd23
	round key	581b3e34	03263f89
	after key add	1eb042a1	dc8dc2aa
	after S	177afd6a	57664735
	after M	51d5740a	110287d7
Round 10	input	933f2ec2 dfabfd23	c48c4bb5 46ab7c95
	F-function	F0	F1
	input	933f2ec2	c48c4bb5
	round key	2f7100cd	05cee171
	after key add	bc4e2e0f	c142aac4
	after S	e0434cd9	22fd2380
	after M	a768d32a	b6ae4f2b
Round 11	input	78c32e09 c48c4bb5	f00533be 933f2ec2
	F-function	F0	F1
	input	78c32e09	f00533be
	round key	b523d4e9	176d7c44
	after key add	cde0fae0	e7684ffa
	after S	3fd410d4	02ef5310
	after M	08bd9b01	2fdb3f65
Round 12	input	cc31d0b4 f00533be	bce411a7 78c32e09
	F-function	F0	F1
	input	cc31d0b4	bce411a7
	round key	6d7ba5d7	f797b2f3
	after key add	a14a7563	4b73a354
	after S	1b512562	c94a71eb
	after M	7c2c762b	81ca0b59
Round 13	input	8c294595 bce411a7	f9092550 cc31d0b4
	F-function	F0	F1
	input	8c294595	f9092550
	round key	25d80df2	a646bba2
	after key add	a9f14867	5f4f9ef2
	after S	93e47852	5c26cae5
	after M	4a87c858	54bc68d5
Round 14	input	f663d9ff f9092550	988db861 8c294595
	F-function	F0	F1
	input	f663d9ff	988db861
	round key	6a3a95e1	3e3a47f0
	after key add	9c594c1e	a6b7ff91
	after S	58ff39b0	054d1d75
	after M	d82301d4	085d5025

Round 15	input	212a2484 988db861	847415b0 f663d9ff
	F-function	F0	F1
	input	212a2484	847415b0
	round key	b304eb20	44f8824e
	after key add	922ecfa4	c08c97fe
	after S	86d2c9a0	b5ff567d
	after M	dbf56073	87e2a6a2
Round 16	input	4378d812 847415b0	71817f5d 212a2484
	F-function	F0	F1
	input	4378d812	71817f5d
	round key	c7557cbc	47401e21
	after key add	842da4ae	36c1617c
	after S	9e19b889	a10c5414
	after M	6791a3e3	e177d3a8
Round 17	input	e3e5b653 71817f5d	c05df72c 4378d812
	F-function	F0	F1
	input	e3e5b653	c05df72c
	round key	d71ff7e9	acaf1b0c
	after key add	34fa41ba	6cfc0c20
	after S	d4e1be2d	32bc13bf
	after M	2743ef2d	6fec0aab
Round 18	input	56c29070 c05df72c	2c94d2b9 e3e5b653
	F-function	F0	F1
	input	56c29070	2c94d2b9
	round key	2deff35d	6ca3a830
	after key add	7b2d632d	40377a89
	after S	56193719	fb13c1b7
	after M	ee6316fa	5e3245b7
Round 19	input	2e3eeld6 2c94d2b9	bdd7f3e4 56c29070
	F-function	F0	F1
	input	2e3eeld6	bdd7f3e4
	round key	4dd7cfb7	ae71c9f6
	after key add	63e92e61	13a63a12
	after S	373c4c54	8fe6c54b
	after M	87aab08e	8f8d16f3
Round 20	input	ab3e6237 bdd7f3e4	d94f8683 2e3eeld6
	F-function	F0	F1
	input	ab3e6237	d94f8683
	round key	4e911fef	90aa95de
	after key add	e5af7dd8	49e5135d
	after S	f6ad88be	65f68f77
	after M	0889df33	f418c84f

Round 21	input	b55e2cd7 d94f8683	da262999 ab3e6237
	F-function	F0	F1
	input	b55e2cd7	da262999
	round key	2c664a7a	8cb5cf6b
	after key add	993866ad	5693e6f2
	after S	2c2b6cee	0df150e5
	after M	8999e772	da5415d2
Round 22	input	50d661f1 da262999	716a77e5 b55e2cd7
	F-function	F0	F1
	input	50d661f1	716a77e5
	round key	14c8de1e	43b9caef
	after key add	441ebfef	32d3bd0a
	after S	12b052ac	c7bbb182
	after M	f5efd89e	744a9ced
Round 23	input	2fc9f107 716a77e5	c114b03a 50d661f1
	F-function	F0	F1
	input	2fc9f107	c114b03a
	round key	568c5a33	07ef7ddd
	after key add	7945ab34	c6fbcde7
	after S	a2a77e2a	4cd7e238
	after M	e84f6d9b	ce67e20a
Round 24	input	99251a7e c114b03a	9eb183fb 2fc9f107
	F-function	F0	F1
	input	99251a7e	9eb183fb
	round key	608dc860	ac9e50f8
	after key add	f9a8d21e	322fd303
	after S	f84572b0	c7d8f1c6
	after M	20634b77	591b3f55
Round 25	input	e177fb4d 9eb183fb	76d2ce52 99251a7e
	F-function	F0	F1
	input	e177fb4d	76d2ce52
	round key	c0c18358	4f53c80e
	after key add	21b67815	3981065c
	after S	a14dd39c	c8e20aa5
	after M	3f88fbef	89ff5caf
Round 26	input	a1397814 76d2ce52	10da46d1 e177fb4d
	F-function	F0	F1
	input	a1397814	10da46d1
	round key	33e01cb9	80251e1c
	after key add	92d964ad	90ff58cd
	after S	864445ee	9a8e803f
	after M	5949235a	183d49c7



output	a1397814	2f9bed08	10da46d1	f94ab28a
final whitening key		07060504		03020100
after whitening	a1397814	289de80c	10da46d1	fa48b38a
ciphertext	a1397814	289de80c	10da46d1	fa48b38a

## Authors' Addresses

Masanobu Katagi  
System Technologies Laboratories  
Sony Corporation  
5-1-12 Kitashinagawa Shinagawa-ku  
Tokyo, 141-0001, Japan  
  
EMail: Masanobu.Katagi@jp.sony.com

Shiho Moriai  
System Technologies Laboratories  
Sony Corporation  
5-1-12 Kitashinagawa Shinagawa-ku  
Tokyo, 141-0001, Japan  
  
Phone: +81-3-5448-3701  
EMail: clefia-q@jp.sony.com

