ISO 9979/0010

ALGORITHM REGISTER ENTRY

a.) ISO entry name

{iso standard 9979 feal (10)}

b.) Proprietary entry name

FEAL : the Fast Data Encipherment Algorithm. One subset of FEAL, i.e., N round FEAL with 64-bit key, is called FEAL-N, and another subset of FEAL, i.e., N round FEAL with 128-bit key, is called FEAL-NX.

c.) Intended range of applications

1. Confidentiality
2. Authentication : as detailed in ISO 9798.
3. Date Integrity : as detailed in ISO 9797.
4. Hash Function : as detailed in ISO 10118 Part 2 for FEAL-N.

d.) Cryptographic interface parameters

FEAL has five interface parameters in which 1, 2 and 3 below are options selected by users.

1. Key length - which is 64 bits or 128 bits.
2. Round number (N) - which determines the round number (N) for FEAL data randomization, where $N \geq 8$ and even.
3. Key parity - The key may contain parity bits as an option (one bit per byte of Key).
4. Input size : 64 bits (fixed)
5. Output size : 64 bits (fixed)

e.) Test words

1. FEAL-8 without key parity bits

(1) Key:	0123 4567 89AB CDEF
(2) Input data:	0000 0000 0000 0000
(3) Output data:	CEEF 2C86 F249 0752

2. FEAL-16 without key parity bits

(1) Key:	0123 4567 89AB CDEF
(2) Input data :	0000 0000 0000 0000
(3) Output data:	3ADE 0D2A D84D 0B6F

3. FEAL-32 without key parity bits

(1) Key:	0123 4567 89AB CDEF
(2) Input data :	0000 0000 0000 0000
(3) Output data:	69B0 FAE6 DDED 6B0B

4. FEAL-8X without key parity bits

(1) Key:	0123 4567 89AB CDEF
	0123 4567 89AB CDEF

(2) Input data :	0000 0000 0000 0000
(3) Output data:	9616 57B2 BA41 31BD

5. FEAL-16X without key parity bits

(1) Key:	0123 4567 89AB CDEF
	0123 4567 89AB CDEF
(2) Input data :	0000 0000 0000 0000
(3) Output data:	F418 0986 EC9A 2E1C

6. FEAL-32X without key parity bits

(1) Key:	0123 4567 89AB CDEF
	0123 4567 89AB CDEF
(2) Input data :	0000 0000 0000 0000
(3) Output data:	70D6 E684 9CAD 2754

7. FEAL-8 with key parity bits

(1) Key:	0123 4567 89AB CDEF
(2) Input data :	0000 0000 0000 0000
(3) Output data:	6A72 2D1C 46B3 9336

8. FEAL-16 with key parity bits

(1) Key:	0123 4567 89AB CDEF
(2) Input data :	0000 0000 0000 0000
(3) Output data:	86BF 0D9D 6255 B4FF

9. FEAL-32 with key parity bits

(1) Key:	0123 4567 89AB CDEF
(2) Input data :	0000 0000 0000 0000
(3) Output data:	010A 53D8 E2B3 E965

10. FEAL-8X with key parity bits

(1) Key:	0123 4567 89AB CDEF
	0123 4567 89AB CDEF
(2) Input data :	0000 0000 0000 0000
(3) Output data:	0009 C4D9 C6E6 DBB3

11. FEAL-16X with key parity bits

(1) Key:	0123 4567 89AB CDEF
	0123 4567 89AB CDEF
(2) Input data :	0000 0000 0000 0000
(3) Output data:	486D 38EB 76C2 4B83

12. FEAL-32X with key parity bits

(1) Key:	0123 4567 89AB CDEF
	0123 4567 89AB CDEF
(2) Input data :	0000 0000 0000 0000
(3) Output data:	C95D 7957 0410 D96C

f.) The identity of the organization

Sponsoring Authority

ISO 9979/0010

INFORMATION-TECHNOLOGY PROMOTION AGENCY, JAPAN (IPA)

Registration Requested by
NIPPON TELEGRAPH AND TELEPHONE CORPORATION

Contact for Information
Secretariat for FEAL Registration,
NTT Information and Communication Systems Laboratories,
1-2356 Take, Yokosuka-shi, Kanagawa, Japan
zip code: 238-03,
Telephone number +81 468-59-3377
Fax number +81 468-59-3858

g.) Dates of registration and modifications 14 November 1994

h.) Whether the subject of a National Standard : No.

i.) Patent license restrictions

1. US Patent number 4,850,019 'Data randomization equipment', registered in 1989
2. New European Application number, 86115279.1-2209, 'Data randomization equipment' filed in 1986, for France, Germany and UK.
3. Japan No. 60-250398, filing date 1985
4. Japan No. 60-252650, filing date 1985
5. Japan No. 62-37231, filing date 1987
6. Japan No. 62-232957, filing date 1987
7. Japan No. 62-308463, filing date 1987
8. Japan No. 1-340384, filing date 1989

Note: 3 and 4 are owned by NTT and NTT-Data Corp., but others are by only NTT.

j.) References

(j-1) S. Miyaguchi: The FEAL Cipher Family, pp. 627-638, Advances in Cryptology - CRYPTO'90, LNCS 537, Springer-Verlag, 1991
(j-2) S. Miyaguchi, S. Kurihara, K. Ohta and H. Morita: Expansion of FEAL Cipher, pp. 117-127, Vol. 2, No. 6, NTT REVIEW, November 1990

k.) Description of Algorithm

FEAL has three options: key length, round number and key parity. The key length selects either a 64 bit key or 128 bit key, the round number (N) specifies the internal iteration number for data randomization, and the key parity option selects either the use or non-use of parity bits in the key block. The exact details of the FEAL algorithm are described in the Annex of this document.

l.) Modes of operation

See ISO 8372 for information on modes of operation.

m.) Other information

m-1.) Selection of key length
With current LSI technologies, an exhaustive key search for a 64-bit key cipher (FEAL-N) seems impossible now, but not one or two decades later. So, a 64-bit key (FEAL-N) is now recommended, but later a 128-bit key (FEAL-NX) will be recommended. Details are discussed in the document 'The FEAL Cipher Family' (See document j-1).

m-2.) Selection of Round number (N)
Where attacks that use pairs of plaintext and their ciphertext block (such as differential attacks, or linear attacks) are not a threat, small N is useful to achieve higher processing speeds, and where these attacks are a threat, the value of N should be increased.

m-3) Parity bit
Parity bits in the key-block of FEAL can be selected for hardware implementation where parity bit error correction is useful. The effective key length becomes 56 bits for FEAL-N and 112 bits for FEAL-NX. If error correction is not so important, no-parity bit option is recommended to maximize the effective key length, i.e. 64 bits for FEAL-N and 128 bits for FEAL-NX.

m-4) Upward Compatibility
FEAL-NX is upwardly compatible with FEAL-N because the right half of the 128-bit key of FEAL-NX is all zero, FEAL- NX works as FEAL-N. This guarantees that FEAL-NX hardware/software modules support the FEAL-N protocol.

m-5) FEAL LSI chips
FEAL has been implemented as below.
(1) 1988, FEAL-8 LSI with/without key parity, with ECB/CBC/CFB-1, 1.5-μm CMOS, which is available as NLC5001F.
(2) 1990, FEAL-8 LSI without key parity, with ECB/CBC/CFB-8/OFB-8, 1.0-μm CMOS, with full duplex.
(3) 1993, FEAL-8/-32 LSI without key parity, with ECB/CBC/CFB-8/OFB-8, 0.5-μm CMOS, with full duplex.

m-6) Examples of enciphering speed with software implementation

It has been confirmed that :
(1) In the case of 32-bit CPUs, i.e. Sun SPARC 10 (f=40MHz). FEAL-8 program written in C language runs at 9 Mbps, and FEAL-32 program at 3 Mbps, table search techniques are not used.
(2) In the case of 8-bit CPUs for smart cards, i.e. H8/310 (f=4.9MHz). FEAL-8 program written in assembler runs at 300 kbps, The FEAL-32 program achieves 100 kbps. Program size is about 500 bytes for each program.

Annex

FEAL SPECIFICATIONS

1 Introduction

1.1 Outline of the FEAL cipher

FEAL , the Fast Data Encipherment Algorithm, is a 64-bit block cipher algorithm that enciphers 64-bit plaintexts into 64-bit ciphertexts and vice versa with either a 64-bit or 128-bit key.

FEAL has three options: key length, round number and key parity. The key length selects either 64-bit key or 128-bit key, the round number (N) specifies the internal iteration number for data randomization, and the key parity option selects either use or non-use of the parity bits in a key block.

One subset of FEAL, called FEAL-N, is N round FEAL with 64-bit key. FEAL-NX is also another subset and is described as N round FEAL with 128-bit key. When the right half of the FEAL-NX 128-bit key is all zero, FEAL-NX is equivalent to FEAL-N.

1.2 FEAL options

FEAL options are defined below.

(1) Key length

Selects either 64-bit key or 128-bit key for FEAL key schedule.

(2) Round number (N)

Determines the round number (N) for FEAL data randomization, where $N \geq 8$ and even.

(3) Key parity

Indicates:

Use of key parity bits in a key-block, or

Non-use of key parity bits in a key-block

1.3 Definitions and explanations

1.3.1 Definitions

(1) key-block: Either a 64-bit block or 128-bit , which includes key parity bits only when the key parity option indicates " use of key parity bits ".

(2) key parity bits: odd parity bits for each 8-bit data in a key-block.

(3) key-block with no parity bits: a key-block that does not include parity bits.

(4) key-block with parity bits: a key-block that includes key parity bits. Parity bit positions are $8 \times i$ where i=1, 2, 3, .., 16.

(5) key: Key information used for enciphering/deciphering.

(6) round number (N): the internal iteration number for FEAL data randomization.

(7) extended key: 16-bit blocks, K_i, which are a randomized and extended form of the key, are output from FEAL key schedule, where i=0, 1, ..., (N+7).

1.3.2 Conventions and Notations

(1) A, Ar,.... : blocks

(2) (A, B,....) : concatenation in this order

(3) A$\oplus$ B: exclusive-or operation of A and B

(4) ϕ : zero block, 32-bits long

(5) =: Transfer from right side to left side

(6) Bit position: 1, 2, 3,.... count from the first left side bit (MSB) in a block towards the right.

2 Enciphering algorithm

2.1 Computation stages

The extended key K_i used in this enciphering procedure is generated by the key schedule described in clause 4. Similarly, function f used here is defined in clause 5. The computation stages, specified more fully in subclauses 2.2 to 2.4, shall be as follows (see also Figure 1):

a) Pre-processing (see 2.2)

b) Iterative calculation (see 2.3)

c) Post-processing (see 2.4)

2.2 Pre-processing

Plaintext P is separated into L_0 and R_0 of equal lengths (32 bits), i.e., $P=(L_0,R_0)$.

First,

$$(L_0, R_0) = (L_0, R_0) \oplus (K_N, K_{N+1}, K_{N+2}, K_{N+3})$$

Next,

$$(L_0, R_0) = (L_0, R_0) \oplus (\phi, L_0)$$

2.3 Iterative calculation

Input (L_0, R_0), and calculate the equations below for r from 1 to N iteratively,

$$R_r = L_{r-1} \oplus f(R_{r-1}, K_{r-1})$$
$$L_r = R_{r-1}$$

Output of the final round is (L_N, R_N).

2.4 Post-processing

Interchange the final output of the iterative calculation, (L_N, R_N), into (R_N, L_N).
Next calculate:

$$(R_N , L_N) = (R_N , L_N) \oplus (\phi , R_N)$$

Lastly,

$$(R_N , L_N) = (R_N, L_N) \oplus (K_{N+4}, K_{N+5}, K_{N+6}, K_{N+7})$$

Ciphertext is given as (R_N, L_N).

3 Deciphering algorithm

3.1 Computation stages

The extended key K_i used in this deciphering procedure is generated by the key schedule described in clause 4. Similarly, function f used here is defined in clause 5. The computation stages, specified more fully in subclauses 3.2 to 3.4, shall be as follows (see also Fig. 1):

a) Pre-processing (see 3.2)

b) Iterative calculation (see 3.3)

c) Post-processing (see 3.4)

3.2 Pre-processing

Ciphertext (R_N, L_N) is separated into R_N and L_N of equal lengths.

First,

$$(R_N , L_N) = (R_N, L_N) \oplus (K_{N+4}, K_{N+5}, K_{N+6}, K_{N+7})$$

Next,

$$(R_N , L_N) = (R_N, L_N) \oplus (\phi , R_N)$$

3.3 Iterative calculation

Input (R_N , L_N), and calculate the equations below for r from N to 1 iteratively,

$$L_{r-1} = R_r \oplus f\,(L_r, K_{r-1})$$
$$R_{r-1} = L_r$$

Output of the final round is (R_0, L_0).

3.4 Post-processing

Change the final output of the iterative calculation, (R_0, L_0), into (L_0, R_0).
Next calculate:

$$(L_0, R_0) = (L_0, R_0) \oplus (\phi, L_0)$$

Lastly,

$$(L_0, R_0) = (L_0, R_0) \oplus (K_N, K_{N+1}, K_{N+2}, K_{N+3})$$

Plaintext is given as (L_0, R_0).

4 Key schedule

4.1 Key schedule of FEAL-NX

First , the key schedule of FEAL-*NX* is described (see also Fig. 2), where the functions used here are defined in Clause 5. The key schedule yields the extended key Ki $(i=0, 1, 2, 3..., N+7)$ from the 128-bit key.

4.1.1 Definition of left key K_L and right key K_R

Input 128-bit key is equally divided into a 64-bit left key, K_L, and a 64-bit right key, K_R. (K_L, K_R) is the inputted 128-bit key.

4.1.2 Parity bit processing

(1) Non-use of key parity bits: There is no processing .

(2) Use of key parity bits: Bit positions, 8, 16, 24, 32, 40, 48, 56, 64 of both K_L and K_R are set to zeros, i.e., all parity bits in the key block are set to zero.

Note: How to use parity bits is outside the scope of the FEAL.

4.1.3 Iterative calculation

(1) Processing of the right key K_R

K_R is divided into left K_{R1} and right half (K_{R2}), (i. e., $K_R = (K_{R1}, K_{R2})$) and the temporary variable, Q_r, is defined as:

$$Q_r = K_{R1} \oplus K_{R2} \quad \text{for } r = 1,\ 4,\ 7..., \quad (r = 3i+1;\ i = 0, 1, ...)$$

$$Q_r = K_{R1} \quad \text{for } r = 2, 5, 8..., \quad (r = 3i+2;\ i = 0, 1, ...)$$

$$Q_r = K_{R2} \quad \text{for } r = 3, 6, 9..., \quad (r = 3i+3;\ i = 0, 1, ...)$$

where $1 \le r \le (N/2)+4$, ($N \ge 8$, N: even).

Note: For FEAL-N, $K_R = (\phi, \phi)$ (64 zeros) and $Q_r = \phi$, (32 zeros).

(2) Processing of the left key K_L

Let A_0 be the left half of K_L and let B_0 be the right half, i.e., $K_L = (A_0, B_0)$. Set $D_0 = \phi$, then calculate K_i (i=0 to N+7) for r =1 to (N/2)+4.

$$\begin{aligned}
D_r &= A_{r-1} \\
A_r &= B_{r-1} \\
B_r &= f_K(\alpha, \beta) \\
&= f_K(A_{r-1}, (B_{r-1} \oplus D_{r-1}) \oplus Q_r)) \\
K_{2(r-1)} &= (B_{r0}, B_{r1}) \\
K_{2(r-1)+1} &= (B_{r2}, B_{r3})
\end{aligned}$$

A_r, B_r, D_r and Q_r are auxiliary variables, where $B_r = (B_{r0}, B_{r1}, B_{r2}, B_{r3})$, B_{rj} 8 - bits long.

4.2 Key schedule of FEAL-N

When the right half (64-bit) of FEAL-NX 128-bit key is all zero, FEAL (i. e., FEAL-NX) works as FEAL-N. Then, the temporary variable $Q_r = \phi$, the iterative calculation is given as follows:

4.2.1 Parity bit processing

(1) Non-use of key parity bits: There is no processing .

(2) Use of key parity bits: Bit positions, 8, 16, 24, 32, 40, 48, 56, 64 of key block is set to zeros, i.e., all parity bits in the key block are set to zero.

Note: How to use parity bits is outside the scope of the FEAL.

4.2.2 Iterative calculation

Let A_0 be the left half of the 64-bit key and let B_0 be the right, i.e., the 64-bit key = (A_0, B_0) and $D_0 = \phi$.

Then calculate K_i (i=0 to N+7) for r = 1 to (N/2)+4, ($N \ge 8$, N: even).

$$\begin{aligned}
D_r &= A_{r-1} \\
A_r &= B_{r-1} \\
B_r &= f_K(\alpha, \beta) = f_K(A_{r-1}, (B_{r-1} \oplus D_{r-1})) \\
K_{2(r-1)} &= (B_{r0}, B_{r1}) \\
K_{2(r-1)+1} &= (B_{r2}, B_{r3})
\end{aligned}$$

A_r, B_r, D_r and Q_r are auxiliary variables, where $B_r = (B_{r0}, B_{r1}, B_{r2}, B_{r3})$, B_{rj} 8 - bits long.

The key schedule of FEAL-N is shown in Fig. 3.

5 Functions

This clause describes functions used in clauses 2, 3 and 4.

5.1 Function f (see also Fig. 4)

$f(\alpha, \beta)$ is shortened to f. α and β are divided as follows, where α_i, and β_i are 8-bits long.

Functions S_0 and S_1 are defined in clause 5.3.

$$\alpha = (\alpha_0, \alpha_1, \alpha_2, \alpha_3),\ \beta = (\beta_0, \beta_1).$$

$f=(f_0, f_1, f_2, f_3)$ are calculated in sequence.

$$f_1 = \alpha_1 \oplus \beta_0$$
$$f_2 = \alpha_2 \oplus \beta_1$$
$$f_1 = f_1 \oplus \alpha_0$$
$$f_2 = f_2 \oplus \alpha_3$$
$$f_1 = S_1(f_1, f_2)$$
$$f_2 = S_0(f_2, f_1)$$
$$f_0 = S_0(\alpha_0, f_1)$$
$$f_3 = S_1(\alpha_3, f_2)$$

Example in hex:

Inputs: α = `00 FF FF 00`, β = `FF FF`, Output: f = `10 04 10 44`

5.2 Function f_K (see also Fig. 5)

Inputs of function f_K, α and β, are divided into four 8-bit blocks as:

$$\alpha = (\alpha_0, \alpha_1, \alpha_2, \alpha_3),\quad \beta = (\beta_0, \beta_1, \beta_2, \beta_3).$$

$f_K(\alpha, \beta)$ is shortened to f

$f_K=(f_{K0}, f_{K1}, f_{K2}, f_{K3})$ are calculated in sequence, where functions S_0 and S_1 are defined in clause5.3.

$$f_{K1} = \alpha_1 \oplus \alpha_0$$
$$f_{K2} = \alpha_2 \oplus \alpha_3$$
$$f_{K1} = S_1(f_{K1}, (f_{K2} \oplus \beta_0))$$
$$f_{K2} = S_0(f_{K2}, (f_{K1} \oplus \beta_1))$$
$$f_{K0} = S_0(\alpha_0, (f_{K1} \oplus \beta_2))$$
$$f_{K3} = S_1(\alpha_3, (f_{K2} \oplus \beta_3))$$

Example in hex:

Inputs: α = 00 00 00 00, β = 00 00 00 00, f_K = 10 04 10 44

5.3 Function S

S_0 and S_1 are defined as follows:

$$S_0(X_1, X_2) = Rot2((X_1 + X_2) \bmod 256)$$
$$S_1(X_1, X_2) = Rot2((X_1 + X_2 + 1) \bmod 256)$$

where X_1 and X_2 are 8-bit blocks and Rot2(T) is the result of a 2-bit left rotation operation on 8-bit block, T.

Example: Suppose X_1 = 00010011, X_2 = 11110010 then

$T = (X_1 + X_2 + 1) \bmod 256$ = 00000110

$S_1(X_1, X_2) = Rot2(T)$ = 00011000

6 Example of working data

Working data are shown in bit sequence and in hexadecimal (hex).

6.1 FEAL options (see clause 1.2)

In this example following FEAL options are selected:

(1) Key length: 64 bits

(2) Round number: N=16

(3) Non-use of key parity bit

6.2 Input data

Input data are the key-block and the plaintext block.

The key-block K is given :

K = 0000 0001 0010 0011 0100 0101 0110 0111
1000 1001 1010 1011 1100 1101 1110 1111 in bit sequence

K = 01 23 45 67 89 AB CD EF in hex

The plaintex P is :

P = 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 in bit sequence

P = 00 00 00 00 00 00 00 00 in hex

6.3 The key schedule (see clause 4)

Consider first the generation of the extended keys, K_0, K_1, K_2,, K_{23}, each consisting of 16 bits generated from the key-block K.

6.3.1 Parity bit processing (see 4.2)

As non-use of key parity bits is selected, there is no special processing here.

6.3.2 Iterative calculation (see 4.3)

Let A_0 be the left half of K_L and let B_0 be the right half of K_L, i.e.,

$K_L = A_0 \| B_0$ and $D_0 = \phi$. Thus:

A_0 = 0000 0001 0010 0011 0100 0101 0110 0111 in bit sequence

= 01 23 45 67 in hex

B_0 = 1000 1001 1010 1011 1100 1101 1110 1111 in bit sequence

= 89 AB CD EF in hex

D_0 = 0000 0000 0000 0000 0000 0000 0000 0000 in bit sequence

= 00 00 00 00 in hex

Calculate D_1, A_1, B_1, K_0 and K_1 as :

$D_1 = A_0$ = 0000 0001 0010 0011 0100 0101 0110 0111 in bit sequence

= 01 23 45 67 in hex

$A_1 = B_0$ = 1000 1001 1010 1011 1100 1101 1110 1111 in bit sequence

= 89 AB CD EF in hex

$B_1 = f_K(A_0, B_0 \oplus D_0)$

= 1101 1111 0011 1011 1100 1010 0011 0110 in bit sequence

= DF 3B CA 36 in hex

K_0 = 1101 1111 0011 1011 in bit sequence

= DF 3B in hex

K_1 = 1100 1010 0011 0110 in bit sequence

= CA 36 in hex

If this procedure is continued it will be found that the extended key K_i is given in hex by:

K_0 = DF 3B	K_1 = CA 36	K_2 = F1 7C	K_3 = 1A EC
K_4 = 45 A5	K_5 = B9 C7	K_6 = 26 EB	K_7 = AD 25
K_8 = 8B 2A	K_9 = EC B7	K_{10} = AC 50	K_{11} = 9D 4C
K_{12} = 22 CD	K_{13} = 47 9B	K_{14} = A8 D5	K_{15} = 0C B5
K_{16} = 53 23	K_{17} = 0A 3E	K_{18} = F3 19	K_{19} = C2 74
K_{20} = 82 83	K_{21} = 0E 38	K_{22} = 01 49	K_{23} = 7E B7

6.4 The Enciphering algorithm (see clause 2)

6.4.1 Pre-processing (see 2.2)

P = 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 in bit sequence
P = 00 00 00 00 00 00 00 00 in hex

P is separated into L_0 and R_0 both 32-bits long.
First,

$(L_0, R_0) = (L_0, R_0) \oplus (K_{16}, K_{17}, K_{18}, K_{19})$
= 0101 0011 0010 0011 0000 1010 0011 1110
1111 0011 0001 1001 1100 0010 0111 0100 in bit sequence
= 53 23 0A 3E F3 19 C2 74 in hex

Next,

$(L_0, R_0) = (L_0, R_0) \oplus (\phi, L_0)$
= 0101 0011 0010 0011 0000 1010 0011 1110
1010 0000 0011 1010 1100 1000 0100 1010 in bit sequence
= 53 23 0A 3E A0 3A C8 4A in hex

Output of this processing stage is:

L_0 = 0101 0011 0010 0011 0000 1010 0011 1110 in bit sequence
= 53 23 0A 3E in hex
R_0 = 1010 0000 0011 1010 1100 1000 0100 1010 in bit sequence
= A0 3A C8 4A in hex

6.4.2 Iterative calculation (see 2.3)

6.4.2.1 Calculation of R_0 and L_0 at the first stage

First, calculate $f(R_0, K_0)$ as:

$f(R_0, K_0)$

= `0111 1110 1111 1111 1110 0010 1011 0100` in bit sequence

= `7E FF E2 B4` in hex

Where details are described in clause 6.4.2.2.

$L_0 \oplus f(R_0, K_0) =$ `2D DC E8 8A` in hex

Output of first stage of the iterative calculation is:

$L_1 = R_0 =$ `1010 0000 0011 1010 1100 1000 0100 1010` in bit sequence

= `A0 3A C8 4A` in hex

$R_1 =$ `0010 1101 1101 1100 1110 1000 1000 1010` in bit sequence

= `2D DC E8 8A` in hex

6.4.2.2 Calculation f of the first stage

In the calculation of $f(R_0, K_0)$, shown below, $f(R_0, K_0)$ is shortened to f, and α and β are defined as:

$\alpha = (\alpha_0, \alpha_1, \alpha_2, \alpha_3) = R_0$

= `1010 0000 0011 1010 1100 1000 0100 1010` in bit sequence

= `A0 3A C8 4A` in hex

$\beta = (\beta_0, \beta_1) = K_0$

= `1101 1111 0011 1011` in bit sequence

= `DF 3B` in hex

α_0 = `1010 0000` = `A0` in hex, α_1 = `0011 1010` = `3A` in hex

α_2 = `1100 1000` = `C8` in hex, α_3 = `0100 1010` = `4A` in hex

β_0 = `1101 1111` = `DF` in hex, β_1 = `0011 1011` = `3B` in hex

$f = (f_0, f_1, f_2, f_3)$ are calculated by the sequence:

$f_1 = \alpha_1 \oplus \beta_0 =$ `1110 0101` = `E5` in hex

$f_2 = \alpha_2 \oplus \beta_1 =$ `1111 0011` = `F3` in hex

$f_1 = f_1 \oplus \alpha_0 =$ `0100 0101` = `45` in hex

$f_2 = f_2 \oplus \alpha_3 =$ `1011 1001` = `B9` in hex

$f_1 = S_1(f_1, f_2) =$ `1111 1111` = `FF` in hex

$f_2 = S_0(f_2, f_1) =$ `1110 0010` = `E2` in hex

$f_0 = S_0(\alpha_0, f_1) =$ `0111 1110` = `7E` in hex

$f_3 = S_1(\alpha_3, f_2) =$ `1011 0100` = `B4` in hex

6.4.2.3 Continued calculations

If the above calculations are continued it will be found that L_i and R_i etc. are as given in hex.

The Process stages

i	L_i	R_i	K_{i-1}	$f(R_{i-1}, K_i)$
0	53 23 0A 3E	A0 3A C8 4A		
1	A0 3A C8 4A	2D DC E8 8A	DF3B	7E FF E2 B4
2	2D DC E8 8A	1D 78 92 DD	CA36	BD 42 5A 97
3	1D 78 92 DD	2C FF B1 56	F17C	01 23 59 DC
4	2C FF B1 56	13 2F 1B 5E	1AEC	0E 57 89 83
5	13 2F 1B 5E	DD 96 94 44	45A5	F1 69 25 12
6	DD 96 94 44	07 07 E7 5B	B9C7	14 28 FC 05
7	07 07 E7 5B	DD 6F D5 32	26EB	00 F9 41 76
8	DD 6F D5 32	A6 8C D2 FA	AD25	A1 8B 35 A1
9	A6 8C D2 FA	3D FD 87 07	8B2A	E0 92 52 35
10	3D FD 87 07	9D 1D F1 56	ECB7	3B 91 23 AC
11	9D 1D F1 56	89 6D 99 D2	AC50	B4 90 1E D5
12	89 6D 99 D2	A3 1B C5 4A	9D4C	3E 06 34 1C
13	A3 1B C5 4A	E1 1A 7F 16	22CD	68 77 E6 C4
14	E1 1A 7F 16	DD A5 07 2D	479B	7E BE C2 67
15	DD A5 07 2D	61 59 76 CA	A8D5	80 43 09 DC
16	61 59 76 CA	B8 5D 03 12	0CB5	65 F8 04 3F

6.4.3 Post processing (see 2.4)

First, interchanging L_{16} and R_{16} yields:

(R_{16}, L_{16}) = 1011 1000 0101 1101 0000 0011 0001 0010
0110 0001 0101 1001 0111 0110 1100 1010 in bit sequence
= B8 5D 03 12 61 59 76 CA in hex.

Next,

$$(R_{16}, L_{16}) = (R_{16}, L_{16}) \oplus (\phi, R_{16})$$

(R_{16}, L_{16}) = 1011 1000 0101 1101 0000 0011 0001 0010
1101 1001 0000 0100 0111 0101 1101 1000 in bit sequence
= B8 5D 03 12 D9 04 75 D8 in hex.

Lastly,

$$(R_{16}, L_{16}) = (R_{16}, L_{16}) \oplus (K_{20}, K_{21}, K_{22}, K_{23})$$

= 0011 1010 1101 1110 0000 1101 0010 1010
1101 1000 0100 1101 0000 1011 0110 1111 in bit sequence
= 3A DE 0D 2A D8 4D 0B 6F in hex.

Ciphertext is given as (R_{16}, L_{16}).

The final result (ciphertext) is :

C = 0011 1010 1101 1110 0000 1101 0010 1010
1101 1000 0100 1101 0000 1011 0110 1111 in bit sequence
= 3A DE 0D 2A D8 4D 0B 6F in hex.

1509979/0010.

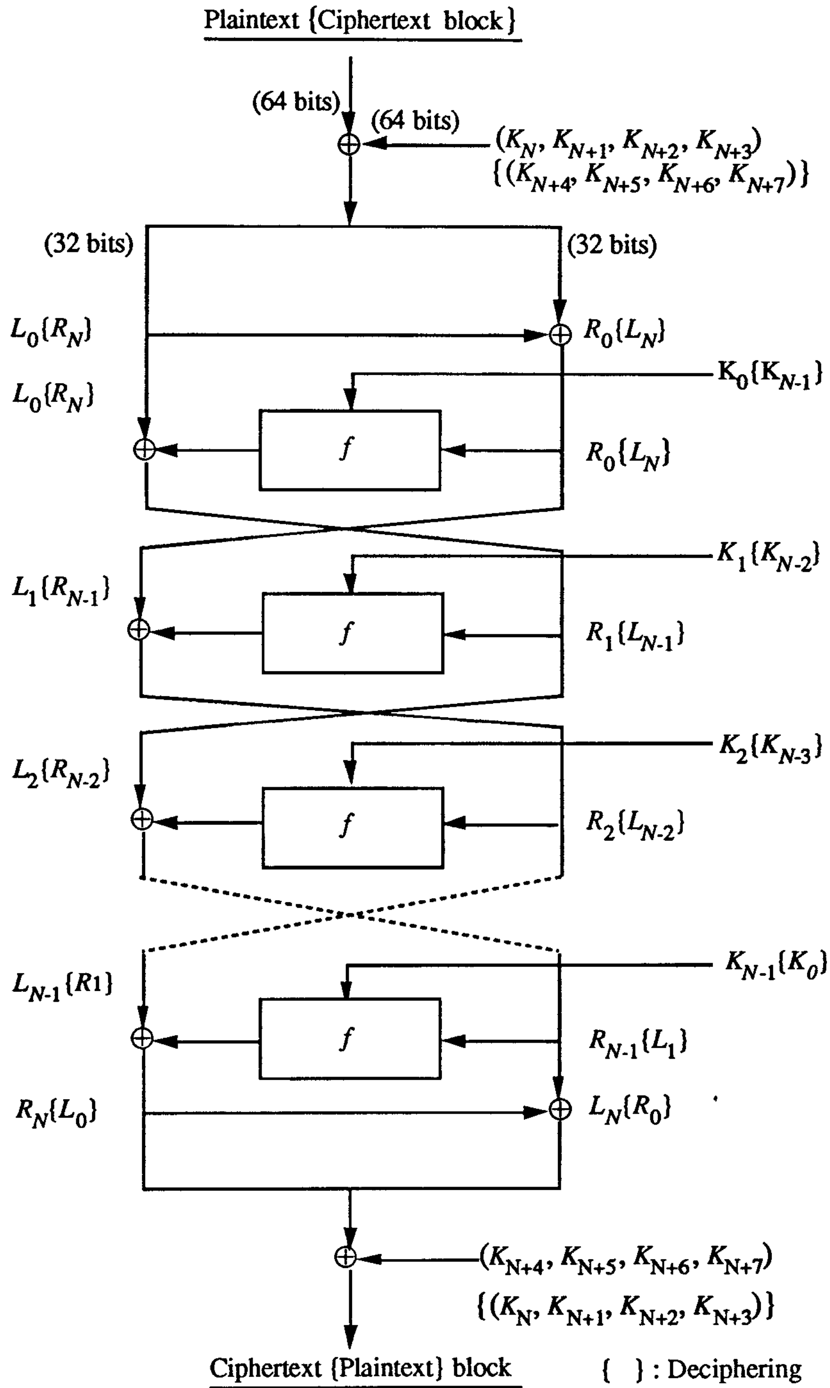
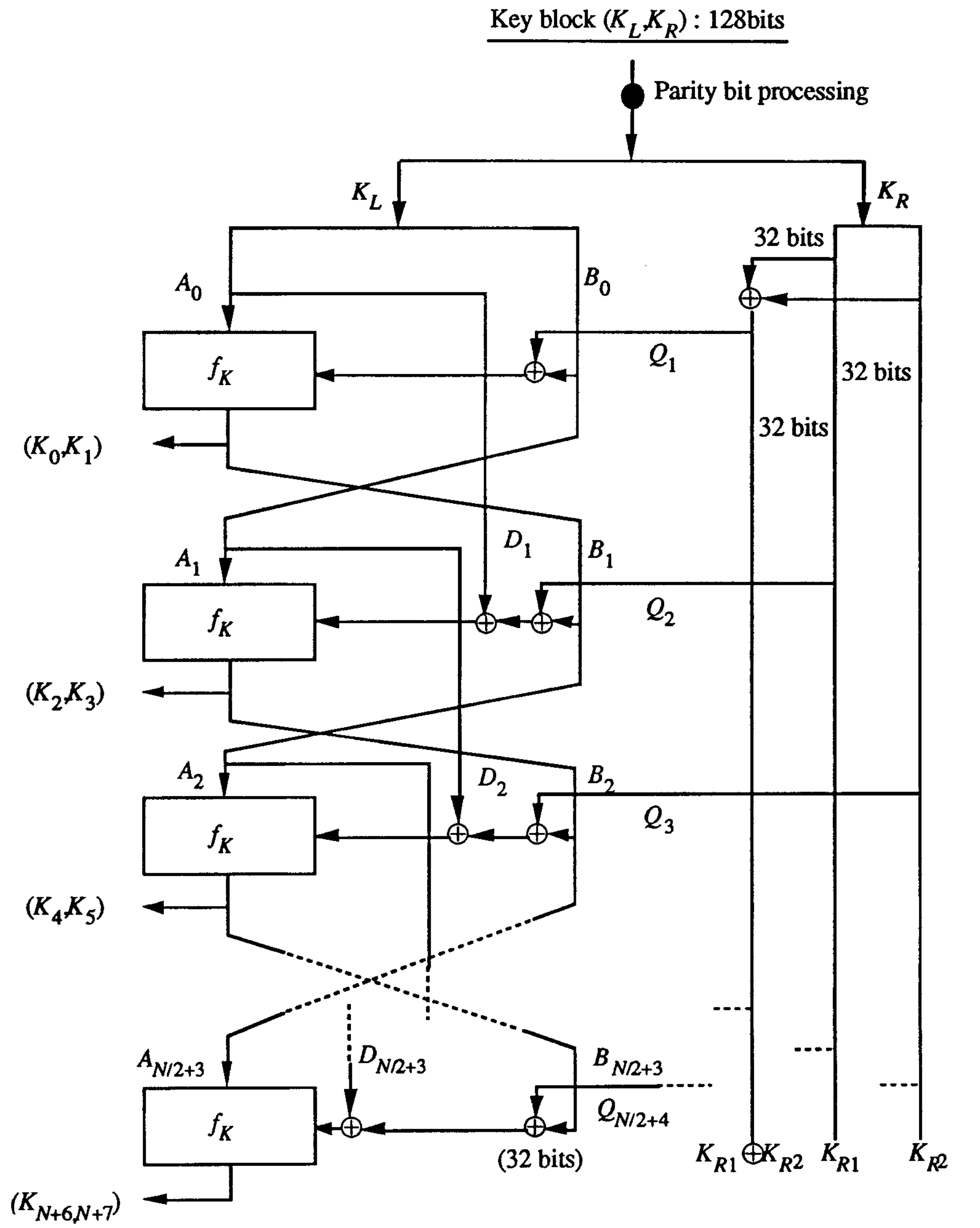


Fig. 1 Data Randomization of FEAL

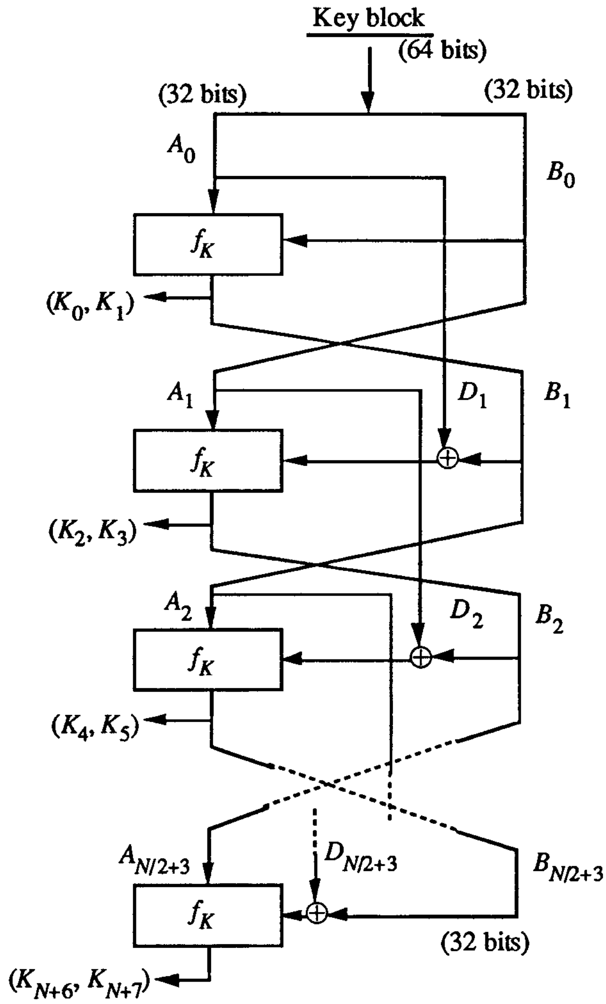


$K_{2(r-1)}$: Left half of B_r (16 bits)
$K_{2(r-1)+1}$: Right half of B_r (16 bits)
Number of iterations is $(N/2) + 4$.

$Q_r = K_{R1} \oplus K_{R2}$, $r = 1, 4, 7, \cdots$
$Q_r = K_{R1}$, $r = 2, 5, 8, \cdots$
$Q_r = K_{R2}$, $r = 3, 6, 9, \cdots$
$K_R = (K_{R1}, K_{R2})$

Fig. 2 Key Schedule of FEAL (FEAL-*NX*)

ISO 9979/0010

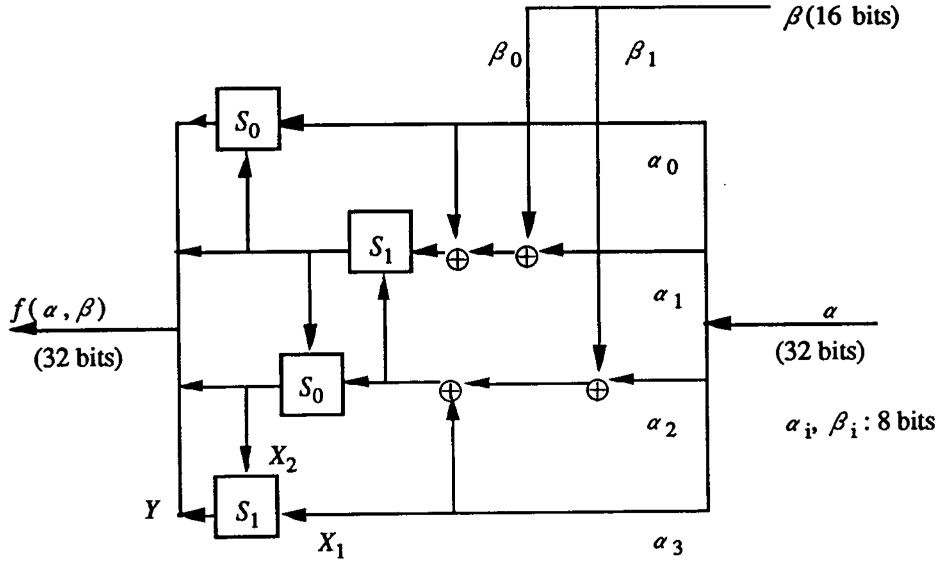


$K_{2(r-1)}$: Left half of Br (16 bits)
$K_{2(r-1)+1}$: Right half of Br (16 bits)
Number of iterations is $(N/2) + 4$.

Fig. 3 Key Schedule of FEAL (FEAL-N)

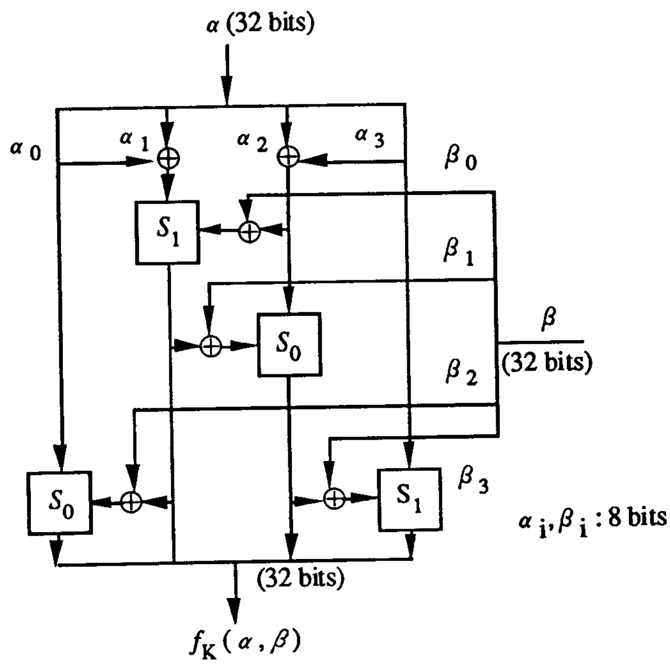
[redacted]

IS09979/0010



$Y = S_0(X_1, X_2) = Rot2\ ((X_1+X_2) \bmod 256)$
$Y = S_1(X_1, X_2) = Rot2\ ((X_1+X_2+1) \bmod 256)$
Y : output (8 bits), X_1 / X_2 : inputs (8 bits),
Rot2 (Y) : a 2-bit left rotation on 8-bit data Y

Fig. 4 f_K-function of FEAL



Note : S_0/S_1 are the same as S_0/S_1 in f-function.

Fig. 5 f_K-function of FEAL