FEAL-NX SPECIFICATIONS

1 Introduction

1.1 Outline of the FEAL-NX 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.

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 the use or non-use of parity bits in a key block.

One subset of FEAL, called FEAL-NX, is N round FEAL using a 128-bit key without key parity.

1.2 FEAL-NX options

FEAL-NX options are defined below.
(1) Round number (N)

Determines the round number (N) for FEAL data randomization, where $N \geq 32$ and even.

1.3 Definitions and explanations

1.3.1 Definitions

(1) key-block: 128-bit.

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

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

(4) 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, A_r,... : blocks (The lengths of blocks are defined in each section)

(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., $(L_0,R_0)=P$.

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 K_i (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 Iterative calculation

(1) Processing of the right key K_R

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

$$Q_r = K_{R1} \oplus K_{R2} \qquad \text{for } r = 1, 4, 7..., \qquad (r = 3i+1;\ i = 0, 1, ...)$$
$$Q_r = K_{R1} \qquad \text{for } r = 2, 5, 8..., \qquad (r = 3i+2;\ i = 0, 1, ...)$$
$$Q_r = K_{R2} \qquad \text{for } r = 3, 6, 9..., \qquad (r = 3i+3;\ i = 0, 1, ...)$$
$$\text{where } 1 \leqq r \leqq (N/2)+4,\ (N \geqq 32,\ N\text{: even}).$$

(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., $(A_0, B_0)=K_L$. 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 $(B_{r0}, B_{r1}, B_{r2}, B_{r3}) = B_r$, B_{rj} (j=0 to 3) 8-bits long, and r = 1 to (N/2)+4.

5 Functions

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

5.1 Function f (see also Fig. 3)

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

Functions S_0 and S_1 are defined in clause 5.3.

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

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

$$
\begin{aligned}
f_1 &= \alpha_1 \oplus \beta_0 \\
f_2 &= \alpha_2 \oplus \beta_1 \\
f_1 &= f_1 \oplus \alpha_0 \\
f_2 &= f_2 \oplus \alpha_3 \\
f_1 &= S_1 (f_1, f_2) \\
f_2 &= S_0 (f_2, f_1) \\
f_0 &= S_0 (\alpha_0, f_1) \\
f_3 &= S_1 (\alpha_3, f_2)
\end{aligned}
$$

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. 4)

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

Functions S_0 and S_1 are defined in clause 5.3.

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

$(f_{K0}, f_{K1}, f_{K2}, f_{K3}) = f_K$ are calculated in sequence.

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

Example in hex:

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

5.3 Function S

S_0 and S_1 are defined as follows:

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

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:

Given 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-NX options (see clause 1.2)

In this example, the following FEAL options are selected:

(1) Round number: N=32

6.2 Input data

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

The key-block K is given as :

```
K    =    0000 0001 0010 0011 0100 0101 0110 0111
          1000 1001 1010 1011 1100 1101 1110 1111
          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
          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_{39} , each consisting of 16 bits generated from the key-block K.

6.3.1 Iterative calculation (see 4.1.2)

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

$(A_0, B_0) = K_L$ 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

Q_1 = `1000 1000 1000 1000 1000 1000 1000 1000` in bit sequence

= `88 88 88 88` in hex

Q_2 = `0000 0001 0010 0011 0100 0101 0110 0111` in bit sequence

= `01 23 45 67` in hex

Q_3 = `1000 1001 1010 1011 1100 1101 1110 1111` in bit sequence

= `89 AB CD EF` 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

$B_1 = f_K (A_0, B_0 \oplus D_0 \oplus Q_1)$

$=$ 0111 0101 0001 1001 0111 0001 1111 1001 in bit sequence

$=$ 75 19 71 F9 in hex

$K_0 =$ 0111 0101 0001 1001 in bit sequence

$=$ 75 19 in hex

$K_1 =$ 0111 0001 1111 1001 in bit sequence

$=$ 71 F9 in hex

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

$K_0 =$ 75 19	$K_1 =$ 71 F9	$K_2 =$ 84 E9	$K_3 =$ 48 86
$K_4 =$ 88 E5	$K_5 =$ 52 3B	$K_6 =$ 4E A4	$K_7 =$ 7A DE
$K_8 =$ FE 40	$K_9 =$ 5E 76	$K_{10} =$ 98 19	$K_{11} =$ EE AC
$K_{12} =$ 1B D4	$K_{13} =$ 24 55	$K_{14} =$ DC A0	$K_{15} =$ 65 3B
$K_{16} =$ 3E 32	$K_{17} =$ 46 52	$K_{18} =$ 1C C1	$K_{19} =$ 34 DF
$K_{20} =$ 77 8B	$K_{21} =$ 77 1D	$K_{22} =$ D3 24	$K_{23} =$ 84 10
$K_{24} =$ 1C A8	$K_{25} =$ BC 64	$K_{26} =$ A0 DB	$K_{27} =$ BD D2
$K_{28} =$ 1F 5F	$K_{29} =$ 8F 1C	$K_{30} =$ 6B 81	$K_{31} =$ B5 60
$K_{32} =$ 19 6A	$K_{33} =$ 9A B1	$K_{34} =$ E0 15	$K_{35} =$ 81 90
$K_{36} =$ 9F 72	$K_{37} =$ 66 43	$K_{38} =$ AD 32	$K_{39} =$ 68 3A

where:

$Q_1 = Q_4 = Q_7 = Q_{10} = Q_{13} = Q_{16} = Q_{19}$,

$Q_2 = Q_5 = Q_8 = Q_{11} = Q_{14} = Q_{17} = Q_{20}$,

$Q_3 = Q_6 = Q_9 = Q_{12} = Q_{15} = Q_{18}$.

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,

$(L_0, R_0) = (L_0, R_0) \oplus (K_{32}, K_{33}, K_{34}, K_{35})$

= 0001 1001 0110 1010 1001 1010 1011 0001
1110 0000 0001 0101 1000 0001 1001 0000
in bit sequence

= 19 6A 9A B1 E0 15 81 90 in hex

Next,

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

= 0001 1001 0110 1010 1001 1010 1011 0001
1111 1001 0111 1111 0001 1011 0010 0001
in bit sequence

= 19 6A 9A B1 F9 7F 1B 21 in hex

Output of this processing stage is:

L_0 = 0001 1001 0110 1010 1001 1010 1011 0001
in bit sequence

= 19 6A 9A B1 in hex

R_0 = 1111 1001 0111 1111 0001 1011 0010 0001
in bit sequence

= F9 7F 1B 21 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:

$f(R_0, K_0)$ = 0101 0101 0101 1100 1111 1101 0111 1100

in bit sequence

= 55 5C FD 7C in hex

Details are described in clause 6.4.2.2.

$L_0 \oplus f(R_0, K_0)$ = 4C 36 67 CD in hex

Output of first stage of the iterative calculation is:

$L_1 = R_0$ = 1111 1001 0111 1111 0001 1011 0010 0001

in bit sequence

= F9 7F 1B 21 in hex

R_1 = 0100 1100 0011 0110 0110 0111 1100 1101

in bit sequence

= 4C 36 67 CD 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:

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

= 1111 1001 0111 1111 0001 1011 0010 0001

in bit sequence

= F9 7F 1B 21 in hex

β = $(\beta_0, \beta_1) = K_0$

= 0111 0101 0001 1001 in bit sequence

$= \texttt{75 19}$ in hex

$\alpha_0 = \texttt{1111 1001} = \texttt{F9}$ in hex, $\alpha_1 = \texttt{0111 1111} = \texttt{7F}$ in hex

$\alpha_2 = \texttt{0001 1011} = \texttt{1B}$ in hex, $\alpha_3 = \texttt{0010 0001} = \texttt{21}$ in hex

$\beta_0 = \texttt{0111 0101} = \texttt{75}$ in hex, $\beta_1 = \texttt{0001 1001} = \texttt{19}$ in hex

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

$f_1 = \alpha_1 \oplus \beta_0 = \texttt{0000 1010} = \texttt{0A}$ in hex

$f_2 = \alpha_2 \oplus \beta_1 = \texttt{0000 0010} = \texttt{02}$ in hex

$f_1 = f_1 \oplus \alpha_0 = \texttt{1111 0011} = \texttt{F3}$ in hex

$f_2 = f_2 \oplus \alpha_3 = \texttt{0010 0011} = \texttt{23}$ in hex

$f_1 = S_1(f_1, f_2) = \texttt{0101 1100} = \texttt{5C}$ in hex

$f_2 = S_0(f_2, f_1) = \texttt{1111 1101} = \texttt{FD}$ in hex

$f_0 = S_0(\alpha_0, f_1) = \texttt{0101 0101} = \texttt{55}$ in hex

$f_3 = S_1(\alpha_3, f_2) = \texttt{0111 1100} = \texttt{7C}$ 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. (only i in decimal)

The Process stages

i	L_i	R_i	K_{i-1}	$f(R_{i-1}, K_{i-1})$
0	19 6A 9A B1	F9 7F 1B 21		
1	F9 7F 1B 21	4C 36 67 CD	75 19	55 5C FD 7C
2	4C 36 67 CD	DE 02 58 65	71 F9	27 7D 43 44
3	DE 02 58 65	06 82 45 EF	84 E9	4A B4 22 22
4	06 82 45 EF	69 E5 14 95	48 86	B7 E7 4C F0
5	69 E5 14 95	3E 27 61 05	88 E5	38 A5 24 EA
6	3E 27 61 05	DA 4B 20 7D	52 3B	B3 AE 34 E8
7	DA 4B 20 7D	3B 40 E0 FA	4E A4	05 67 81 FF
8	3B 40 E0 FA	83 50 5F 94	7A DE	59 1B 7F E9
9	83 50 5F 94	9E A6 25 93	FE 40	A5 E6 C5 69
10	9E A6 25 93	6B CC 2E 80	5E 76	E8 9C 71 14
11	6B CC 2E 80	B7 79 7F FC	98 19	29 DF 5A 6F
12	B7 79 7F FC	88 8D EF 7A	EE AC	E3 41 C1 FA
13	88 8D EF 7A	93 F8 74 E6	1B D4	24 81 0B 1A
14	93 F8 74 E6	37 D1 63 B7	24 55	BF 5C 8C CD
15	37 D1 63 B7	44 46 BC E4	DC A0	D7 BE C8 02
16	44 46 BC E4	FA FE 29 0B	65 3B	CD 2F 4A BC
17	FA FE 29 0B	D8 6B 48 E4	3E 32	9C 2D F4 00
18	D8 6B 48 E4	54 2D 6E BB	46 52	AE D3 47 B0
19	54 2D 6E BB	2C 82 BF 2A	1C C1	F4 E9 F7 CE
20	2C 82 BF 2A	5B BA E9 71	34 DF	0F 97 87 CA
21	5B BA E9 71	38 28 49 8B	77 8B	14 AA F6 A1
22	38 28 49 8B	0E A7 1A 8C	77 1D	55 1D F3 FD

23	0E A7 1A 8C	33 9C D0 13	D3 24	0B B4 99 98
24	33 9C D0 13	C6 58 51 F1	84 10	C8 FF 4B 7D
25	C6 58 51 F1	E0 B2 08 38	1C A8	D3 2E D8 2B
26	E0 B2 08 38	71 55 D4 0B	BC 64	D7 0D 85 FA
27	71 55 D4 0B	BE 94 A0 EA	A0 DB	5E 26 A8 D2
28	BE 94 A0 EA	88 95 B5 3A	BD D2	F9 C0 61 31
29	88 95 B5 3A	E1 DB DC 34	1F 5F	5F 4F 7C DE
30	E1 DB DC 34	A6 3F CF 84	8F 1C	2E AA 7A BE
31	A6 3F CF 84	93 2D DF 16	6B 81	72 F6 03 22
32	93 2D DF 16	03 E9 32 D4	B5 60	A5 D6 FD 50

6.4.3 Post processing (see 2.4)

First, interchanging L_{32} and R_{32} yields:

(R_{32}, L_{32}) = 0000 0011 1110 1001 0011 0010 1101 0100
1001 0011 0010 1101 1101 1111 0001 0110
in bit sequence
= 03 E9 32 D4 93 2D DF 16 in hex.

Next,

$(R_{32}, L_{32}) = (R_{32}, L_{32}) \oplus (\phi, R_{32})$

(R_{32}, L_{32}) = 0000 0011 1110 1001 0011 0010 1101 0100
1001 0000 1100 0100 1110 1101 1100 0010
in bit sequence
= 03 E9 32 D4 90 C4 ED C2 in hex.

Lastly,

$(R_{32}, L_{32}) = (R_{32}, L_{32}) \oplus (K_{36}, K_{37}, K_{38}, K_{39})$

= 1001 1100 1001 1011 0101 0100 1001 0111

0011 1101 1111 0110 1000 0101 1111 1000

in bit sequence

= 9C 9B 54 97 3D F6 85 F8 in hex.

Ciphertext is given as (R_{32}, L_{32}).

The final result (ciphertext) is :

C = 1001 1100 1001 1011 0101 0100 1001 0111

0011 1101 1111 0110 1000 0101 1111 1000

in bit sequence

= 9C 9B 54 97 3D F6 85 F8 in hex.

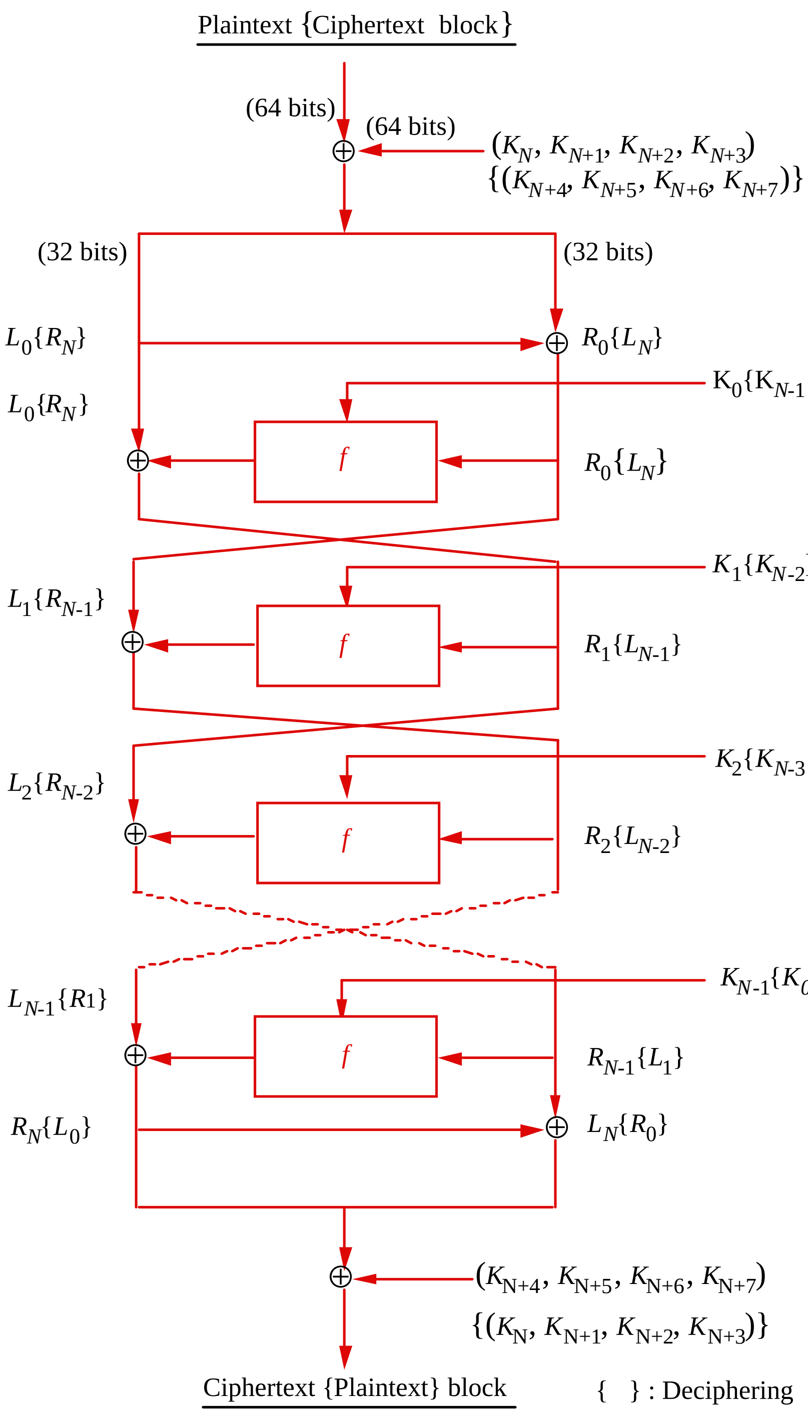
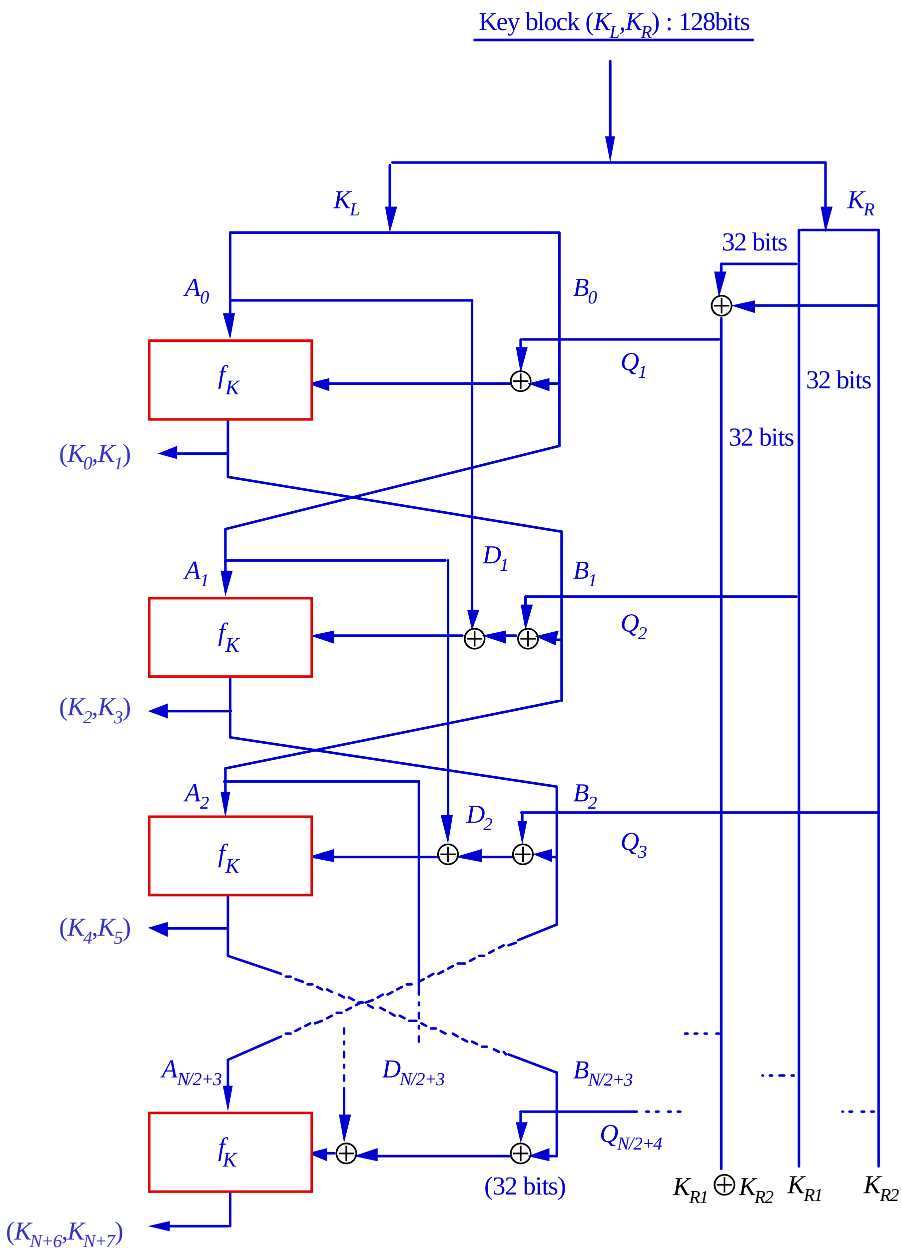


Fig. 1 Data randomization of FEAL-NX (Ciphering/Deciphering algorithm)



$K_{2(r-1)}$: Left half of B_r (16bits)

$K_{2(r-1)+1}$: Right half of B_r (16bits)

Number of iterations is (N/2)+4.

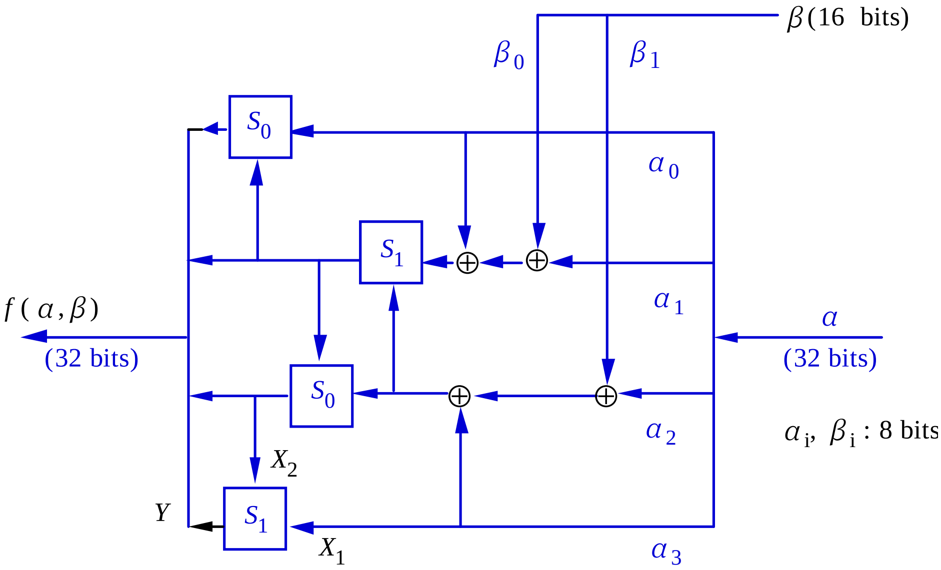
$Q_r = K_{R1} \oplus K_{R2}$ r = 1,4,7,…

$Q_r = K_{R1}$ r = 2,5,8,…

$Q_r = K_{R2}$ r = 3,6,9,…

$K_R = (K_{R1}, K_{R2})$

Fig. 2 Key schedule of FEAL-NX



$Y = S_0(X_1, X_2) = \mathrm{Rot2}\,((X_1+X_2) \bmod 256)$

$Y = S_1(X_1, X_2) = \mathrm{Rot2}\,((X_1+X_2+1) \bmod 256)$

Y : output (8 bits), X_1 / X_2 : inputs (8 bits),

Rot2 (Y) : a 2-bit left rotation on 8-bit data Y

Fig. 3 f -function of FEAL-NX

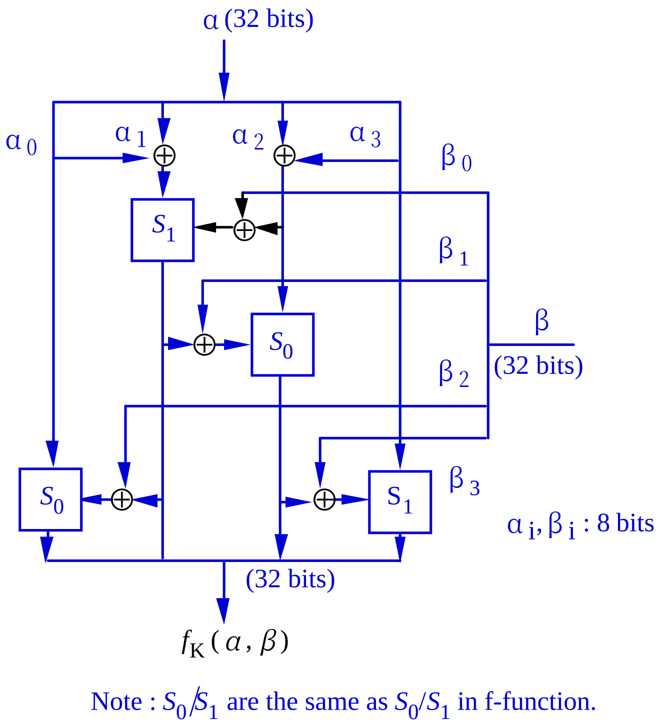


Fig. 4 f_K -function of FEAL-NX