Fast Data Encipherment Algorithm FEAL

Akihiro Shimizu & Shoji Miyaguchi

Electrical Communication Laboratories, NTT
1-2356, Take, Yokosuka-shi, Kanagawa-ken, 238-03, Japan

BACKGROUND

In data communications and information processing systems, cryptography is the most effective way to secure communications and store data. The most commonly used cryptogryphic algorithm is DES [1]. However, it is generally implemented with hardware, and the cost is prohibitive for small scale systems such as personal computer communications. Accordingly, an encipherment algorithm that has safety equal to DES and is suitable for software as well as hardware implementation is needed. The FEAL (Fast data Encipherment ALgorihtm) fills this need.

EVALUATION INDICES FOR ALGORITHM STRENGTH

In FEAL design, two evaluation indices, M and Ms, are adopted to evaluate objectively the data randomization ability of the algorithm. These indices express the approximation degree of ciphertext variation to the binomial distribution B(n, 1/2), in which n is the ciphertext bit length.

M is the average approximation degree of the distribution of

D. Chaum and W.L. Price (Eds.): Advances in Cryptology - EUROCRYPT '87, LNCS 304, pp. 267-278, 1988.

ciphertext variations according to the plaintext or key variations from one-bit to n-bit. Ms is the standard deviation of the approximation degree. When M approaches one (100 percent) and Ms approaches zero, the algorithm does not leave clues which could be used to count backward to the input plaintext or key in the ciphertext. M and Ms are definded separately so that Mp and Mps are for plaintext variations and Mk and Mks are for key variations.

To get the indices, many plaintexts or keys have to be used. Nevertheless, the amount of data which can be treated is generally small compared to the population. Thus, it is important to determine the theoretical index values according to the amount of data by means of statistical calculation. For example, the theoretical values for 16•16•16•63 pieces of data, which are a combination of 16 plaintexts, 16 keys and 16•63 plaintext or key variations, are M = 96.5 percent and Ms = 2.6 percent (Table 1). When the measured values of the indices are close to the theoretical values, the randomness of algorithm ciphertexts is considered saturated.

Table 1 Indices for FEAL and DES

Items		FEAL	DES	Theoretical values
Key indices	Mk	96.5	93.4	96.5
	Mks	2.6	4.9	2.6
Plaintext indices	Mp	96.5	95.5	96.5
	Mps	2.6	3.4	2.6

Note: Data amount=16•16•16•63

DESIGN

FEAL consists of two processing parts. One is the key schedule which generates the 256-bit extended key from the 64-bit secret key. It is designed to generate different extended keys for different secret keys (Fig. 1). The other is the data randomizer (Fig. 2), which generates 64-bit ciphertext from 64-bit plaintext under control of the extended key. The data randomizer uses combinations of involutions (3). One program can perform two functions, enciphering and deciphering, except for the extended key entry. Moreover, the setting of 64-bit extended keys by means of an exclusive-OR operation at the entrance and exit makes attack on the algorithm difficult.

The construction of f (Fig. 3) is such that input bit variations influence all output data. Experiments confirmed that FEAL's f function randomization efficiency is two to three times that of DES.

The S function in the f function, a one-byte data substitution, is as effective as DES's S-box.
The S fucntion is defined as:

S(x, y, delta) = ROT2(T); T=x+y+delta mod256;

x, y: one-byte data; delta:constant (0 or 1);

ROT2(T): 2-bit left rotation operation on T.

Example 1: Where x = 00010011, y = 11110010, delta=1, T = 00000110.
Example 2: ROT2(11011100) = 01110011.

The fk function (Fig. 4) used in the key schedule is the same as the f fucntion except for the entry positon of parameter beta.

FEAL VERSIONS

There is an earlier cryptoanalysis report(4) for FEAL(5). For this reason, the iterative number of data randomizer in FEAL is increased from 4 stages to 8 stages. FEAL described in (5) and (6) is called FEAL version 1.00, and the modified FEAL referred to in this paper is called FEAL Version 2.00. Details for FEAL versions are reported in (7).

STRENGTH AND PERFORMANCE of FEAL(Version 2.00)

FEAL working with no parity in a key block is safe from the all-key attack because it is controlled by a 64-bit key, which is more secure than the 56-bit DES key. Regarding ciphertext randomization, FEAL is considered safe because the randomization indices are closer to the theoretical values than those of DES.

When FEAL is implemented in assembly language on an i-8086 16-bit microprocessor with 8 MHz frequency, it is confirmed that the program size is 400 bytes and the excution time speed reaches 120 kbps.

CONCLUSION

FEAL is an encipherment algorithm suitable for software implementation. It can be applied widely to small scale or other existing systems unable to use DES hardware because of cost. Moreover, FEAL is suitable for hardware implementation, too. Implementated as an LSI, it can be used as the cryptographic method in all data communication fields.

ACKNOWLEGEMENT

We thank Dr. Bert den Boer for finding problems hidden in FEAL Version 1.00.

RERERENCES

(1) FIPS PUB 46, Data Encryption Standard (1977).

(2) S. Miyaguchi, M.Hirano:'Evaluation Criteria for Encipherment and Authentication Algorihtms', Trans. of IECE of Japan. Vol.J69-A, No.10.pp.1252-1259 (Oct. 1986) (in Japanese).

(3) A.G. Kohnheim: 'Cryptography: A Primer', A Wiley Interscience Publication, pp. 236-240 (Jan. 1986).

(4) Bert den Boer,'Cryptonalysis of FEAL', Crypto'87-Rump Session ,Aug. 1987.

(5) A. Shimizu, S. Miyaguchi, 'Fast Data Encipherment Algorihtm FEAL', ABSTRACTS of EUROCRYPT 87 AMSTERDAM, April 1987.

(6) A. Shimizu, S. Miyaguchi,'Fast Data Enciphement Algorihtm FEAL', Trans. of IECE of Japan, Vol.J70-D No.7 pp.1413-1423,July 1987 (in Japanese).

(7) An Extension of Fast Data Encipherment Algorithm FEAL, SITA'87,19-21,Nov.1987 (in English).

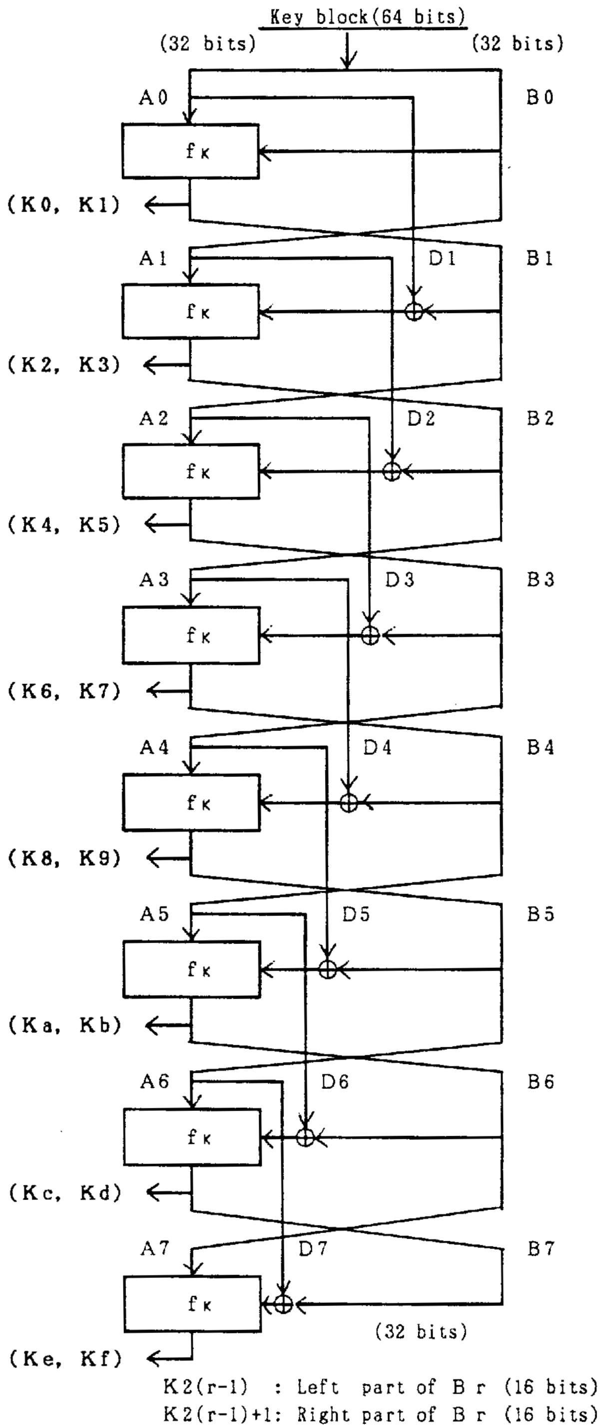


Fig. 1 Key processing part

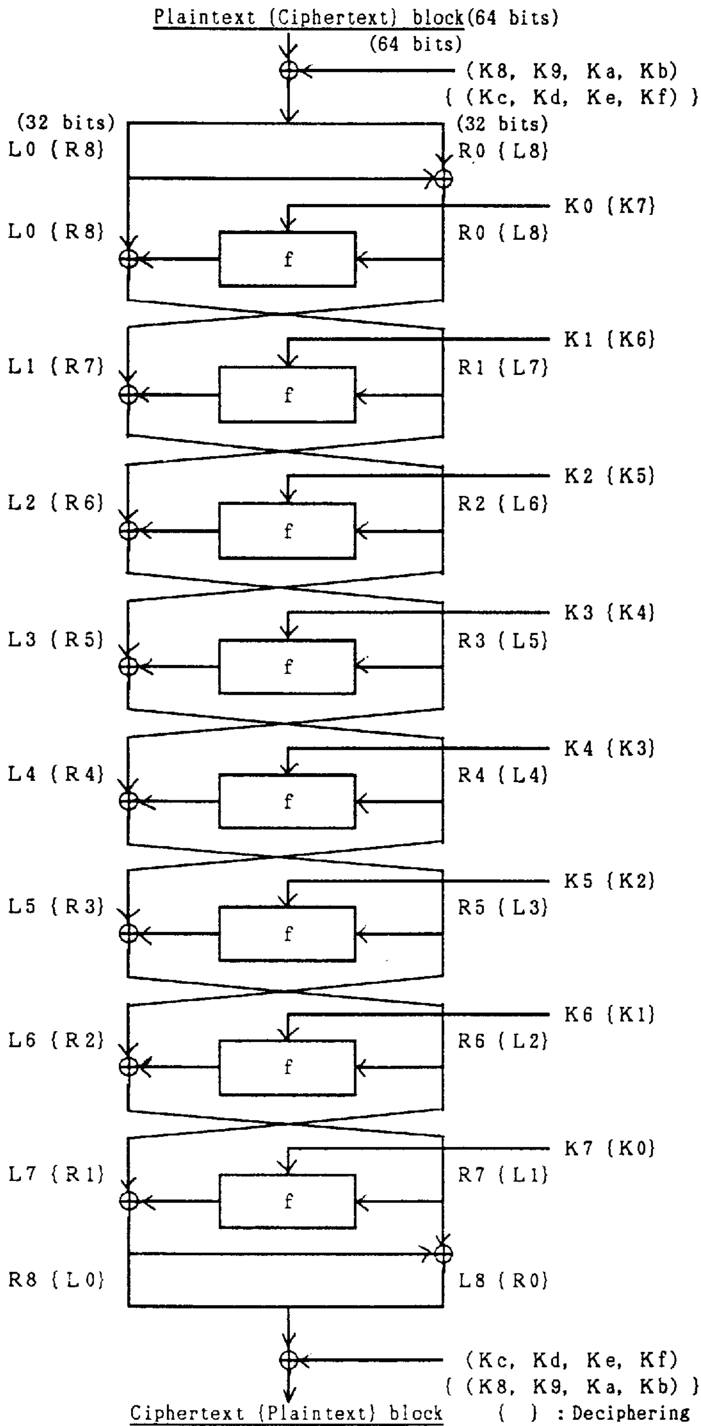
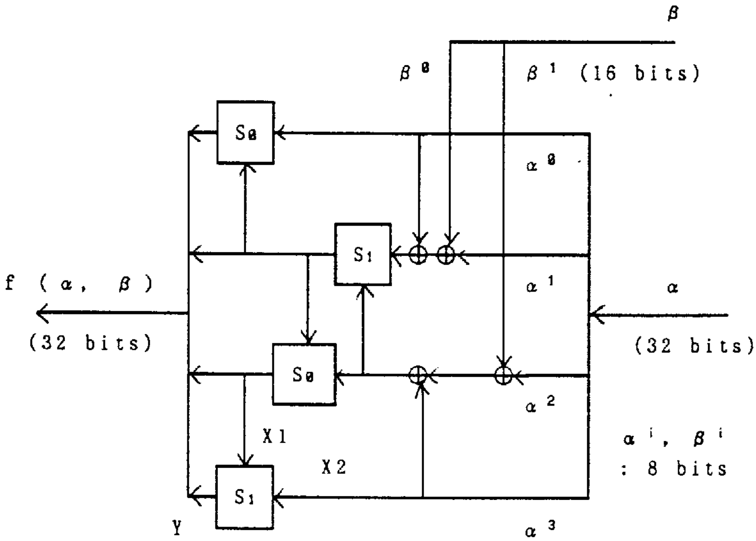


Fig. 2 Data randomizer



Y=Sδ(X1,X2)=ROT2((X1+X2+δ)mod256)
Y:output, X1/X2:inputs, δ:parameter(0 or 1)
ROT2:2 bit left rotation on 8-bit data

Fig. 3 Function f

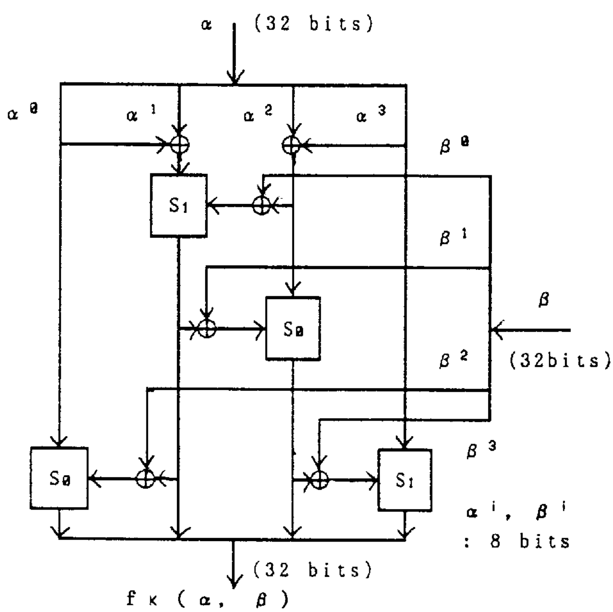


Fig. 4 Function fK

Appendix FEAL Specifications

1 Notations

(1) Block: U, U_r … are blocks of plural octets.

(2) Octet block: U^j, U_r^j are the j th octets in the blocks U, U_r where j = 0, 1, ….

(3) Concatenation: (U, V, ……) is a block concatenated with U, V, …… in this order.

(4) Exclusive-or: $U \oplus V$ is bitwise exclusive-or of block U and V.

(5) Φ is a null block, four octets long.

(6) Assignment: The value of the left side of = sign is assigned the value of the right side.

2 Functions

2.1 Function S

$S(X1, X2, \delta) = ROT2(T)$

$T = X1 + X2 + \delta \bmod 256$

where X1, X2 and T are blocks of one-octet, δ = 0 or 1 (constant value), and ROT2 (T) is the result of a 2 bit left rotation operation on T.

Example 1: Where X1 = 00010011, X2 = 11110010, δ = 1, T = 00000110

Example 2: Rot2 (11011100) = 01110011

2.2 Function f_K

Inputs of function f_K, α and β, are divided into four 1-octet blocks as:

$\alpha = (\alpha^0, \alpha^1, \alpha^2, \alpha^3)$

$\beta = (\beta^0, \beta^1, \beta^2, \beta^3)$.

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

$f = (f^0, f^1, f^2, f^3)$ are calculated in order.

$f_K{}^1 = \alpha^1 \oplus \alpha^0$

$f_K{}^2 = \alpha^2 \oplus \alpha^3$

$f_K{}^1 = S(f_K{}^1, f_K{}^2 \oplus \beta^0, 1)$

$f_K{}^2 = S(f_K{}^2, f_K{}^1 \oplus \beta^1, 0)$

$f_K{}^0 = S(\alpha^0, f_K{}^1 \oplus \beta^2, 0)$

$f_K{}^3 = S(\alpha^3, f_K{}^2 \oplus \beta^3, 1)$

2.3 Function f

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

$f = (f^0, f^1, f^2, f^3)$ are calculated in order.

$f^1 = \alpha^1 \oplus \beta^0 \oplus \alpha^0$

$f^2 = \alpha^2 \oplus \beta^1 \oplus \alpha^3$

$f^1 = S(f^1, f^2, 1)$

$f^2 = S(f^2, f^1, 0)$

$f^0 = S(\alpha^0, f^1, 0)$

$f^3 = S(\alpha^3, f^2, 1)$

3. Key processing

Let A_0 be to the left of the key K and B_0 to the right, i.e.,

$K = (A_0, B_0)$ and $D0 = \Phi$.

Then calculate K_i (i = 0 to 15) for r = 1 to 8,

$D_r = A_{r-1}$

$A_r = B_{r-1}$

$B_r = f_K(A_{r-1}, B_{r-1} \oplus D_{r-1})$

$K_{2(r-1)} = (B_r{}^0, B_r{}^1)$

$K_{2(r-1)+1} = (B_r{}^2, B_r{}^3)$

where A_r, B_r and D_r are auxiliary variables.

4. Enciphering and deciphering

4.1 Enciphering procedure

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

Thus,

$$(L_0, R_0) = (L_0, R_0) \oplus (K_8, K_9, K_{10}, K_{11})$$

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

Then calculate $r = 1$ to 8 in that order,

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

$$L_r = R_{r-1}$$

Lastly, calculate:

$$(R_8, L_8) = (R_8, L_8) \oplus (\Phi, R_8)$$

$$(R_8, L_8) = (R_8, L_8) \oplus (K_{12}, K_{13}, K_{14}, K_{15})$$

Ciphertext is (R_8, L_8).

4.2 Decipehring procedure

Ciphertext is separated into R_8, L_8 of equal lengths. Then,

$$(R_8, L_8) = (R_8, L_8) \oplus (K_{12}, K_{13}, K_{14}, K_{15})$$

$$(R_8, L_8) = (R_8, L_8) \oplus (\Phi, R_8)$$

Then calculate $r = 8$ to 1 in that order,

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

$$R_{r-1} = L_r$$

Lastly, calculate:

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

$$(L_0, R_0) = (L_0, R_0) \oplus (K_8, K_9, K_{10}, K_{11})$$

Plaintext is (L_0, R_0).

5 Parity bits

If parity bits are requested in a key block, the following rule is applied.

Rule: At the begining of key processing, bit positions $8 \times i$ of key block are set to zero where $1 \leqq i \leqq 16$.

6. Working data

is shown in hexadecimal notation.

6.1 When no parity bits exist in a key block

(1) Key = 01 23 45 67 89 AB CD EF

(2) Extended value of the key

(K0, K1, K2, K3, K4, K5, K6, K7) =

DF 3B CA 36 F1 7C 1A EC 45 A5 B9 C7 26 EB AD 25

(K8, K9, K10, K11, K12, K13, K14, K15) =

8B 2A EC B7 AC 50 9D 4C 22 CD 47 9B A8 D5 0C B5

(3) Plaintext = 00 00 00 00 00 00 00 00

(4) Ciphertext= CE EF 2C 86 F2 49 07 52

6.2 When parity bits exist in a key block

(1) Key = 01 23 45 67 89 AB CD EF

(2) Extended value of the key

(K0, K1, K2, K3, K4, K5, K6, K7) =

EF 37 FE DD 04 C3 E3 1D F3 22 B9 A0 C7 AA F6 A6

(K8, K9, K10, K11, K12, K13, K14, K15) =

6A B2 D3 24 F5 DC 72 76 A1 7A 0C 04 B4 E7 CC 8D

(3) Plaintext = 00 00 00 00 00 00 00 00

(4) Ciphertext= 6A 72 2D 1C 46 B3 93 36