Journal of Cryptology
© 1990 International Association for Cryptologic Research

The Cryptanalysis of FEAL-4 with 20 Chosen Plaintexts[1]

Sean Murphy
Department of Mathematics, Royal Holloway and Bedford New College,
University of London, Egham, Surrey TW20 0EX, England

Abstract. An algebraic method is given for a chosen plaintext cryptanalysis of the Nippon Telegraph and Telephone Corporation's FEAL-4 block cipher. The method given uses 20 chosen plaintexts, but can be adapted to use as few as four chosen plaintexts.

Key words. Block cipher, FEAL-4, Cryptanalysis, Chosen plaintext attack.

1. The FEAL-4 Ciphering Algorithm

The FEAL-N cryptosystem has been developed by N.T.T. as a highly programming-efficient block-cipher system, as it does not use look-up tables. It was first presented in [2]. It is essentially an N-round Feistel block cipher operating on 64-bit blocks and determined by a 64-bit key. FEAL-8 is the standard block cipher, but N.T.T. intend that FEAL-4 can be used in cipher block chaining mode when plaintexts are not revealed, a cryptogram-only environment, or for data integrity usage. The best published attack on FEAL-4 was given by Den Boer [1], who used 10,000 chosen plaintexts to recover the key. We give a method that uses at most 20 chosen plaintexts to recover the key. Whereas it may be possible to ensure the absence of 10,000 chosen plaintexts, ensuring the absence of 20 plaintexts may well be too restrictive for most uses.

The functions used to construct FEAL-N are, for $i = 0, 1$, S_i: $\mathbf{Z}_2^8 \times \mathbf{Z}_2^8 \to \mathbf{Z}_2^8$. These are defined for $\mathbf{x}, \mathbf{y} \in \mathbf{Z}_2^8$ by regarding $\mathbf{x}, \mathbf{y}$ as binary numbers x, y in the range $0, \ldots, 255$, so

$$S_i(\mathbf{x}, \mathbf{y}) = \mathrm{Rot}_2(x + y + i \pmod{256}), \tag{1.1}$$

where Rot_2 is a 2-bit rotation to the left. S_0 and S_1 are then used to define two functions, f_K: $\mathbf{Z}_2^{32} \times \mathbf{Z}_2^{32} \to \mathbf{Z}_2^{32}$, which is used to process the key, and f: $\mathbf{Z}_2^{32} \times \mathbf{Z}_2^{16} \to \mathbf{Z}_2^{32}$, which is used to encipher the plaintext.

[1] Date received: January 22, 1990. Date revised: March 29, 1990. This research was supported by S.E.R.C. Research Grant GR/E 64640.

Suppose $a_i, b_i, c_i \in \mathbf{Z}_2^8$ for $i = 0, 1, 2, 3$, and $\mathbf{a} = (a_0, a_1, a_2, a_3) \in \mathbf{Z}_2^{32}$, etc., then

$$\mathbf{c} = f_K(\mathbf{a}, \mathbf{b}) \tag{1.2}$$

is defined in the following manner:

$$\begin{aligned} d_1 &= a_0 \oplus a_1, \\ d_2 &= a_2 \oplus a_3, \\ c_1 &= S_1(d_1, d_2 \oplus b_0), \\ c_2 &= S_0(d_2, c_1 \oplus b_1), \\ c_0 &= S_0(a_0, c_1 \oplus b_2), \\ c_3 &= S_1(a_3, c_2 \oplus b_3). \end{aligned} \tag{1.3}$$

A schematic representation of f_K is given in Fig. 1.

The key is processed by using f_K to obtain twelve 6-bit subkeys. This is done by splitting the 64-bit key K into its left and right halves to give two 32-bit strings K_L and K_R. We can define

$$B_{-2} = 0, \qquad B_{-1} = K_L, \qquad B_0 = K_R, \tag{1.4}$$

and, for $i = 1, \ldots, 6$,

$$B_i = f_K(B_{i-2}, B_{i-1} \oplus B_{i-3}). \tag{1.5}$$

The twelve 16-bit subkeys, $K_i, i = 0, \ldots, 11$, used in the enciphering process are then just the left and right halves of B_i, $i = 1, \ldots, 6$, so

$$K_{2(i-1)} = B_i^{\mathrm{L}}, \qquad K_{2i-1} = B_i^{\mathrm{R}}. \tag{1.6}$$

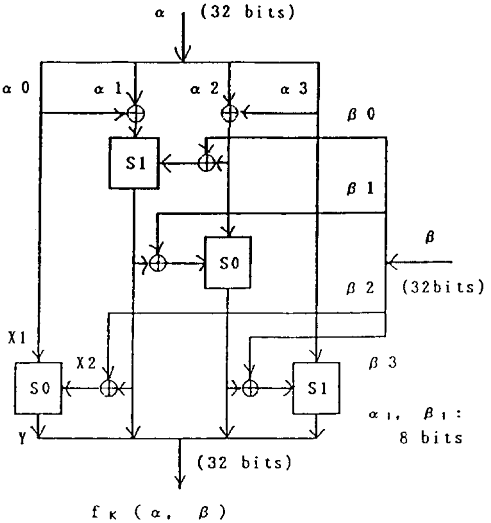

Fig. 1. Function f_K. $Y = S_0(X_1, X_2) = \mathrm{Rot}_2((X_1 + X_2) \bmod 256)$, $Y = S_1(X_1, X_2) = \mathrm{Rot}_2((X_1 + X_2 + 1) \bmod 256)$, Y: output (8 bits); X_1/X_2 (8 bits): inputs. $\mathrm{Rot}_2(Y)$: a 2-bit left rotation on 8-bit data Y.

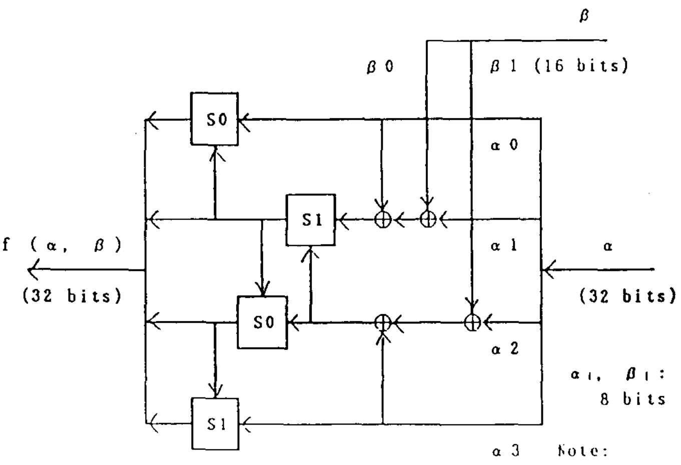

Fig. 2. Function f. S_0/S_1 are the same as S_0/S_1 of f_K.

Now suppose that a_i, $c_i \in \mathbf{Z}_2^8$ for $i = 0, 1, 2, 3$, and also that b_1, $b_2 \in \mathbf{Z}_2^8$, with $\mathbf{b} = (b_1, b_2) \in \mathbf{Z}_2^{16}$ and $\mathbf{a} = (a_0, a_1, a_2, a_3)$, $\mathbf{c} \in \mathbf{Z}_2^{32}$, etc., then we can define

$$\mathbf{c} = f(\mathbf{a}, \mathbf{b}) \tag{1.7}$$

as follows:

$$\begin{aligned} d_1 &= a_0 \oplus a_1 \oplus b_1, \\ d_2 &= a_2 \oplus a_3 \oplus b_2, \\ c_1 &= S_1(d_1, d_2), \\ c_2 &= S_0(d_2, c_1), \\ c_0 &= S_0(a_0, c_1), \\ c_3 &= S_1(a_3, c_2). \end{aligned} \tag{1.8}$$

Figure 2 is a schematic diagram of f.

Suppose we wish to encode the 64-bit plaintext P. Firstly, we split P into its left and right halves to give 32-bit strings P_L and P_R. From these we can calcualte L_0 and R_0:

$$\begin{aligned} L_0 &= P_L \oplus (K_4, K_5), \\ R_0 &= P_L \oplus P_R \oplus (K_4, K_5) \oplus (K_6, K_7). \end{aligned} \tag{1.9}$$

We then perform four rounds of Feistel cipher defined by f and the keys K_0, K_1, K_2, K_3. Thus, for $i = 1, 2, 3, 4$, we calculate

$$\begin{aligned} L_i &= R_{i-1}, \\ R_i &= L_{i-1} \oplus f(R_{i-1}, K_{i-1}). \end{aligned} \tag{1.10}$$

Finally, the enciphered message is $C = (C_L, C_R)$, where

$$\begin{aligned} C_L &= R_4 \oplus (K_8, K_9), \\ C_R &= R_4 \oplus L_4 \oplus (K_{10}, K_{11}). \end{aligned} \tag{1.11}$$

Similarly, if we know the key, we can decode any cryptogram simply by following the above procedure in reverse.

2. Reformulation of FEAL-4 Algorithm

In order to attack the algorithm, we reformulate it by the method given by Den Boer [1]. Firstly, we define a function $G: \mathbf{Z}_2^{32} \to \mathbf{Z}_2^{32}$ that expresses the linear nature of f. Suppose $a_i, c_i \in \mathbf{Z}_2^8$ for $i = 0, 1, 2, 3$, and $\mathbf{a} = (a_0, a_1, a_2, a_3)$, $\mathbf{c} \in \mathbf{Z}_2^{32}$, etc., then we can define

$$\mathbf{c} = G(\mathbf{a}) \tag{2.1}$$

by

$$\begin{aligned} d_1 &= a_0 \oplus a_1, \\ d_2 &= a_2 \oplus a_3, \\ c_1 &= S_1(d_1, d_2), \\ c_2 &= S_0(d_2, c_1), \\ c_0 &= S_0(a_0, c_1), \\ c_3 &= S_1(a_3, c_2), \end{aligned} \tag{2.2}$$

so clearly

$$f(\mathbf{a}, \mathbf{b}) = G(a_0, a_1 \oplus b_1, a_2 \oplus b_2, a_3). \tag{2.3}$$

Therefore Fig. 2 is a schematic diagram of G if we take $\beta_0 = \beta_1 = 0$. The cryptanalysis of FEAL-4 will depend upon the fast solution of linear equations involving G. This is considered in the next section.

We finally need to define two further simple functions, $\theta_L, \theta_R: \mathbf{Z}_2^{32} \to \mathbf{Z}_2^{32}$, by

$$\begin{aligned} \theta_L(a_0, a_1, a_2, a_3) &= (0, a_0, a_1, 0), \\ \theta_R(a_0, a_1, a_2, a_3) &= (0, a_2, a_3, 0), \end{aligned} \tag{2.4}$$

where $a_i \in \mathbf{Z}_2^8$, so

$$\begin{aligned} \theta_L(B_i) &= (0, K_{2(i-1)}, 0), \\ \theta_R(B_i) &= (0, K_{2i-1}, 0). \end{aligned} \tag{2.5}$$

These two functions can be used to define the following six 32-bit key-dependent constants:

$$\begin{aligned} M_1 &= B_3 \oplus \theta_R(B_1), \\ N_1 &= B_3 \oplus B_4 \oplus \theta_L(B_1), \\ M_2 &= \theta_L(B_1) \oplus \theta_L(B_2), \\ N_2 &= \theta_R(B_1) \oplus \theta_R(B_2), \\ M_3 &= B_5 \oplus B_6 \oplus \theta_R(B_1), \\ N_3 &= B_5 \oplus \theta_L(B_1). \end{aligned} \tag{2.6}$$

Note that the outer 16 bits in both M_2 and N_2 are zero.

We are now in a position to rewrite the FEAL-4 algorithm in the following manner:

$$\begin{aligned} X_0 &= P_L \oplus M_1 = L_0 \oplus \theta_R(B_1), \\ Y_0 &= P_L \oplus P_R \oplus N_1 = R_0 \oplus \theta_L(B_1) = L_1 \oplus \theta_L(B_1), \\ X_1 &= X_0 \oplus G(Y_0) = R_1 \oplus \theta_R(B_1) = L_2 \oplus \theta_R(B_1), \\ Y_1 &= Y_0 \oplus G(X_1) = R_2 \oplus \theta_L(B_1) = L_3 \oplus \theta_L(B_1), \\ X_2 &= X_1 \oplus G(Y_1 \oplus M_2) = R_3 \oplus \theta_R(B_1) = L_4 \oplus \theta_R(B_1), \\ Y_2 &= Y_1 \oplus G(X_2 \oplus N_2) = R_4 \oplus \theta_L(B_1), \\ C_L &= Y_2 \oplus N_3, \\ C_R &= X_2 \oplus M_3 \oplus C_L. \end{aligned} \tag{2.7}$$

Again, we can decode a cryptogram by following the above procedure in reverse. Thus, if we can caluclate the 160 unknown bits in the constants M_1, M_2, M_3, N_1, N_2, N_3, we can decipher any cryptogram, and also use the key processing equations to recover the key.

3. The Fast Solution of Linear Equations Involving *G*

In order to find the constants M_1, M_2, M_3, N_1, N_2, N_3 we need to solve equations involving the function G. The simplest such problems involve solving

$$G(\mathbf{x} \oplus \mathbf{a}) = \mathbf{b} \tag{3.1}$$

for $\mathbf{x}$, where $\mathbf{a}$ and $\mathbf{b}$ are known. We can solve this directly, since S_i is an invertible function in the sense that we can solve $S_i(\mathbf{x}, \mathbf{a}) = \mathbf{b}$ uniquely for $\mathbf{x}$. We can however give a general method to solve (3.1), irrespective of whether S_i is an invertible function. There are two reasons for doing this, firstly to show that FEAL-4 is weak cipher no matter how S_i is defined, and secondly to motivate the solution of linear equations involving G. Thus, suppose G were not invertible, then the most naive method to solve (3.1) would be to calculate $G(\mathbf{x} \oplus \mathbf{a})$ for every $\mathbf{x} \in \mathbf{Z}_2^{32}$. However this would require 2^{32} evaluations of G, that is 2^{34} S_i evaluations. However, suppose we check whether

$$S_1(z_1 \oplus a_0 \oplus a_1, z_2 \oplus a_2 \oplus a_3) = b_1 \tag{3.2}$$

for each $z_1, z_2 \in \mathbf{Z}_2^8$. This will require 2^{16} S_1 evaluations. For most values of z_1 and z_2, (3.2) will be false. For those values for which (3.2) is true, we can check whether

$$\begin{aligned} S_0(b_1, z_2 \oplus a_2 \oplus a_3) &= b_2, \\ S_0(b_1, x_0 \oplus a_0) &= b_0, \\ S_1(b_2, x_3 \oplus a_3) &= b_3 \end{aligned} \tag{3.3}$$

for values of x_0, $x_3 \in \mathbf{Z}_2^8$, stopping when one of the equalities is false. If all the equations in (3.2) and (3.3) are true, then we can recover x_1 and x_2 by

$$x_1 = z_1 \oplus x_0, \qquad x_2 = z_2 \oplus x_3 \tag{3.4}$$

to obtain solutions for $\mathbf{x}$.

Another equation we need to solve is

$$G(\mathbf{x} \oplus \mathbf{a}) \oplus G(\mathbf{x} \oplus \mathbf{b}) = \mathbf{d}, \tag{3.5}$$

where **a, b, d** are known constants. We can amend (3.2) and (3.3) to give the following equations to be checked for $z_1, z_2, x_0, x_3 \in \mathbf{Z}_2^8$:

$$\begin{gathered}
S_1(z_1 \oplus a_0 \oplus a_1, z_2 \oplus a_2 \oplus a_3) = \alpha_1, \qquad S_1(z_1 \oplus b_0 \oplus b_1, z_2 \oplus b_2 \oplus b_3) = \beta_1, \\
\alpha_1 \oplus \beta_1 = d_1, \\
S_0(\alpha_1, z_2 \oplus a_2 \oplus a_3) = \alpha_2, \qquad S_0(\beta_1, z_2 \oplus b_2 \oplus b_3) = \beta_2, \\
\alpha_2 \oplus \beta_2 = d_2, \\
S_0(\alpha_1, x_0 \oplus a_0) \oplus S_0(\beta_1, x_0 \oplus b_0) = d_0, \\
S_1(\alpha_2, x_3 \oplus a_3) \oplus S_1(\beta_2, x_3 \oplus b_3) = d_3.
\end{gathered} \tag{3.6}$$

Equation (3.4) then gives us solutions for **x**. In this case, we need 2^{17} evaluations of S_1 to check the truth of $\alpha_1 \oplus \beta_1 = d_1$ for each $z_1, z_2 \in \mathbf{Z}_2^8$.

Solving (3.5) will often give us too many solutions for **x** than we can efficiently handle, so instead we often solve simultaneous equations of the form:

$$\begin{gathered}
G(\mathbf{x} \oplus \mathbf{a}) \oplus G(\mathbf{x} \oplus \mathbf{b}) = \mathbf{d}, \\
G(\mathbf{x} \oplus \mathbf{a}) \oplus G(\mathbf{x} \oplus \mathbf{c}) = \mathbf{e}.
\end{gathered} \tag{3.7}$$

We can do this efficiently by checking whether the analagous pairs of simultaneous equations to (3.6) hold at every stage. This will require only 2^{18} evaluations of S_i to check the first pair of simultaneous equations.

4. Choosing the Plaintexts

Let P^i denote the ith plaintext, $i = 0, \ldots, 19$, with P_L^i and P_R^i being the left and right halves of P^i. Similarly suppose C^i denotes the ith coded plaintext having left and right halves C_L^i and C_R^i. We can then define

$$Q^i = P_L^i \oplus P_R^i \tag{4.1}$$

and

$$D^i = C_L^i \oplus C_R^i. \tag{4.2}$$

The 20 plaintexts are then chosen according to the following rules:

(1) Choose $P^0, P^{12}, P^{14}, P^{16}, P^{17}, P^{18}, P^{19}$ randomly.

(2) Choose $P_L^5, P_L^6, P_L^7, P_L^8, P_L^9, P_L^{10}, P_L^{11}, P_L^{13}, P_L^{15}$ randomly.

(3) Define

$$\begin{aligned}
P_L^1 &= P_L^0 \oplus 80800000, \\
P_L^2 &= P_L^0 \oplus 00008080, \\
P_L^3 &= P_L^0 \oplus 40400000, \\
P_L^4 &= P_L^0 \oplus 00004040.
\end{aligned} \tag{4.3}$$

(4) Define

$$\begin{aligned} P_{\mathrm{R}}^{i} &= P_{\mathrm{L}}^{i} \oplus Q^{0}, \qquad i = 1, \ldots, 11, \\ P_{\mathrm{R}}^{13} &= P_{\mathrm{L}}^{13} \oplus Q^{12}, \\ P_{\mathrm{R}}^{15} &= P_{\mathrm{L}}^{15} \oplus Q^{13}. \end{aligned} \tag{4.4}$$

Thus we have chosen seven plaintexts and nine half-plaintexts at random, that is 736 random bits out of a total of 1280 bits.

5. Cryptanalysis of FEAL-4

Referring to (2.7), we see that

$$\begin{aligned} Y_1 &= Y_0 \oplus G(X_1) = Y_0 \oplus G(X_0 \oplus G(Y_0)) = Y_0 \oplus G(P_{\mathrm{L}} \oplus M_1 \oplus G(Y_0)) \\ &= Y_2 \oplus G(X_2 \oplus N_2) = C_{\mathrm{L}} \oplus N_3 \oplus G(D \oplus M_3 \oplus N_2), \end{aligned} \tag{5.1}$$

and hence

$$C_{\mathrm{L}} \oplus (Y_0 \oplus N_3) \oplus G[P_{\mathrm{L}} \oplus (M_1 \oplus G(Y_0))] \oplus G[D \oplus (M_3 \oplus N_2)] = 0. \tag{5.2}$$

Thus, for a particular plaintext P^i, $i = 0, \ldots, 19$, we can define

$$\begin{aligned} U^i &= Y_0^i \oplus N_3, \\ V^i &= M_1 \oplus G(Y_0^i), \\ W &= M_3 \oplus N_2, \end{aligned} \tag{5.3}$$

so (5.2) becomes

$$C_{\mathrm{L}}^i \oplus U^i \oplus G(P_{\mathrm{L}}^i \oplus V^i) \oplus G(D^i \oplus W) = 0. \tag{5.4}$$

However, for $i = 0, \ldots, 11$, $Y_0 = Q^i \oplus N_1$ and $G(Y_0)$ is constant, and hence $U^i = U^0$ and $V^i = V^0$, and so we can rewrite (5.4) as

$$C_{\mathrm{L}}^i \oplus U^0 \oplus G(P_{\mathrm{L}}^i \oplus V^0) \oplus G(D^i \oplus W) = 0, \qquad i = 0, \ldots, 11. \tag{5.5}$$

In order to solve (5.5) for U^0, V^0, and W, we can first eliminate U^0 by adding two copies of (5.5) to obtain

$$C_{\mathrm{L}}^0 \oplus C_{\mathrm{L}}^i \oplus G(P_{\mathrm{L}}^0 \oplus V^0) \oplus G(P_{\mathrm{L}}^i \oplus V^0) \oplus G(D^0 \oplus W) \oplus G(D^i \oplus W) = 0. \tag{5.6}$$

Thus, if we knew the value of $G(P_{\mathrm{L}}^0 \oplus V^0) \oplus G(P_{\mathrm{L}}^i \oplus V^0)$, (5.6) would give us an equation for W alone. Consider $G(\mathbf{a})$ and $G(\mathbf{a} \oplus 80800000)$. It is easy to see that in both cases, d_1 and d_2 in (2.2) are the same, and hence only c_0 differs. a_0 and $a_0 \oplus 80$ differ only in the first place, so c_0 differs only in the seventh place. By a similar reasoning we can evaluate other sums, and so we have

$$\begin{aligned} G(\mathbf{a}) \oplus G(\mathbf{a} \oplus 80800000) &= 02000000, \\ G(\mathbf{a}) \oplus G(\mathbf{a} \oplus 00008080) &= 00000020, \\ G(\mathbf{a}) \oplus G(\mathbf{a} \oplus 40400000) &= 01000000, 03000000, \\ G(\mathbf{a}) \oplus G(\mathbf{a} \oplus 00004040) &= 00000001, 00000003. \end{aligned} \tag{5.7}$$

Hence, from (4.3), we have

$$\begin{aligned} G(D^0 \oplus W) \oplus G(D^1 \oplus W) &= C_L^0 \oplus C_L^1 \oplus 02000000, \\ G(D^0 \oplus W) \oplus G(D^2 \oplus W) &= C_L^0 \oplus C_L^2 \oplus 00000002. \end{aligned} \tag{5.8}$$

This is an equation of the form of (3.7), so we can solve it efficiently and get solutions for W. We can eliminate many of these solutions by checking to see whether they satisfy

$$\begin{aligned} G(D^0 \oplus W) \oplus G(D^3 \oplus W) \oplus C_L^0 \oplus C_L^3 &= 01000000, 03000000, \\ G(D^0 \oplus W) \oplus G(D^1 \oplus W) \oplus C_L^0 \oplus C_L^1 &= 00000001, 00000003. \end{aligned} \tag{5.9}$$

This typically gives us up to ten different values for W. For each value of W, we can find values of V^0 by solving

$$\begin{aligned} G(P_L^0 \oplus V^0) \oplus G(P_L^5 \oplus V^0) &= C_L^0 \oplus C_L^5 \oplus G(D^0 \oplus W) \oplus G(D^5 \oplus W), \\ G(P_L^0 \oplus V^0) \oplus G(P_L^6 \oplus V^0) &= C_L^0 \oplus C_L^6 \oplus G(D^0 \oplus W) \oplus G(D^6 \oplus W), \end{aligned} \tag{5.10}$$

which is again of the form of (3.7). Equation (5.5) then gives us U^0. We can then check each triplet (W, V^0, U^0) to see if it satifies (5.5) for the other plaintexts with $Q^i = Q^0$, that is to say $i = 7, 8, 9, 10, 11$. This will usually give us less than 20 triplets (W, V ^{0}, U^0).

For each triplet, we can try and solve for the key constants M_1, M_2, M_3, N_1, N_2, N_3. Now,

$$\begin{aligned} U^{12} = U^{13} &= U^0 \oplus Q^0 \oplus Q^{12}, \\ U^{14} = U^{15} &= U^0 \oplus Q^0 \oplus Q^{14}, \end{aligned} \tag{5.11}$$

and so (5.4) gives us

$$\begin{aligned} G(P_L^{12} \oplus V^{12}) &= G(D^{12} \oplus W) \oplus C_L^{12} \oplus U^{12}, \\ G(P_L^{14} \oplus V^{14}) &= G(D^{14} \oplus W) \oplus C_L^{14} \oplus U^{14}. \end{aligned} \tag{5.12}$$

These are two equations of the form (3.1), so we can solve them for $V^{12} = V^{13}$ and $V^{14} = V^{15}$. These two values can then be checked with (5.4) for $i = 13, 15$.

If we obtain solutions for V^{12} and V^{14}, we can attempt to calculate the key constant N_1. Equation (5.3) gives us

$$\begin{aligned} G(Q^0 \oplus N_1) \oplus G(Q^{12} \oplus N_1) &= V^0 \oplus V^{12}, \\ G(Q^0 \oplus N_1) \oplus G(Q^{14} \oplus N_1) &= V^0 \oplus V^{14}, \end{aligned} \tag{5.13}$$

which is again of the form (3.7), so it can be efficiently solved for N_1. For each possibility for N_1, we can calulate V^{16}, and see if (5.4) is satisfied. Knowing possible solutions for N_1 immediately gives us corresponding possible solutions for M_1 and N_3.

We now proceed by finding M_2. We can do this by calculating the values of X_1 and Y_1 in (2.7) for plaintexts P^0, P^{17}, and P^{18}, and noting that

$$G(Y_1 \oplus M_2) = X_1 \oplus X_2 = X_1 \oplus D \oplus M_3. \tag{5.14}$$

Hence,

$$\begin{aligned} G(Y_1^0 \oplus M_2) \oplus G(Y_1^{17} \oplus M_2) &= X_1^0 \oplus X_1^{17} \oplus D^0 \oplus D^{17}, \\ G(Y_1^0 \oplus M_2) \oplus G(Y_1^{18} \oplus M_2) &= X_1^0 \oplus X_1^{18} \oplus D^0 \oplus D^{18}, \end{aligned} \tag{5.15}$$

which is of the form of (3.7). However, the outer 16 bits of M_2 are zero, so we have to solve (5.15) for M_2 allowing for this. For each possible value of M_2, we can calculate X_2^0, X_2^{17}, and X_2^{18}, and hence three values for M_3, which should of course agree. If not, we can reject M_2. Finally, we can calculate N_2, checking that the outer 16 bits are zero.

Thus, we have calculated M_1, M_2, M_3, N_1, N_2, N_3, and we can do a last check by coding all 20 plaintexts with (2.7), including the previously unused P^{19}.

If we need to recover the key, we can use a method given by Den Boer [1]. The knowledge of M_1, N_1, M_3, N_3 and (2.6) gives us the outer 16 bits of B_3, B_4, B_5, B_6. If we know the outer 16 bits of both the output and the two inputs to f_K, we can determine all the input and output bits of f_K. We can thus solve the final iteration of the key scheduling,

$$B_6 = f_K(B_4, B_5 \oplus B_3), \tag{5.16}$$

to find the values of B_4, B_6, and $B_3 \oplus B_5$. We can also now caluclate $B_2^0 \oplus B_2^2$ and $B_2^1 \oplus B_2^3$. Therefore, if we knew B_3^2, we would know all the bits of B_2, and hence $B_1, \ldots, B_6$. We can thus simply try all 256 possibilities for B_3^2 in

$$B_5 = f_K(B_3, B_4 \oplus B_2). \tag{5.17}$$

Having solved (5.17), we thus have sufficient information to determine $B_1, \ldots, B_6$. We can now recover the key by first solving $B_3 = f_K(B_1, B_2 \oplus K_R)$ for K_R, and then solving $B_2 = f_K(K_R, B_1 \oplus K_L)$ for K_L. The key, K, is then given by $K = (K_L, K_R)$.

Of course, we do not need all 20 plaintexts to recover the key. We could dispense with some of the plaintexts that are only used to check possibilities for the various constants. This would of course mean that we would have to compute more possibilities for the various constants until later in the algorihm, and consequently computing time would be increased. For example, we could cut the number of plaintexts to seven, using P^0, P^1, P^2, P^5, P^6, P^{12}, P^{14}, and taking $P^{17} = P^{12}$ and $P^{18} = P^{14}$. If we are prepared to handle equations of the form of (3.5) rather than (3.7), we could only use four plaintexts, P^0, P^1, P^5, P^{12}, with $P^{17} = P^{12}$.

It may be possible to extend this method of attack to a known plaintext attack. The idea is to take similar pairs of plaintext P^i and P^j and predict the value of some of the bits of $V^i \oplus V^j$ with high probability, and hence the value of certain bits of $G(P_L^i \oplus V^i) \oplus G(P_L^j \oplus V^j)$ with high probability. We can thus write down an equation for certain bit positions of the form of (5.8), which we may be able to solve for some of the bits of W. We could solve many such equations and hence find W. We then proceed as before, solving equations in certain bit positions as best we can by using similar pairs of plaintext and predicting the evaluation of the function G in certain bit positions with high probability.

6. Conclusions

This method of attack, with 20 plaintexts takes up to 4 hours computing on a Sun 3/60 Workstation, not a particularly powerful computer. The length of time depends on the key, most keys having been found in less than an hour.

However the function G is defined, any four round cipher is vulnerable to the type of attack based on (5.4) that is outlined above. Obviously, the more easily equations involving G are solved, the quicker the attack. The problem is not so much the S_i transformation, since the methods of Sections 3 would work for any function S_i with a 16-bit input and 8-bit output, as that the two inner 8-bit blocks of the output of G, c_1 and c_2 in (2.2), both depend only on the same 16 bits, d_1 and d_2. We are therefore easily able to find a_1 and d_2 by exhaustive search, and hence invert G. G would be much harder to invert if it was redesigned so that every output block of 8 bits depended on 16 different input bits and an exhaustive search became infeasible. A further improvement would be to redesign the function f so as to remove the linear connection between $\mathbf{a}$ and $\mathbf{b}$ in (1.7). This would make the definition of a funcion like G impossible and ensure that every output block of 8 bits of f depended on all 48 input bits.

Whilst FEAL-4 is not intended for use in a chosen plaintext environment, a cipher that falls so quickly to so few plaintexts must be too weak for most practical putposes. If the protocol for the use of a cipher system has to be such so as to preclude any possibiltiy of less than 12 chosen plaintexts, then the advantages of using a fast ciphering algorithm like FEAL-4 are less important and it would be better to use a more secure cipher. Such a protocol would seem to be too restrictive for most data integrity uses. Even if such a protocol could be guaranteed, data integrity usage would give rise to many pairs of similar plaintexts, so a known plaintext attack of the type outlined above might well succeed.

References

[1] B. Den Boer, Cryptanalysis of FEAL, *Advances in Cryptology—Eurocrypt 88*, Lecture Notes in Computer Science, Vol. 330, Springer-Verlag, Berlin, 1989.

[2] A. Shimizu and S. Miyaguchi, Fast Data Encipherment Algorithm FEAL, *Advances in Cryptology—Eurocrypt 87*, Lecture Notes in Computer Science, Vol. 304, Springer-Verlag, Berlin, 1988.