A STATISTICAL ATTACK OF THE FEAL-8 CRYPTOSYSTEM

Henri Gilbert and Guy Chassé
Centre National d'Etudes des Télécommunications (CNET)
PAA-TIM
38-40, rue du Général Leclerc
92131 Issy les Moulineaux
FRANCE

ABSTRACT.

This paper presents a chosen plaintext cryptanalysis of the FEAL-8 cryptosystem. The attack requires the ciphertext corresponding to approximately 10000 pairs of 64 bit plaintext blocks. The difference (bitwise xor) between the two blocks of each pair is equal to an appropriately selected constant. We first state that some differential statistics for intermediate values of the data randomizer are non uniform and independent of the encryption key. We then show that these statistics can be used to compute gradually the expanded key of the data randomizer.

In 1989 some announcements were made that the so-called FEAL-8, 8 round version of the FEAL cryptosystem, was vulnerable to a chosen plaintext attack [1]. So far, however, only the cryptanalysis of the 4 round version FEAL-4 by Bert Den Boer [2] was published. In this paper we present a chosen plaintext attack of FEAL-8 based on some differential statistics of its data randomization scheme.

1 Description of the FEAL-8 randomizer and first remarks.

We are using the following notations.

- If A represents a 32 bit word $(a_0, a_1, ..., a_{31})$, A_0 is the byte $(a_0 a_1 ... a_7)$, A_1 is the byte $(a_8 a_9 ... a_{15})$,... etc. We also write $A = (A_0, A_1, A_2, A_3)$.

- If A and A' are two binary strings (e.g. bits or bytes or 32 bit words, etc...), $A \oplus A'$ is the bitwise "exclusive or" ("xor" or addition modulo 2) between A and A'.

- If B is the byte $(b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0)$, the right side bit b_0 is also referred to as $B[0]$, the bit b_1 as $B[1]$,... etc. The byte $(b_5 b_4 b_3 b_2 b_1 b_0 b_7 b_6)$ is denoted by $ROT_2(B)$.

- If B and B' are two bytes, they will be sometimes considered as two integers in the usual way (the right side bit is equal to the integer modulo 2) and the byte $B + B'$ will be the sum modulo 256 of these 2 integers. We also define the ternary operator SBOX : $SBOX(B, B', \epsilon) = ROT_2(B + B' + \epsilon)$ where ϵ is equal to 0 or 1.

The FEAL-8 algorithm, which is specified in the reference [3], can be divided in two components : the key schedule and the data randomizer. We do not need here to consider the details of the key schedule : let us only tell that the key schedule transforms the 64 bit secret key K in an expanded key of 32 bytes $K_0, K_1, ..., K_{31}$.

The data randomizer operates on a 64 bit plaintext block I divided into a left 32 bit word I^0 and a right one I^1. It produces the 64 bit ciphertext block O divided into a left 32 bit word O^0 and a right one O^1. The data randomization can be split in the three following steps.

The initial step.

We start with a 64 bit word (I^0, I^1) as input. The we compute a new 64 bit word (X^0, X^1) defined by :

$$X^0 = I^0 \oplus (K_{16}, K_{17}, K_{18}, K_{19})$$
$$X^1 = I^1 \oplus I^0 \oplus (K_{16} \oplus K_{20}, K_{17} \oplus K_{21}, K_{18} \oplus K_{22}, K_{19} \oplus K_{23})$$

The main step.

The 64 bit word (X^0, X^1) is taken as the input to an 8 round Feistel scheme. The rounds are numbered from 0 to 7. At round i, a new 32 bit word X^{i+2} is produced, given by the relation :

$$X^{i+2} = f_i(X^{i+1}) \oplus X^i.$$

The function f_i is defined by :

$$\{0,1\}^{32} \longrightarrow \{0,1\}^{32}$$

$$X = (X_0, X_1, X_2, X_3) \longmapsto Y = (Y_0, Y_1, Y_2, Y_3)$$

where the bytes of Y are computed in the following order :

$$\begin{aligned}
Y_1 &= SBOX(X_0 \oplus X_1 \oplus K_{2i}, X_2 \oplus X_3 \oplus K_{2i+1}, 1), \\
Y_0 &= SBOX(X_0, Y_1, 0), \\
Y_2 &= SBOX(Y_1, X_2 \oplus X_3 \oplus K_{2i+1}, 0), \\
Y_3 &= SBOX(Y_2, X_3, 1).
\end{aligned}$$

It is easier to understand such a transformation with a diagram.

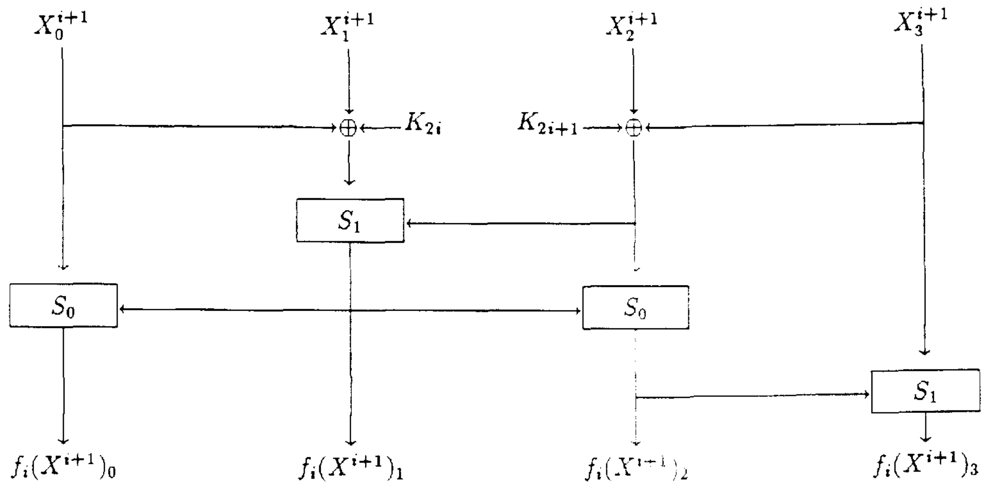

Figure 1 : diagram of the f function.

The function f_i is one to one and only depending on the two expanded key bytes K_{2i} and K_{2i+1}. In the (traditional) 64 bit representation of the Feistel scheme, the round i output is the 64 bit word (X^{i+1},X^{i+2}).

The final step.

The 64 bit word (X^8,X^9) is taken as input to this final step. The intermediate 64 bit word $(X^9,X^8 \oplus X^9)$ is xored with the 64 bit word $(K_{24},K_{25},...K_{31})$, giving the 64 bit output (O^0,O^1).

It is worth making two elementary remarks on the FEAL randomizer. We state them in an intentionnaly informal manner.

• If we consider bytes as elements of the GF(2) vector space $GF(2)^8$, the only nonlinear elementary operation of the whole randomization is the modulo 256 addition $+$ (the ROT_2 function and the xor operation $\oplus$ are linear).

• The diffusion introduced by an f_i function is quite poor for some of the input bits. The f_i functions are built with the $SBOX$ operator. Notice that if two bytes B and B' are taken as input to an $SBOX$, the ouput is equal, up to the ROT_2 rotation, to a byte B'' which takes one of the values $B + B'$ and $B + B' + 1$. Each bit of B or B' only influences the bits of equal or higher weight in B''. The diffusion between bits of different weight is entirely based on the carry propagation phenomenon and the diffusion effect decreases fastly as the distance between the bit positions is increasing. In fact, the modulo 256 sum $B + B'$ of two bytes B and B' has in most cases many bits equal to the corresponding ones of $B \oplus B'$: roughly speaking the two operations $+$ and $\oplus$ do not strongly differ.

These properties will be very useful in the Sections 3 and 4 where we show that an attacker can take advantage from them by a suitable choice of the plaintext.

2 Outline of our attack method.

The FEAL-8 data randomizer belongs to the quite large family of the layered blockciphers. We are calling "statistical meet in the middle attack" the following chosen plaintext attack method, which is appropriate for some layered blockciphers with a limited number of rounds.

The attacker first chooses the plaintext blocks, later referred to as plaintext samples. This selection is the crucial issue of the attack. The plaintext samples must be chosen in such a way that the distribution of some "last round input bits" (i.e. bits which appear in the enciphering scheme as input bits of the last round) is not uniform and independent of the actual secret key K. The number n of plaintext blocks must be taken sufficiently large to prevent the concealment of this phenomenon. The n selected plaintext samples are encrypted with the secret key K, providing n ciphertext samples.

Observing these n ciphertext samples, the attacker tries to guess the last round input bits, using the upward calculation scheme given by the deciphering algorithm. For that purpose he makes an exhaustive trial of all the expanded key bits involved in the upward calculation scheme. Denote by "last round key bits" these unknown expanded key bits. For each assumption on the last round key bits, the attacker makes n upward calculations of the last round input bits (one calculation for each ciphertext sample) and finally obtains a distribution of their values. One may reasonably expect that only the right last round key bits lead to the expected non uniform distribution, and that any wrong hypothesis leads to a much more uniform, or at least different distribution. If this happens to be true, once having computed the distribution associated to each assumption, the attacker is able to guess the last round key bits.

Once the last round key bits have been correctly guessed, the problem of obtaining the remaining expanded key bits is more or less similar to finding a chosen plaintext attack against the same blockcipher with a reduced number of rounds. Of course there exist many variants of this attack method.

In the two following sections, we apply the statistical meet in the middle attack method to the FEAL-8 algorithm. We first show how to choose the plaintext samples in order to obtain a non uniform statistical distribution of some intermediate bits. We then show how to use these properties to guess the expanded key.

3 Choice of the plaintext samples.

Our strategy for the choice of the plaintext samples is based on the previous remark concerning the diffusion in the FEAL randomizer. In the + operation between two bytes, the diffusion of the highest weight bit of each input byte is limited to the highest weight bit of the output byte since there is no carry propagation at the left of this bit. In other words, if two samples are such that at any stage of the calculation the input bytes to an $SBOX$ operator of the f function differ only by their highest weight bit, the output bytes will only differ by their second lowest weight bit.

In order to take advantage from this weakness, it seems appropriate to select the plaintext samples in pairs. We use the notations I^0, I^1, $X^0,\ldots,X^9$, O^0, O^1 introduced in Section 1 for the first component of a sample pair and the notations I'^0, I'^1, $X'^0,\ldots,X'^9$, O'^0, O'^1 for the second one. If A represents any variable related to the first element of a sample pair and A' the similar variable for the second element, the notation ΔA will be used instead of $A \oplus A'$.

It is easy to generate random sample pairs (I, I') with the constraints :

$$\begin{aligned}\Delta I^0 &= (2,0,0,2),\\ \Delta I^1 &= (130,128,128,130).\end{aligned}$$

There are 2^{63} such pairs because the order is irrelevant . So we only need to randomly generate the first 64 bit word $I = (I^0, I^1)$, then we deduce $I' = (I'^0, I'^1)$ using the constraints.

The following observations are the basis of our results.

• Looking at the corresponding values of X^0 and X^1 we notice immediately from the initial step of the algorithm that for every value of the secret key :

$$\begin{aligned}\Delta X^0 &= (2,0,0,2),\\ \Delta X^1 &= (128,128,128,128).\end{aligned}$$

(what means, for instance, that only the highest weight bit of the bytes X_i^1 and $X_i'^1$ differ for the integers i, $0 \leq i \leq 3$).

• The 32 bit word :

$$\Delta X^2 = f^0(X^1) \oplus X^0 \oplus f^0(X'^1) \oplus X'^0$$

is equal for every key to the quadruple of bytes $(0,0,0,0)$.

Similarly we obtain :

$$\Delta X^3 = (128,128,128,128)$$

and then :

$$\Delta X^4 = (2,0,0,2).$$

• Some bits of $\Delta X^5, \Delta X^6$ and ΔX^7 are not uniformly distributed in a way which does not strongly depend on the actual key.

The last statement can be deduced from computations similar to those made for $\Delta X^2, \Delta X^3$ and ΔX^4. Nevertheless, experiments are necessary to describe with precision the statistics for the bits of $\Delta X^5, \Delta X^6$ and ΔX^7. These statistics, which are the framework of our cryptanalysis are gathered in the annex, at the end of the paper. The four arrays contain the observed frequency of 0 for each bit of the 32 bit words ΔX^0, ΔX^1, ΔX^2, ΔX^3, ΔX^4, ΔX^5, ΔX^6 and ΔX^7 for random input pairs (I^0, I^1) and (I'^0, I'^1) with the above described constraints. It appears for instance that, even after the seventh round, two bits are not uniformly distributed : $\Delta X_1^7[7]$ and $\Delta X_1^7[6]$. Their 0 frequency is printed in boldface.

The occurence rates in the annex have been obtained by computing the empirical 0 frequency of the corresponding bits with $n = 20000$ sample pairs for a given fixed key K. Let Ω be the set of the 2^{63} inputs respecting the constraints. Considering the set Ω equipped with the uniform probability, we can consider the n trials :

$$\Omega \longrightarrow \{0,1\}$$

$$\omega \longmapsto \Delta X_j^i[k]$$

mapping one sample pair ω to the corresponding bit $\Delta X_j^i[k]$ (the integer i, j, k being fixed to a chosen value) as n independant random variables of same law. So applying the approximation suggested by the Central Limit Theorem we can get an order of magnitude of the error on the experimentally obtained 0 frequency using n samples. Using the Central Limit Theorem to the finite value $n = 20000$ leads to the conclusion that, for the key used in an experiment, the absolute value of the error on each estimated occurence rate is less than 0.01 with a probability larger than 0.99. In order to check that those occurence rates do not strongly depend on the secret key used, we have simply made experiments with different keys : the order of magnitude of the discrepancies between the obtained results is less than 0.01.

Through the same method, we have estimated the occurence rate of the value 0 for the bit $b = \Delta X_0^7[7] \oplus \Delta X_1^7[5]$. The rate of 0.54, was found for several different keys.

We have shown that if an attacker chooses n samples pairs according to the described constraints, he is able to predict some irregularities in the statistics of some intermediate outputs even after seven rounds of the Feistel scheme.

A similar phenomenon can be obtained with other choices of the constraints, for instance :

- $\Delta I^0 = (2,0,0,0)$ and $\Delta I^1 = (130,128,0,0)$,
- $\Delta I^0 = (0,0,0,2)$ and $\Delta I^1 = (0,0,128,130)$.

These two alternative choices for the constraints are not used in the attack described in the next section.

4 Estimation of the expanded key.

We now show how to guess stepwise the secret expanded key from, say, 10000 plaintext sample pairs selected as explained in Section 3 and from the corresponding ciphertext.

The key estimation process can be subdivided in several elementary steps. At each step, one of the statistical properties stated in Section 3 is used for guessing some new bits or new linear bit combinations of the unknown expanded key by an exhaustive search on these new bits.

The splitting of the estimation process in steps is somewhat arbitrary. The division presented here seemed convenient to us for showing the feasablility of the attack, although it is far from being optimal from a performance point of view.

The first step, which provides the 10 first linear bit combinations of the expanded key, is explained in some detail in Section 4.1. In Section 4.2 we summarize the subsequent steps more briefly ; it is mainly intended for readers interested in details. Since all the steps are very similar to the first one, section 4.2 is not essential for the understanding of our attack. Section 4.3 gives the outcome of our attack.

4.1 Description of the first step.

Step 1 uses the non uniform distribution of bit $\Delta X_1^7[7]$. This bit is related to the ciphertext samples (O^0, O^1) and (O'^0, O'^1) by the following relations :

$$\begin{aligned}
X_1^7 &= SBOX(X_0^8 \oplus X_1^8 \oplus K_{14}, X_2^8 \oplus X_3^8 \oplus K_{15}, 1) \oplus X_1^9,\\
X_0^8 &= O_0^0 \oplus O_0^1 \oplus K_{24} \oplus K_{28},\\
X_1^8 &= O_1^0 \oplus O_1^1 \oplus K_{25} \oplus K_{29},\\
X_2^8 &= O_2^0 \oplus O_2^1 \oplus K_{26} \oplus K_{30},\\
X_3^8 &= O_3^0 \oplus O_3^1 \oplus K_{27} \oplus K_{31},\\
X_1^9 &= O_1^0 \oplus K_{25}.
\end{aligned}$$

Let us replace in the first relation the bytes $X_0^8, X_1^8, X_2^8, X_3^8$ and X_1^9 using the subsequent relations. Let us do the same for the second component of a sample pair and let us "xor" the two obtained relations. We can see that the unknown byte ΔX_1^7 can be expressed in terms of the observed ciphertext bytes and of the two unknown bytes $B_1 = K_{14} \oplus K_{24} \oplus K_{28} \oplus K_{25} \oplus K_{29}$ and $B_2 = K_{15} \oplus K_{26} \oplus K_{30} \oplus K_{27} \oplus K_{31}$. More precisely, considering the details of the binary operations (addition and $SBOX$ operation) we see that bit $\Delta X_1^7[7]$ depends only on the 5 lowest weight bits of B_1 and B_2 , i.e. on 10 unknown bits.

For each of the 2^{10} possible assumptions about these 10 bits, the attacker performs 10000 estimations of bit $\Delta X_1^7[7]$ (one for each ciphertext sample pair). For the correct assumption, his estimations will be distributed according to the rates indicated in Section 3 (i.e. the value 0 will occur with probability almost equal to .54), otherwise, for a wrong assumption, his estimations will be more uniformly distributed. So the attacker is able to recognise the right value of the 5 lowest weight bits of B_1 and the 5 lowest weight bits of B_2 in less than 2^{24} upward computations of bit $\Delta X_1^7[7]$.

4.2 Description of the further steps.

Step 2 uses the non uniform distribution of bit $b = \Delta X_0^7[7] \oplus \Delta X_1^7[5]$ for which the value 0 occurs with a probability .54 (see Section 3). In order to relate bit b to the ciphertext, we need the six relations used in the step 1 and the two following additional ones :

$$X_0^7 = SBOX(SBOX(X_0^8 \oplus X_1^8 \oplus K_{14}, X_2^8 \oplus X_3^8 \oplus K_{15}, 1), X_0^8, 0) \oplus X_0^9,$$
$$X_0^9 = O_0^0 \oplus K_{24}.$$

It follows from these eight relations that bit b is a function of the ciphertext and of the three bytes : $B_1 = K_{14} \oplus K_{24} \oplus K_{28} \oplus K_{25} \oplus K_{29}$, $B_2 = K_{15} \oplus K_{26} \oplus K_{30} \oplus K_{27} \oplus K_{31}$ and $B_3 = K_{24} \oplus K_{28}$.

More precisely, one can check that b actually depends on the 5 lowest weight bits of byte B_3, on the 7 lowest weight bits of B_1 and B_2 and on the bit $(B_1[7] \oplus B_2[7])$. Since the 5 lowest weight bits of B_1 and B_2 have been already guessed at step 1, b only depends on 10 new unknown bits. An exhaustive search on these 10 bits provides the right solution.

Step 3 uses the non uniform distribution of bit $\Delta X_1^6[2]$ (for which the value 0 occurs with a probability .62). One can check that this bit only depends on :
- the 6 lowest weight bits of byte B_3,
- the 6 lowest weight bits of byte $B_4 = K_{27} \oplus K_{31}$,
- and on all the bits of B_1 and B_2.

There are 8 new unknown bits : 6 for B_4, and one for B_3 and B_1. An exhaustive search provides the right solution .

Step 4 uses the non uniform distribution of bit $\Delta X_1^6[5]$ (which takes the value 0 with a probability .17). This bit can be expressed in terms of the ciphertext, of the already determined key bits, and of the following 8 new unknown key bits :
- bit $B_3[6]$,
- bit $B_4[6]$,
- the bits 0 and 2 of bytes $B_5 = K_{12} \oplus K_{24} \oplus K_{25}$ and $B_6 = K_{13} \oplus K_{26} \oplus K_{27}$,

- and the two bits $(B_3[7] \oplus B_5[1])$ and $(B_4[7] \oplus B_6[1])$.

An exhaustive search provides the right value for those 8 bits.

Let us summarize the outcome of the four first steps. We have now determined all the bits of bytes B_1, B_2, B_3 and B_4 (except the highest weight bit of B_3 and B_4, which will be determined later on). This implies (as may be checked in using the relations between the 4 byte words X^7, X^8 and X^9) that X^7 is now known up to four unknown constant bytes, so we have gained one round with respect to the initial situation, where only X^8 and X^9 were known up to four unknown constant bytes. So, roughly speaking, the problem of guessing the remaining expanded key bits is now easier than breaking the FEAL-7 cryptosystem.

Steps 5, 6 and 7 use the non uniform distribution of bits $\Delta X_1^6[1], \Delta X_1^5[2]$ and $\Delta X_1^5[5]$ (which take the value 0 with probability .55, 1 and .75 respectively). They enable us to determine :

- the whole bytes $B_7 = K_{12} \oplus K_{25}$ and $B_8 = K_{13} \oplus K_{26}$,
- the not yet known bits of the partially known bytes B_5 and B_6, except the 4 bits $B_5[1]$, $B_6[1]$, $B_5[7]$ and $B_6[7]$ which will be determined later,
- the two bits $(B_5[7] \oplus B_9[1])$ and $(B_6[7] \oplus B_{10}[1])$, which are combinations of the bytes B_5, B_6 and of the unknown bytes $B_9 = K_{10} \oplus K_{14}$ and $B_{10} = K_{11} \oplus K_{15}$,
- the bits number 0 and 2 of the bytes B_9 and B_{10}.

Consequently, the 4 byte word X^6 is now known for each sample up to four unknown constant bytes and we have now gained two rounds with respect to the situation at the beginning of step 1.

Step 8 uses the unbalanced distribution of the 3 lowest weight bits of ΔX_0^5 and of the 4 lowest weight bits of ΔX_1^5 (the probability that all of these 7 bits take the value 0 is about .83) for recovering all the still unknown bits of bytes $B_9 = K_{10} \oplus K_{14}$ and $B_{10} = K_{11} \oplus K_{15}$. Only 100 sample pairs do now suffice for making the correct guess.

The following steps enable us to guess successively :

- the unknown bytes $B_{11} = K_8 \oplus K_{12}$ and $B_{12} = K_9 \oplus K_{13}$ and the two still unknown bits $B_3[7]$ and $B_4[7]$, using the statistics of the 4 bytes word ΔX^4, which is constant and equal to (2,0,0,2) ; less than 100 sample pairs are sufficient ;
- the unknown bytes $B_{13} = K_6 \oplus K_{10}$ and $B_{14} = K_7 \oplus K_{11}$, due to the statistics on the 4 bytes word ΔX^3, which is constant and equal to (128,128,128,128) ; less than 100 sample pairs are sufficient.

At this point of the attack, the use of the ciphertext alone does no longer suffice. This is because the statistics on the 4 byte words $\Delta X^2, \Delta X^1$ and ΔX^0 are trivial consequences of the statistics on the 4 byte words ΔX^3 and ΔX^4. So the statistics on $\Delta X^2, \Delta X^1$ and ΔX^0 do not allow to guess any new unknown expanded key bit. There are two ways to solve this slight difficulty :

- the first approach is to use, say, 100 additional sample pairs, selected according to other constraints, e.g. according to the relations $\Delta I^0 = (128,128,128,128)$ and $\Delta I^1 = (130,128,128,130)$. The statistics on $\Delta X^2, \Delta X^1$ and ΔX^0 are no longer trivial consequences of those on ΔX^3 and ΔX^4 ; they enable us to guess new unknown expanded key bits based on the new ciphertext. Also the plaintext for at least one of the ciphertext samples is required at the end of the attack ;

- the second approach is now to take the plaintext and the ciphertext of the initial samples into account (instead of the ciphertext samples alone) ; the property that, for each sample bit $X_1^1[2]$ differs from the plaintext bit $(I_1^1[2] \oplus I_1^0[2])$ by one constant bit can be first used for recovering new expanded key bits, etc...

We summarize here the last steps of the attack with the first method. The following bytes are successively guessed :

- the unknown bytes $B_{15} = K_4 \oplus K_8$ and $B_{16} = K_5 \oplus K_9$ (using the statistics on ΔX^2) ;

- the unknown bytes $B_{17} = K_2 \oplus K_6$ and $B_{18} = K_3 \oplus K_7$ (using the statistics on ΔX^1) ;

- the unknown bytes $B_{19} = K_0 \oplus K_4$ and $B_{20} = K_1 \oplus K_5$ (using the statistics on ΔX^0).

The eight unknown bytes $B_{21} = K_{16} \oplus K_{20}, B_{22} = K_0 \oplus K_{17} \oplus K_{21}, B_{23} = K_1 \oplus K_{18} \oplus K_{22}, B_{24} = K_{19} \oplus K_{23}, B_{25} = K_{16}, B_{26} = K_2 \oplus K_{17}, B_{27} = K_3 \oplus K_{18}$ and $B_{28} = K_{19}$ are now available with almost no calculation, using one single plaintext sample and the corresponding ciphertext.

4.3 Outcome of the attack.

The two previous sections have described a complete attack. Not surprisingly, this estimation process has provided the 28 combinations $B_1, B_2, ...B_{28}$ instead of the 32 bytes $K_0, K_1, ...K_{31}$: this is because the encryption and decryption functions associated with any 32-uple of expanded key bytes $K_0, K_1, ...K_{31}$ is entirely determined by the bytes $B_1, B_2, ...B_{28}$.

According to our simulations on the thirteen steps of the above attack, it does not require more than two hours computing time on a SUN4 workstation.

As already stated, we have not tried to optimize the performance of the attack. It seems feasible to us to split the computation of the expanded key in much more steps, each of them requiring an exhaustive search on substantially fewer bits. Indeed, instead of performing at each step an exhaustive search on all the new unknown involved expanded key bits, it is feasible to first perform an exhaustive search on the bits which have the major impact due to their position in the addition processes and, after that, to determine the remaining ones by another exhaustive search. This would save much computing time.

5 Conclusions.

In this paper we have presented a statistical attack method which we propose to call "statistical meet in the middle". We have shown how to apply this method to a complete attack on the FEAL-8 enciphering algorithm. Our attack is based on the analysis of about 10000 sample pairs, and requires a rather limited computational expense.

Remaining open questions include :

- how to extend our attack to FEAL-N, when $N > 8$;
- whether our methods are applicable to a known plaintext attack on FEAL-8.

6 Acknowledgements.

We are grateful to our colleagues Mireille Campana and David Arditti, from CNET. The content of this paper is directly related to some previous studies on blockciphers carried out in collaboration with them.

REFERENCES

[1] **A. Shamir** *Lecture at Securicom 89.*

[2] **Bert Den Boer** *Cryptanalysis of F.E.A.L.,* Proceedings of Eurocrypt'88, pp 293-299.

[3] **S. Miyaguchi, S. Shiraishi, S. Shimizu** *Fast Data Encipherment Algorithm FEAL-8* Review of the Electrical Communication Laboratories, Vol. 36, N°4 (1988).

ANNEX

$j =$	7	6	5	4	3	2	1	0
$\Delta X_0^0[j]$	1	1	1	1	1	1	0	1
$\Delta X_0^1[j]$	0	1	1	1	1	1	1	1
$\Delta X_0^2[j]$	1	1	1	1	1	1	1	1
$\Delta X_0^3[j]$	0	1	1	1	1	1	1	1
$\Delta X_0^4[j]$	1	1	1	1	1	1	0	1
$\Delta X_0^5[j]$	.277	.506	.745	.497	.016	.969	.930	.860
$\Delta X_0^6[j]$	.406	.496	.351	.571	.555	.494	.476	.502
$\Delta X_0^7[j]$	.492	.498	.497	.508	.497	.499	.500	.500
$\Delta X_0^8[j]$	.505	.502	.496	.501	.496	.497	.507	.509
$\Delta X_0^9[j]$	.494	.501	.499	.499	.500	.504	.500	.495

$j =$	7	6	5	4	3	2	1	0
$\Delta X_1^0[j]$	1	1	1	1	1	1	1	1
$\Delta X_1^1[j]$	0	1	1	1	1	1	1	1
$\Delta X_1^2[j]$	1	1	1	1	1	1	1	1
$\Delta X_1^3[j]$	0	1	1	1	1	1	1	1
$\Delta X_1^4[j]$	1	1	1	1	1	1	1	1
$\Delta X_1^5[j]$	.063	.871	.750	.504	1	1	.984	.969
$\Delta X_1^6[j]$	.474	.499	.176	.718	.690	.624	.556	.498
$\Delta X_1^7[j]$	.536	.514	.497	.497	.507	.496	.494	.501
$\Delta X_1^8[j]$	.502	.501	.500	.504	.500	.497	.497	.498
$\Delta X_1^9[j]$	.495	.505	.501	.503	.497	.502	.501	.499

$j =$	7	6	5	4	3	2	1	0
$\Delta X_2^0[j]$	1	1	1	1	1	1	1	1
$\Delta X_2^1[j]$	1	1	1	1	1	1	1	1
$\Delta X_2^2[j]$	1	1	1	1	1	1	1	1
$\Delta X_2^3[j]$	0	1	1	1	1	1	1	1
$\Delta X_2^4[j]$	1	1	1	1	1	1	1	1
$\Delta X_2^5[j]$	.310	.506	.742	.491	.015	.969	.920	.844
$\Delta X_2^6[j]$	.577	.502	.650	.591	.553	.501	.493	.503
$\Delta X_2^7[j]$	.497	.493	.497	.502	.502	.504	.502	.497
$\Delta X_2^8[j]$	.500	.492	.504	.498	.507	.503	.503	.498
$\Delta X_2^9[j]$	.499	.496	.503	.498	.503	.501	.499	.502

$j =$	7	6	5	4	3	2	1	0
$\Delta X_3^0[j]$	1	1	1	1	1	1	0	1
$\Delta X_3^1[j]$	0	1	1	1	1	1	1	1
$\Delta X_3^2[j]$	1	1	1	1	1	1	1	1
$\Delta X_3^3[j]$	0	1	1	1	1	1	1	1
$\Delta X_3^4[j]$	1	1	1	1	1	1	0	1
$\Delta X_3^5[j]$	.248	.496	.265	.485	.080	.844	.689	.503
$\Delta X_3^6[j]$	.456	.499	.469	.500	.487	.502	.522	.495
$\Delta X_3^7[j]$	.499	.496	.496	.500	.496	.492	.502	.506
$\Delta X_3^8[j]$	.495	.499	.505	.500	.500	.497	.502	.501
$\Delta X_3^9[j]$	.502	.502	.502	.499	.508	.503	.504	.502

Frequency of 0 on the bits $\Delta X_j^i[k]$ observed for 20000 sample pairs with constraints described in section 3