**Cryptography Assignment**

Differential Cryptanalysis of FEAL – 4  
Vrashabh V Joshi  
Student ID – 13059

**FEAL-4** (Fast Data Encipherment Algorithm with 4 rounds) is a highly uncomplicated block cipher commonly employed in the academic cryptanalysis. Differential cryptanalysis A differential cryptanalysis attack is a chosen plaintext attack in which differences in inputs are followed through the encryption process, used to deduce key bits.

In this assignment, a step-by-step differential cryptanalysis of FEAL-4 has been presented that consists of five independent phases where each is carried out in a different file.  
All the phases are written separately so you may download and directly execute any phase. The stages are carried out in series, and the output of one stage serves as an input to the other one. With every iteration, I have stored all the keys of an earlier output in an array and run the phases sequentially. In the present stage, it consequently applied. Moreover, according to the nature of the diagram, every plaintext and ciphertext pairs are precisely included as a source of input. The inputs are checked and correspond with the output of the Einstein tool.  
These files are submitted in the Loop, and also the recovered keys are also uploaded accordingly.  
Following are the links to the phases:

* FealPhase.java - [LINK](https://github.com/vrush0938/FEAL-4/blob/main/FealPhase.java)
* FealPhase1.java - [LINK](https://github.com/vrush0938/FEAL-4/blob/main/FealPhase1.java)
* FealPhase2.java - [LINK](https://github.com/vrush0938/FEAL-4/blob/main/FealPhases2.java)
* FealPhase3.java - [LINK](https://github.com/vrush0938/FEAL-4/blob/main/FealPhases3.java)
* FEAL4WhiteningKeysRecovery - [LINK](https://github.com/vrush0938/FEAL-4/blob/main/FEAL4WhiteningKeysRecovery.java)

**Representation of FEAL-4**

A diagram of a electrical scheme

AI-generated content may be incorrect.

In this fig, it shows the overall working representation for feal-4, in which when it would be attacking the keys one at a time and deciphering the entire set. It goes by cracking K3 -> K2 -> K1 -> K0 -> K4 & K5

**DIFFERENTIAL CRYPTANALYSIS**

Differential cryptanalysis is one there are out there, where how the plaintext was manipulated can be analyzed to approximate how the shift in the plaintext manipulation corresponds to the ciphertext. It aids in determining the encryption key by watching these patterns. In the case of FEAL 4, this method is exploiting certain input-output behaviour and structural attributes to crack the code.

A diagram of a circuit

AI-generated content may be incorrect.

Y’ = L’ ^ R’

Z’ = L’ ^ 0x02000000

X’ = 0x80800000 ^ Y’

L’ = Left half of cipher text

R’ = Right half of cipher text

During the K3 stage, we will determine the K3 key and its possible candidates. The diagrammatic representation indicates that the selection of plaintext pairs having 0x8080000080800000 difference matches the characteristic of the FEAL-4 differential diagram with the one described in Figure 2.

Using this diagram, we can derive formulas of Y', Z', and X'. The K3 identification process is categorized as Primary and Secondary.

Primary:

During the first step, we are concerned with calculating possible candidates of middle 16 bits of K3 which has the form of (z, a0, a1, z) with z containing all zeroes.

Y Structure: y = (a0, a1,a2, a3)

M(Y) Transformation: M(Y) = (0, a 0 + a 1, a 2 + a 3, 0)

Key Structure: 32bit, M (k) = (0, a0+a1, a2+a3, 0)

To find these possible candidates, we can do a loop over all the 16-bit values (0 to 2 ^ 16 - 1). To know the output we compute it for each value. In the event that this output corresponds to Z' = L' ^ 0x02000000 a candidate key corresponding to this output is stored. These are the stored candidates which are possibilities of values of K3 in the main issue

This helps in eliminating invalid candidates, in which the remaining valid candidates can be forwarded to the secondary phase to be analyzed further.

Secondary:

During the second round, our attention goes to the resolution of K3 (outward 16 bits). The most significant data structure of this phase is K = (a0, primary phase 8-bit, primary phase 8-bit, a1).

In the case of these outer bits we are encrypting 16-bit values (0 to 2 16-1) in a loop to determine possible candidates. Output is computed in every value. When such output agrees with Z' = L' ^ 0x02000000 the key candidate will be stored. Stored candidates are possible candidates in K3 that can be used in the second phase.

Output:

A screen shot of a computer code

AI-generated content may be incorrect.

Key 2 Phase:

A diagram of a computer program

AI-generated content may be incorrect.

u0 = l0 ^ r0;

u1 = f(u0)

Y’ = u1 ^ L

Z’ = u0 ^ 0x60600000

X’ = Y’ ^ 0x00000000

After having the K3 keys during this stage, we have constructed our characteristic so that the plaintext pair difference is 0x6060000060600000 as shown in Fig 3. The decision will help to ensure that it minimizes the number of possible K2 key candidates.

We have changed the value of X' and Z' as a result of our changing the characteristic. This is one of the ways to reduce the number of keys we have to look after.

We have strayed away an aspect of Figure 2 for the reason that it results in a far higher number of K2 key candidates during the primary phase. This brings about the need to make it computationally intensive thus not possible to crack in single shot.

Just like the K3 phase we shall follow up the main and secondary phases to conclude possible K2 key candidates.

Output:

Candidate Key: f96d00, Count: 24

Candidate Key: fb6d00, Count: 24

Candidate Key: f96f00, Count: 24

Candidate Key: fb6f00, Count: 24

Candidate Key: 79ed00, Count: 24

Candidate Key: 7bed00, Count: 24

Candidate Key: 79ef00, Count: 24

Candidate Key: 7bef00, Count: 24

Filter completes with threshold: 16

Candidate Key: bf22e43, Count: 16

Candidate Key: 8b72aec3, Count: 16

Candidate Key: bf22e41, Count: 16

Candidate Key: 8b72aec1, Count: 16

Candidate Key: 9f22e43, Count: 16

Candidate Key: 8972aec3, Count: 16

Candidate Key: 9f22e41, Count: 16

Candidate Key: 8972aec1, Count: 16

Candidate Key: 4bb22e43, Count: 16

Candidate Key: cb32aec3, Count: 16

Candidate Key: 4bb22e41, Count: 16

Candidate Key: cb32aec1, Count: 16

Candidate Key: 49b22e43, Count: 16

Candidate Key: c932aec3, Count: 16

Candidate Key: 49b22e41, Count: 16

Candidate Key: c932aec1, Count: 16

Candidate Key: bf2aec3, Count: 16

Candidate Key: 8b722e43, Count: 16

Candidate Key: bf2aec1, Count: 16

Candidate Key: 8b722e41, Count: 16

Candidate Key: 9f2aec3, Count: 16

Candidate Key: 89722e43, Count: 16

Candidate Key: 9f2aec1, Count: 16

Candidate Key: 89722e41, Count: 16

Candidate Key: 4bb2aec3, Count: 16

Candidate Key: cb322e43, Count: 16

Candidate Key: 4bb2aec1, Count: 16

Candidate Key: cb322e41, Count: 16

Candidate Key: 49b2aec3, Count: 16

Candidate Key: c9322e43, Count: 16

Candidate Key: 49b2aec1, Count: 16

Candidate Key: c9322e41, Count: 16

Final likely K2: bf22e43

Hit Count: 16

**Key 1 Phase:**

A diagram of a computer program

AI-generated content may be incorrect.

u0 = l0 ^ r0

u1 = f(u0)

v0 = u1 ^ L

v1 = f(v0)

Y’ = v1 ^ u0

Z’ = v0 ^ 0x00012b90

X’ = Y’ ^ (p0Left ^ p1Left)

In order to minimize the possible K1 key candidates even further we will make some modifications to the characteristic diagram in Figure 4. This is an adjustment just like the one made previously, where primary and secondary stages are adopted to find out possible key applicants.

Output:

k2 possible candidates Key: 587900

k2 possible candidates Key: 5a7900

k2 possible candidates Key: d8f900

k2 possible candidates Key: daf900

k2 possible candidates Key: 587b00

k2 possible candidates Key: 5a7b00

k2 possible candidates Key: d8fb00

k2 possible candidates Key: dafb00

Key with maximum value: -945846274

Maximum value: 16

completed

k2 possible candidates Key: c79f87fe

k2 possible candidates Key: 471f077e

k2 possible candidates Key: c79f87fc

k2 possible candidates Key: 471f077c

k2 possible candidates Key: c59f87fe

k2 possible candidates Key: 451f077e

k2 possible candidates Key: c59f87fc

k2 possible candidates Key: 451f077c

k2 possible candidates Key: c79f077e

k2 possible candidates Key: 471f87fe

k2 possible candidates Key: c79f077c

k2 possible candidates Key: 471f87fc

k2 possible candidates Key: c59f077e

k2 possible candidates Key: 451f87fe

k2 possible candidates Key: c59f077c

k2 possible candidates Key: 451f87fc

**Key 0 Phase:**

A diagram of a machine

AI-generated content may be incorrect.

u0 = l0 ^ r0

u1 = f(u0)

v0 = u1 ^ L

v1 = f(v0)

w0 = v1 ^u0

w1 = f(w0)

Y’ = w1 ^ v0

Z’ = w0 ^ (p0Left ^ p1Left)

To minimize further the potential K0 key candidates we have modified the characteristic diagram, Figure 5. The change, as in former method, includes the use of initial and second-level stages to detect possible key applicants.

Output:

Key with maximum value: 160178504

Maximum value: 68

completed

possible candidate Keys for K0: 98c2148, Value: 68

possible candidate Keys for K0: 890ca1c8, Value: 68

possible candidate Keys for K0: 98c214a, Value: 68

possible candidate Keys for K0: 890ca1ca, Value: 68

possible candidate Keys for K0: 8b0ca1c8, Value: 68

possible candidate Keys for K0: b8c2148, Value: 68

possible candidate Keys for K0: 8b0ca1ca, Value: 68

possible candidate Keys for K0: b8c214a, Value: 68

possible candidate Keys for K0: 98ca1c8, Value: 68

possible candidate Keys for K0: 890c2148, Value: 68

possible candidate Keys for K0: 98ca1ca, Value: 68

possible candidate Keys for K0: 890c214a, Value: 68

possible candidate Keys for K0: 8b0c2148, Value: 68

possible candidate Keys for K0: b8ca1c8, Value: 68

possible candidate Keys for K0: 8b0c214a, Value: 68

possible candidate Keys for K0: b8ca1ca, Value: 68

K4 & K5 Phase:

K4 and K5 can be directly evaluated according to the data in Figure 1 and Figure 6. As we have the following equations: K4 = Plaintext First Half ^ x1 K5 = Plaintext Second Half ^ (x1 ^ x0) with x0 = w1 ^ y0 and x1 = f(x0) we can find all the possible combinations of K3 and K2 and K1 and K0 in order to discover corresponding codes of K4 and K5.

Output:

For K4 possible keys are 256 and k5 goes same. I posted the links to check on it.  
K4 possible keys - [LINK](https://github.com/vrush0938/FEAL-4/blob/main/K4%20Keys.txt)

K5 possible keys - [LINK](https://github.com/vrush0938/FEAL-4/blob/main/K5%20Keys.txt)

**In order to come across the appropriate 6 key pair**:

In order to represent all of the possible keys one can cycle through all the possible key combinations and give six keys as the input to the FEAL-4 algorithm. In case the output of the ciphertext is equal to the same output of the Einstein ciphertext the given set of keys can be discussed as an appropriate solution.

CONCLUSION:

A step-by-step process used to break FEAL-4 using the differential cryptanalysis method shows that the process is systematic. Thereby, exploring its weakness and using the concept of the difference, we can find the right key quickly and easily. This breakdown explains why there is the need to develop a strong cryptography that cannot be hacked by such attacks.