Cryptography Assignment - Report

Linear Cryptanalysis of the FEAL-4 Block Cipher

Author: Hari Sridharan

Student Number: A00051012

GitHub Link: https://github.com/hariof98/linear-cryptanalysis-feal-4-cipher

Date: November 2025

---

1. Introduction

FEAL-4 (Fast Data Encipherment Algorithm with 4 rounds) is a Feistel cipher that uses a 64-bit key to generate six 32-bit subkeys. The cipher has known vulnerabilities to linear cryptanalysis due to non-random behavior in its F-function. This implementation attempts to exploit these linear approximations to recover the secret subkeys using 200 known plaintext-ciphertext pairs.

---

2. Theoretical Foundation

FEAL-4 Structure

FEAL-4 operates on 64-bit blocks split into two 32-bit halves (L and R). The encryption process:

```
Input: (L0, R0)
Round 1: L1 = R0,       R1 = L0 ⊕ F(R0 ⊕ K0)
Round 2: L2 = R1,        R2 = L1 ⊕ F(R1 ⊕ K1)
Round 3: L3 = R2,        R3 = L2 ⊕ F(R2 ⊕ K2)
Round 4: L4' = R3,       R4' = L3 ⊕ F(R3 ⊕ K3)
```

```
Final:   L4 = R4' ⊕ K4,  R4 = L4' ⊕ R4' ⊕ K5
Output: (L4, R4)
```

F-Function Structure

The F-function uses S-boxes S0 and S1 defined as:

```
S0(a,b) = ROL2((a + b) mod 256)
S1(a,b) = ROL2((a + b + 1) mod 256)
```

Where ROL2 is rotate left by 2 bits.

---

3. Cryptanalysis Approach

Linear Approximation Equations

The attack is based on linear approximations of the F-function. For subkey K0, the following approximations were implemented:

Inner Bytes Approximation (K0):

```
S[5,13,21](L0 ⊕ R0 ⊕ L4) ⊕ S[15](L0 ⊕ L4 ⊕ R4) ⊕ S[15](F(L0 ⊕ R0 ⊕ K0)) = constant
```

Outer Bytes Approximation (K0):

```
S[13](L0 ⊕ R0 ⊕ L4) ⊕ S[7,15,23,31](L0 ⊕ L4 ⊕ R4) ⊕ S[7,15,23,31](F(L0 ⊕ R0 ⊕ K0)) = constant
```

Where S[i] denotes extraction of bit i, and S[i,j,k,l] denotes XOR of bits i, j, k, l.

Similar equations were constructed for K1, K2, and K3:

For K1:

```
Inner: S[5,13,21](L0 ⊕ L4 ⊕ R4) ⊕ S[15](F(L0 ⊕ Y0 ⊕ K1))
Outer: S[13](L0 ⊕ L4 ⊕ R4) ⊕ S[7,15,23,31](F(L0 ⊕ Y0 ⊕ K1))
```

```
Where Y0 = F(L0 ⊕ R0 ⊕ K0)
```

For K2:

```
Inner: S[5,13,21](L0 ⊕ R0 ⊕ L4) ⊕ S[15](F(L0 ⊕ R0 ⊕ Y1 ⊕ K2))
Outer: S[13](L0 ⊕ R0 ⊕ L4) ⊕ S[7,15,23,31](F(L0 ⊕ R0 ⊕ Y1 ⊕ K2))
Where Y1 = F(L0 ⊕ Y0 ⊕ K1)
```

For K3:

```
Inner: S[5,13,21](L0 ⊕ L4 ⊕ R4) ⊕ S[15](L0 ⊕ R0 ⊕ L4) ⊕ S[15](F(L0 ⊕ Y0 ⊕ Y2 ⊕ K3))
Outer: S[13](L0 ⊕ L4 ⊕ R4) ⊕ S[7,15,23,31](L0 ⊕ R0 ⊕ L4) ⊕ S[7,15,23,31](F(L0 ⊕ Y0 ⊕ Y2 ⊕ K3))
Where Y2 = F(L0 ⊕ R0 ⊕ Y1 ⊕ K2)
```

Divide-and-Conquer Strategy

To manage computational complexity, each 32-bit key search was divided into:

- **Inner bytes** (bits 8-23): 12 bits requiring 4,096 iterations
- **Outer bytes** (bits 0-7, 24-31): 20 bits requiring 1,048,576 iterations

Key Construction Process:

Inner key mapping (12-bit to 32-bit):

```
K_inner = ((candidate >> 6) & 0x3F) << 16 | ((candidate & 0x3F) << 8)
```

Outer key construction (20-bit + inner to 32-bit):

```
a0 = (((candidate & 0xF) >> 2) << 6) + ((K_inner >> 16) & 0xFF)
a1 = ((candidate & 0x3) << 6) + ((K_inner >> 8) & 0xFF)
b0 = (candidate >> 12) & 0xFF
b3 = (candidate >> 4) & 0xFF
b1 = b0 XOR a0
```

```
b2 = b3 XOR a1
K = (b0 << 24) | (b1 << 16) | (b2 << 8) | b3
```

Key Derivation for K4 and K5

After finding K0 through K3, the remaining keys were derived using:

```
Y0 = F(L0 XOR R0 XOR K0)
Y1 = F(L0 XOR Y0 XOR K1)
Y2 = F(L0 XOR R0 XOR Y1 XOR K2)
Y3 = F(L0 XOR Y0 XOR Y2 XOR K3)

K4 = L0 XOR R0 XOR Y1 XOR Y3 XOR L4
K5 = R0 XOR Y1 XOR Y3 XOR Y0 XOR Y2 XOR R4
```

---

4. Implementation Details

Program Structure

The implementation consists of three modules:

attack.c - Main cryptanalysis logic:

- Linear approximation functions for each subkey
- Cascading search functions (processKey0Candidate through processKey3Candidate)
- Key construction and validation routines

cipher.c - FEAL-4 cipher operations:

- F-function implementation with S-box operations
- Block decryption for key validation
- Data conversion utilities

data.c - Data loading functions:

- Dynamic memory allocation with expandable arrays

- File parsing for plaintext-ciphertext pairs
- Accessor functions for data retrieval

Search Algorithm

The attack follows this procedure:

```
1. Load 200 plaintext-ciphertext pairs from file
2. Search for K0:
   - Iterate through 4,096 inner byte candidates
   - Check consistency of linear approximation across all pairs
   - For consistent candidates, search 1,048,576 outer byte values
   - Validate complete K0 candidates

3. For each valid K0, search for K1:
   - Apply same inner/outer search strategy
   - Use K0 in linear approximation calculations

4. Continue cascading search for K2 and K3

5. For each K0-K3 combination:
   - Derive K4 and K5 using mathematical relationships
   - Validate complete key set by decryption test

6. Output valid key sets (limited to 256 for efficiency)
```

Consistency Checking

The implementation uses strict consistency checking:

```
// Require same result for all plaintext-ciphertext pairs
for (int pairIdx = 1; pairIdx < numPairs; pairIdx++) {
    if (firstResult != linearApproxK0Inner(pairIdx, innerKey)) {
        // Candidate rejected
        break;
```

```
    }
}
```

This approach allows early termination of invalid candidates, improving performance.

5. Results

Recovered Keys

The attack recovered the following key values:

Subkey	Hexadecimal Value	Bits Found
K0	0x63cab942	32
K1	0x00a0c541	32
K2	0x4674095a	32
K3	0x64204c03	32
K4	0x4b37d10a	32
K5	0xd0a24877	32
Total	-	**192 bits**

Multiple Key Sets

The attack identified 256 different key combinations that decrypt the test data correctly.

The entire result can be found in **result.txt** file.

All keys are treated as 32-bit values. If a key has fewer than 8 hexadecimal characters, it is padded with leading zeros to ensure a full 32-bit representation, using the formula:

$$8 \text{ hex digits} \times 4 \text{ bits per hex digit} = 32 \text{ bits}$$

This guarantees that every key is correctly aligned as a 32-bit word.

6. Conclusion

This implementation demonstrates a linear cryptanalysis attack on FEAL-4 that recovers all 192 bits of the cipher's subkeys. Using a divide-and-conquer approach with consistency checking, the attack executes efficiently and identifies 256 valid key sets.

The results confirm FEAL-4's documented vulnerability to linear cryptanalysis and illustrate fundamental principles of statistical cryptanalysis. The modular implementation provides a clear example of how theoretical cryptanalytic techniques can be applied in practice.