Specification TC03

March 11, 2019

1 Cipher

The cipher takes a 64-bit key and 8-bit words and computes the ciphertext in 16 rounds. The cipher is constructed using a feistel network. We index s.t. the 0-th bit is the LSB.

1.1 Round function

In the round function the state is split into two parts: left and right, where left (l) consists of the 4 most significant bits and right (r) consists of the 4 least significant bits. We denote the i-th state as $w_i = l_i | r_i$ and the i-th round key rk_i, note that l_i, r_i, rk_i are all 4 bits wide. We define the function F to be:

$$F(x) = ((x \lll 1) \& (x \lll 2)) \oplus x$$

The round function is then defined by:

$$\begin{aligned} l_{i+1} &= F(l_i) \oplus r_i \oplus rk_i \\ r_{i+1} &= l_i \end{aligned}$$

The cipher consists of 16 applications of the round functions with round keys:

$$rk_i = (K \ggg 4) \& \texttt{0xF}$$

1.2 Test Vectors

Plaintext	Ciphertext	Key
AB	C8	1234567890ABCDEF
FF	7C	1234567890ABCDEF
99	64	136855A334CD90EF
53	86	136855A334CD90EF

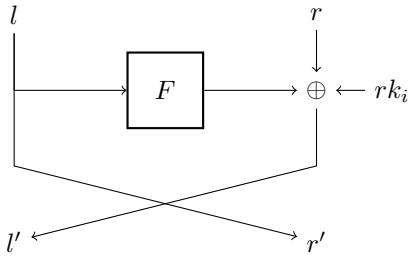

Figure 1: Feistel structure of TC03

1.3 Reference Implementation

```
#!/usr/bin/env python3

def rotate_left(word, n, word_size=4):
    mask = 2**word_size - 1
    return ((word << n) & mask) | ((word >> (word_size - n) & mask))

def rotate_right(word, n, word_size=4):
    mask = 2**word_size - 1
    return ((word >> n) & mask) | ((word << (word_size - n) & mask))

def F(word):
    return (rotate_left(word, 2, 4) & rotate_left(word, 1, 4)) ^ word

def round_function(left, right, key):
    return ((F(left) ^ right ^ key), left)

def encrypt(word, key, rounds=16):
    left = (word >> 4) & 0xF
    right = word & 0xF

    for i in range(rounds):
        left, right = round_function(left, right, key & 0xF)
        key = rotate_right(key, 4, 64)

    return (left << 4) | right

def decrypt(word, key, rounds=16, from_round=16):
    left = word & 0xF
    right = (word >> 4) & 0xF

    # wind the key
    key = rotate_right(key, ((from_round-1)*4) % 64, 64)

    for i in range(rounds):
        left, right = round_function(left, right, key & 0xF)
        key = rotate_left(key, 4, 64)

    return left | (right << 4)

def create_testvectors():
    key = 0x1234567890ABCDEF
    c = encrypt(0xAB, key, 16)
    print("%02X %02X %016X"%(0xAB, c, key))
    c = encrypt(0xFF, key, 16)
    print("%02X %02X %016X"%(0xFF, c, key))
    key = 0x136855A334CD90EF
    c = encrypt(0x99, key, 16)
    print("%02X %02X %016X"%(0x99, c, key))
    c = encrypt(0x53, key, 16)
    print("%02X %02X %016X"%(0x53, c, key))

if __name__ == "__main__":
    import sys
    import random

    if len(sys.argv) == 1:
        print("Error occured")
```

```
    exit()

key = int(sys.argv[1], 16)
for i in range(80):
    word = random.getrandbits(64)
    cipher = encrypt(word, key)
    print("%016X %016X"%(word, cipher))
```