Forry Cipher Block

Muhammad Rizki Fonna[1], **Gloryanson Ginting** [2].

[1,2] Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika (STEI), Institut Teknologi Bandung (ITB), Jalan Ganesha 10, Bandung 40132
E-mail: 13516001@std.stei.itb.ac.id, 13516060@std.stei.itb.ac.id

Abstract. In this paper, the authors aim to design a new block cipher using a feistel structure, apply Shannon's diffusion and confusion principles, apply basic operations: substitution and transposition (permutation), shift, rotation, modulo addition, and apply a number of iterated ciphers and each round using a round key. The resulting block cipher has a block size of 64 bits, the key length of 64 bit, and operates in bits of the message. The cipher block is also implemented in python programming language and then analyzed about the results of the encryption and decryption process of the sample messages used. The results show that the processing time required for both encryption and decryption is quite fast, and the equipment needed to run the application is also light, so the designed algorithm is categorized as economical in terms of computer resources and processing time required. The resulting cipher text is also very sensitive to changes even though only 1 bit of data changed, can completely change the entire cipher text, so that the authenticity of the cipher text is also maintained because each block of cipher text is always associated with the previous block. Each of the same block will generate a varying and completely different block cipher text so that the cipher text is safe from frequency analysis. **Keywords**: block, cipher, feistel, key.

1. Pendahuluan

Kriptografi adalah salah satu teknik yang dapat digunakan untuk mengamankan informasi. Kriptografi memiliki dua tahapan yang biasa dilakukan, yaitu tahapan enkripsi dan dekripsi. Enkripsi adalah proses yang dilakukan untuk mengubah pesan asli menjadi *ciphertext*, sedangkan dekripsi adalah proses yang dilakukan untuk mengubah pesan terenkripsi menjadi pesan yang dapat dibaca dan dipahami. Saat ini algoritma kriptografi yang digunakan merupakan optimasi dari algoritma-algoritma sebelumnya terutama untuk mewujudkan prinsip-prinsip teknik kriptografi yaitu diffusion dan confusion.

Salah satu mode yang sering digunakan pada berbagai algoritma kriptografi modern saat ini adalah mode *cipher block chaining* (CBC). Pada mode ini, setiap blok n-bit *plaintext* akan di XOR-kan dengan blok n-bit *ciphertext* sebelumnya. Hal ini dilakukan pada semua blok kecuali pada blok pertama yang akan di XOR-kan dengan konstanta awal atau initialization vector (IV), dengan ukuran n bit. Berdasarkan gambaran proses tersebut, dapat disimpulkan bahwa *plaintext* yang sama belum tentu akan menghasilkan *ciphertext* yang sama, yang dapat membuat proses kriptanalisis menjadi lebih sulit.

Makalah ini mendeskripsikan kombinasi mode operasi electronic code book (ECB), cipher block chaining (CBC), dan counter mode, dimana *plaintext* terlebih dahulu dienkripsi berdasarkan menggunakan mode operasi electronic code book (ECB), lalu cipher block chaining, kemudian ciphertext enkripsi ulang berdasarkan counter mode. Ciphertext terakhir

yang dihasilkan dari proses enkripsi berdasarkan counter mode akan ciphertext hasil dari algoritma ini.

2. Studi Pustaka

2.1. *Electronic Code Book* (ECB)

Pada salah satu mode DES ini, *plaintext* terbagi dalam beberapa blok dan dienkripsi secara terpisah. Seluruh proses enkripsi tidak tergantung satu sama lain. Kelemahan dari metode ini adalah kurangnya *diffusion*. Karena ECB mengenkripsi blok plaintext yang identik ke dalam blok ciphertext yang identik, ECB tidak menyembunyikan pola data dengan baik.

2.2. *Cipher Block Chaining* (CBC)

Mode *cipher block chaining* dikembangkan oleh IBM pada tahun 1976. Sebelum proses enkripsi, setiap blok *plaintext* di-XOR dengan blok ciphertext sebelumnya. Vektor inisialisasi digunakan pada blok pertama untuk membuat setiap plaintext unik. Walaupun begitu, masih terdapat beberapa kelemahan yang dimiliki oleh mode ini, yaitu:

1. Proses enkripsi dilakukan secara berurutan dan tidak dapat diparalelkan
2. Setiap blok *ciphertext* tergantung pada semua blok plaintext.
3. Dengan demikian, perubahan *plaintext* mempengaruhi semua blok *ciphertext*.
4. Memerlukan konstanta awal (IV) yang harus diketahui pengirim dan penerima tetapi jika IV dikirim dalam bentuk yang jelas, penyerang dapat mengubah bit dari blok pertama.

Proses enkripsi dan deskripsi dari mode ini mengikuti persamaan berikut.
Enkripsi : Ci = Ek (Pi Ci-1), C0 = IV
Deskripsi: Pi = Dk (Ci Ci-1), C0 = IV

2.3. Counter Mode

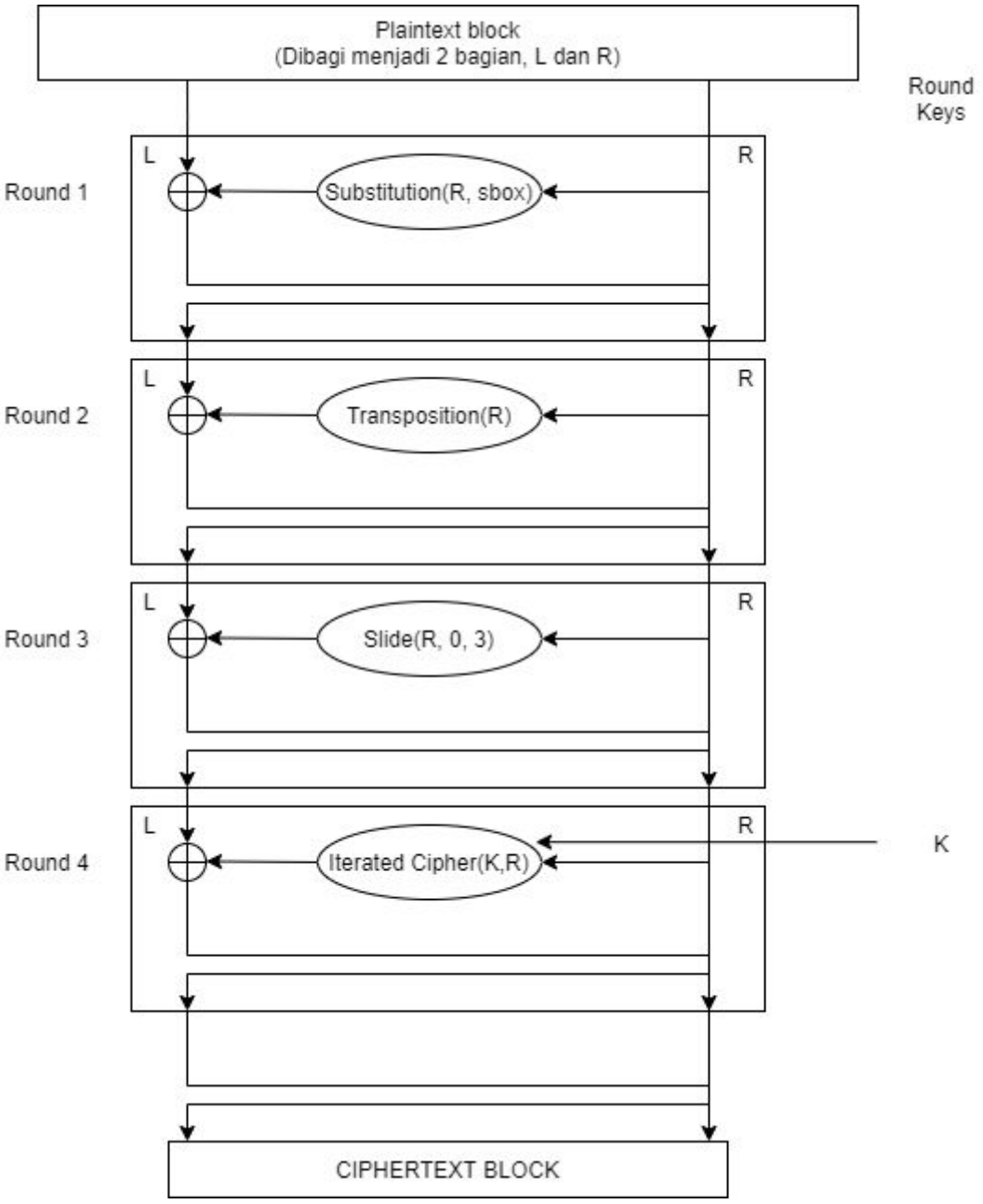
Counter mode mengubah cipher block menjadi stream cipher. Hal ini akan menghasilkan blok keystream berikutnya dengan mengenkripsi nilai inkremental. Counter dapat berupa fungsi apa pun yang menghasilkan urutan yang dijamin tidak akan berulang untuk waktu yang lama, meskipun counter dengan kenaikan satu per satu sebenarnya adalah yang paling sederhana dan paling populer. Bersama dengan CBC, mode CTR adalah salah satu dari dua mode block cipher yang direkomendasikan oleh Niels Ferguson dan Bruce Schneier.

CTR memungkinkan random access property selama proses dekripsi dan juga dikenal sebagai integer counter mode (ICM) dan mode statement integer counter (SIC). Kelebihan mode ini adalah proses enkripsi blok dapat dilakukan secara paralel.

3. Block Cipher yang Diusulkan

Kami menggunakan block cipher dengan ukuran blok 64 bit dan ukuran key juga 64 bit. *Plain text* dioperasikan dalam bits. Pada block cipher diterapkan struktur Feistel di dalam algoritmanya sehingga tidak diperlukan algoritma dekripsi yang berbeda dengan enkripsi. Algoritma juga menggunakan prinsip confusion dan diffusion untuk menyembunyikan hubungan apapun antara plainteks, cipherteks, dan kunci.

Struktur Feistel yang diterapkan pada block cipher terdiri atas beberapa round, yaitu subtitusi, transposisi, pergeseran, dan cipher berulang sebanyak 4 rounds. Berikut diagram Feistel Network yang kami gunakan



Berikut implementasi algoritma enkripsi dan dekripsi yang digunakan dalam bahasa python:

```
def Encrypt(block, key):
    cipher = ""
    temp = []

    n = int(len(block)//2)
    L = stringToBin(block[:n])
    R = stringToBin(block[n:])
    key_0 = stringToBin(key[:4])
    key_1 = stringToBin(key[4:])

    #ROUND 1 (SUBSTITUTION)
```

```
n = int(len(R)//8)
R_temp = []
R_temp.append(R[:8])
R_temp.append(R[8:16])
R_temp.append(R[16:24])
R_temp.append(R[24:])
f_R = ""
for i in range(len(R_temp)):
    sbox_row = binToDec(R_temp[i][:n])
    sbox_col = binToDec(R_temp[i][n:])
    f_R += sbox[i][sbox_row][sbox_col]
temp = R
R = exor(L, f_R)
L = temp

#ROUND 2 (TRANSPOSITION)
f_R = Transpose (R)
temp = R
R = exor(L, f_R)
L = temp

#ROUND 3 (SLIDING)
f_R = slide(R, 0, 3)   #3 BIT TO RIGHT
temp = R
R = exor(L, f_R)
L = temp

#ROUND 4 (ITERATED CIPHER)
#1st (key = first 4 key)
key_temp = stringToBin(key[:4])
f_R = exor(key_temp, R)
temp = R
R = exor(L, f_R)
L = temp

#2nd (key = even index)
key_temp = stringToBin(key[0] + key[2] + key[4] + key[6])
f_R = exor(key_temp, R)
temp = R
R = exor(L, f_R)
L = temp

#3rd (key = odd index)
key_temp = stringToBin(key[1] + key[3] + key[5] + key[7])
f_R = exor(key_temp, R)
temp = R
R = exor(L, f_R)
L = temp

#4th (key = last 4 key)
key_temp = stringToBin(key[4:])
f_R = exor(key_temp, R)
temp = R
```

```
    R = exor(L, f_R)
    L = temp

    #SWAPPING R and l
    temp = R
    R = L
    L = temp
    #convert back to string
    string = L + R
    i=0
    while (i < len(string)):
        char = ""
        for j in range(8):
            char += string[i]
            i+=1
        char = chr(binToDec(char))
        cipher += char
    return cipher

def Decrypt(block, key):
    plain = ""
    temp = []

    n = int(len(block)//2)
    L = stringToBin(block[:n])
    R = stringToBin(block[n:])
    key_0 = stringToBin(key[:4])
    key_1 = stringToBin(key[4:])

    #ROUND 4 (ITERATED CIPHER)

    #4th (key = last 4 key)
    key_temp = stringToBin(key[4:])
    f_R = exor(key_temp, R)
    temp = R
    R = exor(L, f_R)
    L = temp

    #3rd (key = odd index)
    key_temp = stringToBin(key[1] + key[3] + key[5] + key[7])
    f_R = exor(key_temp, R)
    temp = R
    R = exor(L, f_R)
    L = temp

    #2nd (key = even index)
    key_temp = stringToBin(key[0] + key[2] + key[4] + key[6])
    f_R = exor(key_temp, R)
    temp = R
    R = exor(L, f_R)
    L = temp

    #1st (key = first 4 key)
```

```
key_temp = stringToBin(key[:4])
f_R = exor(key_temp, R)
temp = R
R = exor(L, f_R)
L = temp

#ROUND 3 (SLIDING)
f_R = slide(R, 0, 3)   #3 BIT TO RIGHT
temp = R
R = exor(L, f_R)
L = temp

#ROUND 2 (TRANSPOSITION)
f_R = Transpose (R)
temp = R
R = exor(L, f_R)
L = temp

#ROUND 1 (SUBSTITUTION)
n = int(len(R)//8)
R_temp = []
R_temp.append(R[:8])
R_temp.append(R[8:16])
R_temp.append(R[16:24])
R_temp.append(R[24:])
f_R = ""
for i in range(len(R_temp)):
    sbox_row = binToDec(R_temp[i][:n])
    sbox_col = binToDec(R_temp[i][n:])
    f_R += sbox[i][sbox_row][sbox_col]
temp = R
R = exor(L, f_R)
L = temp

#SWAPPING R and l
temp = R
R = L
L = temp
#convert back to string
string = L + R
i=0
while (i < len(string)):
    char = ""
    for j in range(8):
        char += string[i]
        i+=1
    char = chr(binToDec(char))
    plain += char
return plain
```

3.1. Enkripsi

- Subtitusi

Pada algoritma kami menggunakan kotak S (S-Box) dengan dimensi 4 x 16 x 16. Kotak S[0] di-generate secara random dengan seed yang di-generate dari total ord(key) dan kotak S selanjutnya di generate dari total ord(key) + 1. Berikut contoh kotak S yang di-generate dari kunci “abcdefgh”:

```
SBOX[0]
[['11110101' '11101011' '00101100' '10101101' '00001010' '00100000'
  '01111010' '01010101' '10111110' '00010100' '10101110' '10111001'
  '00001000' '01100001' '11010111' '01000111']
 ['00100100' '11100100' '00010011' '01011000' '00010011' '10110110'
  '11010001' '10111101' '00001100' '00100100' '10100111' '10110001'
  '00000011' '01100101' '10110001' '11011111']
 ['11111000' '10010011' '00000110' '10101100' '10011100' '11101100'
  '11110000' '01110100' '01110010' '10101000' '00111100' '11110011'
  '11000110' '00001010' '01001101' '01000100']
 ['10010011' '01000100' '10101010' '11000001' '11100111' '01111010'
  '00110010' '01110100' '01011100' '01001101' '00100100' '10010011'
  '00110000' '10001000' '01000111' '01011001']
 ['00001010' '10011011' '11110101' '00010101' '00011001' '11101011'
  '10011110' '01101001' '01101101' '10110000' '10000110' '00001000'
  '01101001' '00011010' '10101001' '10000010']
 ['00001111' '00111010' '11001010' '10001000' '11110000' '01101001'
  '10111101' '01011011' '11111010' '00000111' '01111111' '11111011'
  '01110101' '11000111' '01100110' '10110000']
 ['01111111' '01010011' '10100100' '00001001' '11000110' '01110010'
  '11101000' '01010101' '00111101' '11011110' '01010111' '11011110'
  '01000010' '11101101' '00110110' '10111110']
 ['01010011' '00111000' '00000010' '11101101' '11000100' '10100100'
  '00000000' '11000011' '01001011' '00101100' '11110111' '10110110'
  '00011011' '11000011' '00011011' '01001000']
 ['01011100' '01111111' '00100110' '10011110' '10110101' '11000111'
  '00110001' '11000111' '10110101' '01100000' '01111011' '10100010'
  '01101010' '10001100' '00100011' '11111100']
 ['01011011' '10001000' '01000000' '00001111' '11000100' '01011111'
  '00110101' '00111011' '10110111' '10100101' '11000010' '10001101'
  '10010111' '10111100' '10011100' '11001000']
 ['00000001' '11000110' '00110011' '10111101' '10100011' '00000011'
  '00000101' '01001011' '01111101' '11101011' '01101001' '00011111'
  '00010111' '10011000' '01110011' '01010010']
 ['10110000' '01011111' '01111001' '10111000' '00010000' '10110010'
  '10110101' '11010100' '00010010' '01011110' '01011111' '10011100'
  '11001101' '10010111' '11100100' '11001011']
 ['00000001' '11011101' '11000010' '00001000' '10111100' '01101000'
  '00110000' '01100101' '10010100' '10111000' '01111011' '10001100'
  '11011001' '00110000' '00000111' '11011101']
 ['00000101' '01110101' '00001001' '00000110' '00000110' '10100000'
  '11101111' '00110010' '11111000' '10011111' '01111011' '11101011'
  '00010001' '00110100' '10110100' '01101101']
 ['10100011' '01000110' '00101100' '01100001' '11100111' '10110110'
  '00000000' '00100001' '00000111' '10010001' '10011111' '10110111'
  '10010100' '10001001' '00110000' '10111101']
 ['10101000' '11010000' '00111110' '11010110' '11010101' '00101110'
  '11011110' '00110111' '01110001' '11111000' '00000001' '10101111'
  '11001110' '11011110' '00000001' '01000111']]
```

SBOX[1]

```
[['11111100' '01110000' '00100101' '10001111' '00011100' '11101110'
  '11101110' '10101100' '11100111' '11111010' '10001100' '00101110'
  '11100111' '11111110' '11100001' '10101111']
 ['11110100' '11010100' '01111111' '01100010' '01100001' '10101010'
  '01011010' '01110011' '00000000' '01111111' '10000100' '01100001'
  '11000101' '10110100' '00101011' '10010101']
 ['10111000' '01011110' '10100000' '00001001' '00111101' '11101000'
  '00001110' '11011100' '10000011' '01111111' '11100110' '11000011'
  '11010010' '00110100' '11111000' '11110010']
 ['10010011' '10110101' '10111100' '01110011' '00010110' '00000000'
  '00101011' '10001010' '10011111' '00001001' '01100110' '00101000'
  '00000101' '11000101' '00110101' '00010101']
 ['10101100' '10000001' '11101100' '11101001' '11110000' '00111010'
  '01111011' '00011111' '00000010' '10110100' '11101010' '11000001'
  '10011111' '11000010' '00001110' '10011110']
 ['01000000' '00000111' '10100001' '10111001' '01001101' '01001100'
  '10110000' '01010010' '11100101' '10101110' '10001111' '01010010'
  '00011000' '01000110' '00111001' '11101110']
 ['11000110' '11101011' '00010110' '11011101' '01100000' '11100110'
  '01100111' '10001011' '10110011' '10100000' '01000001' '10001010'
  '11101101' '01101100' '10111100' '01001010']
 ['00111010' '10001111' '11000001' '01101101' '01001011' '01110001'
  '11011001' '01011110' '11001000' '00001111' '00000010' '01000101'
  '11111001' '11101000' '01110110' '11011011']
 ['01001101' '10000000' '00010010' '11000010' '01011001' '01010110'
  '11110100' '00010111' '01011010' '11100010' '10000101' '01010100'
  '01011110' '00110111' '11010010' '00111000']
 ['01011011' '01010100' '11111101' '11000101' '00101010' '00000010'
  '10101000' '01010000' '01010000' '11111011' '00110100' '11100000'
  '00100101' '10111110' '00011000' '11101001']
 ['10101011' '01100001' '01111110' '10101100' '11000100' '10110010'
  '00010111' '01000101' '01111001' '10011110' '10101010' '00001011'
  '10101100' '01110111' '10110111' '01010011']
 ['11010011' '10000000' '10010111' '01111011' '00100010' '00000001'
  '00100111' '10110100' '00000010' '10101000' '11000110' '00111101'
  '01111110' '01101010' '01101011' '00001101']
 ['11101111' '01110100' '11110001' '10011011' '00010000' '00100011'
  '00100010' '10111100' '00001100' '10000011' '00100011' '00000101'
  '10001100' '11011101' '01011101' '00001011']
 ['11001101' '11010001' '11100110' '10010110' '00111011' '00000011'
  '01111111' '01000011' '01110000' '00101010' '11000100' '01011100'
  '10000001' '10100110' '00000000' '11100001']
 ['00101001' '11001011' '10100111' '11000110' '11000011' '01001010'
  '11011011' '01010101' '11011100' '01101000' '10011010' '01000111'
  '11001011' '11100010' '11000100' '01101011']
 ['00110011' '11100100' '01001001' '10011101' '01110110' '11001010'
  '01110010' '11110000' '01101100' '11001111' '00100001' '11011100'
  '00110010' '00111011' '01010101' '11010100']]
```

SBOX[2]

```
[['11101001' '00000100' '10011010' '10100010' '10011010' '01110110'
  '11101101' '00101000' '11111100' '11110001' '10110010' '11010001'
  '10000001' '01111110' '00110010' '00011111']
 ['01110110' '01100011' '00100011' '00100000' '00000111' '01100100'
  '11010011' '00100000' '00011111' '10101011' '00111101' '01100010'
  '00111010' '00111010' '01000011' '10000010']
```

['11111010' '01010001' '00101011' '01000110' '01110000' '10011101'
 '11000101' '01000111' '01001101' '10110100' '01101110' '10101100'
 '01010110' '10000110' '01100001' '10110011']
['10001100' '00111110' '00110000' '11001110' '01000110' '11000111'
 '11110000' '01010010' '10011011' '11001100' '10010111' '11010010'
 '11010010' '00000001' '11011110' '11111000']
['01110000' '10010011' '01010111' '01001000' '01101100' '00100101'
 '10011011' '11111001' '01110101' '10000001' '00100111' '10011010'
 '01100100' '10001101' '01100111' '10101111']
['10000010' '01111110' '00110110' '00101101' '00101010' '11011111'
 '10100111' '10101111' '00101001' '01010001' '00111010' '01000110'
 '10110010' '00001000' '01100011' '11111101']
['11010010' '10010011' '01100001' '00100011' '11000111' '01101101'
 '11111110' '10101000' '10010110' '11101011' '00001110' '11101111'
 '01011100' '01011000' '11000110' '10110010']
['10001111' '01111100' '01100000' '00111000' '10111011' '01111000'
 '11010001' '01011111' '11111000' '00111100' '01010100' '11100000'
 '00000111' '10110111' '01110000' '01101001']
['11011111' '00101000' '01101001' '10011000' '00010110' '11101111'
 '10011010' '10000110' '11111011' '11110000' '00010110' '01000111'
 '11100010' '11111010' '11101110' '01000111']
['00010101' '00110110' '11010000' '00100100' '10100001' '01110000'
 '11010001' '00101010' '01010110' '11101011' '01010011' '10101111'
 '10110111' '00001011' '00101001' '10111100']
['00000001' '11101010' '00111100' '01000100' '01101011' '01000011'
 '10111011' '00101110' '10110111' '11001100' '10010100' '10010010'
 '01110111' '00011010' '10000010' '00001100']
['11010110' '11010100' '10011000' '01111110' '01001010' '10010010'
 '10010100' '11111110' '00011010' '11000111' '01110100' '01000010'
 '00111010' '10011000' '11101001' '01000001']
['00111001' '00000101' '10110010' '00110000' '10011101' '00011011'
 '01001111' '10011110' '01010011' '01101000' '11110110' '00001000'
 '01110001' '01001100' '00101001' '01111011']
['01011111' '00110010' '11111000' '01010000' '00110011' '10110010'
 '00011011' '11011010' '00011001' '01111110' '01010111' '00110011'
 '10010001' '01010110' '00110000' '01011001']
['11000010' '00111010' '11000110' '01111111' '01111100' '00110010'
 '01011000' '00111001' '01011101' '00100000' '01111100' '11011000'
 '01101011' '00010011' '11111010' '10101000']
['00001000' '10100010' '11111101' '10001101' '11110010' '11011100'
 '10111011' '01101010' '00001011' '01011110' '10011101' '11011001'
 '10000101' '11110101' '10101011' '11110100']]

SBOX[3]
[['10111100' '00110011' '01010100' '01110110' '10011011' '01011010'
 '01001111' '00010111' '01010001' '11000111' '00110001' '01000001'
 '11100110' '10110100' '01111100' '10100110']
['01110110' '01000101' '11010100' '11001110' '00001101' '11111000'
 '01110011' '11111011' '10110110' '01011101' '00011011' '10110100'
 '10111001' '11101001' '11000110' '11111000']
['11011111' '00001011' '10001101' '10100110' '01000000' '11100011'
 '00010101' '01110101' '10000111' '11100110' '01110000' '11101011'
 '00000101' '10111101' '00011101' '10110100']
['10100101' '11110111' '01001101' '00100111' '11101001' '11011000'
 '00101100' '10101010' '01011110' '10111011' '00110111' '10100110'
 '10001110' '10110011' '11000001' '00110001']
['10110100' '00111010' '00011010' '11101101' '10100110' '10000111'

```
 '10001101' '00111111' '11110000' '10011100' '00001000' '10110100'
 '10110111' '00011100' '11000111' '01011001']
['10101000' '00111000' '10100010' '00001010' '10001111' '11000000'
 '10011001' '01000000' '01001101' '00101111' '11011111' '00000011'
 '01010000' '10001010' '10001010' '01011101']
['01010010' '00101000' '00111010' '10011111' '11001011' '10010010'
 '10101101' '00111110' '10101101' '10101101' '01011111' '11111110'
 '00110100' '01100111' '00001001' '10101011']
['10010111' '01100100' '10000011' '00101100' '11000010' '00110111'
 '10000001' '10111111' '00111001' '11111010' '01101011' '01011010'
 '01001100' '10011101' '11101101' '10001001']
['00111010' '00000001' '01010000' '01011110' '11011100' '01100011'
 '01101000' '01011011' '10011100' '01001111' '00000111' '00010110'
 '10101100' '11101100' '00010111' '01100100']
['11011111' '01100100' '10111001' '00100100' '10110010' '11011110'
 '00100010' '10100111' '00010010' '01011110' '10011000' '01101100'
 '01001101' '11111110' '00011100' '00010001']
['11011001' '01000110' '01111100' '00101011' '00001101' '10010001'
 '01001001' '00111010' '00011010' '01011101' '10101100' '10000011'
 '10001011' '01101110' '01101110' '00101010']
['00101000' '11001100' '10011000' '11011100' '01101100' '10101011'
 '01000000' '10101000' '01000101' '00111000' '00010110' '00000001'
 '00111010' '01010110' '11011110' '11100001']
['10100111' '11011011' '11100001' '11101001' '00010001' '00101011'
 '01110000' '01011010' '00101011' '11010011' '01001101' '01011101'
 '01100000' '00110011' '01100110' '01110101']
['11101011' '01010100' '11101110' '00110111' '11111000' '10110000'
 '10001111' '10111001' '00111111' '10000111' '10100000' '00010111'
 '01100011' '11000101' '10000000' '01110100']
['01011001' '00110010' '00000111' '00111010' '01100111' '10001011'
 '01100001' '11101010' '10111110' '01111100' '10011111' '11010010'
 '00100101' '01011111' '10110100' '10111111']
['01000111' '10011010' '11011110' '01000011' '11000101' '11010110'
 '11000010' '00110110' '00001110' '00010101' '11011110' '01111111'
 '01110000' '00110010' '00010000' '11101101']]
```

Karena yang disubstitusi adalah setengah bagian kanan (R), maka ukurannya adalah 32 bit. Pertama blok yang terdiri dari 32 bit dibagi menjadi 4 bagian yang sama. Masing - masing 8 bit ini kemudian dibagi menjadi 2, 4 bit menjadi selektor baris dan 4 bit lainnya menjadi selektor kolom pada kotak S. Setiap 8 bit akan disubstitusi dengan 8 bit baru di dalam baris dan kolom yang sesuai. Setelah keempat bagian disubstitusi, bit-bit disatukan kembali menjadi 32 bit yang baru.

- Transposisi
  Setiap blok, setengah blok bagian kanan(R), yang terdiri dari 4 karakter (32 bit) ditransposisi dengan mengubah bit-bit menjadi matriks berukuran 4 x 8, lalu dilakukan transposisi seperti biasa.

- Pergeseran
  Bit - bit pada blok digeser sesuai dengan arah dan banyak step yang dipilih. Dalam algoritma ini kami menggunakan arah ke kanan dan step sebanyak 3.

- Cipher Berulang

Round dilakukan sebanyak 4 kali. Fungsi F(K,R) yang digunakan adalah fungsi XOR dan key yang digunakan di generate dari key yang dipakai di awal dan masing-masing round memiliki key yang berbeda-beda.

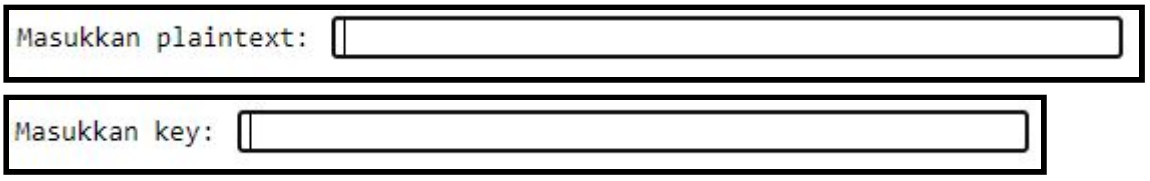
3.2. Dekripsi

Algoritma dekripsi mirip dengan algoritma enkripsi, karena menggunakan struktur Feistel Network. Hanya saja input pada fungsi dekripsi adalah ciphertext dan urutan round-round menjadi kebalikan dari enkripsi.

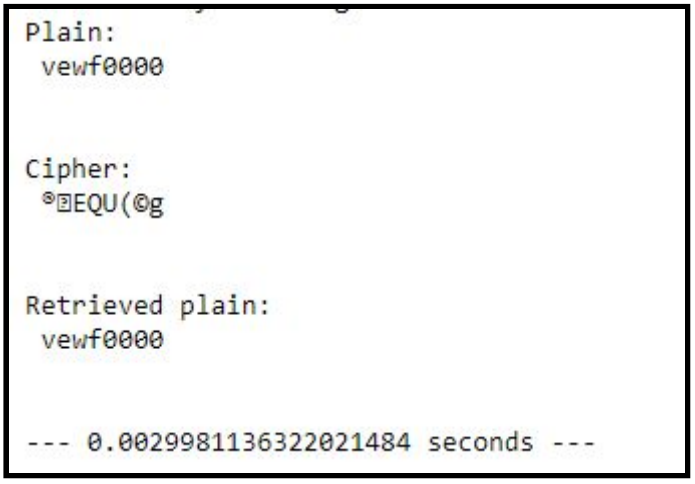
4. Eksperimen dan Hasil Analisis

4.1. Screenshot Program

Input:



Output:



4.2. EBC

```
Masukkan key: abcdefgh
Plain:
 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor
in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proiden
t, sunt in culpa qui officia deserunt mollit anim id est laborum.000


Cipher:
 ÜÇ^Ùz²Å¼z▯|Ïèß6¤<▯¾2Ö▯S>▯ï3Iní«½üû+▯0▯▯ôëÉ]%J¿gr/Ãä)¨æÂS&iÛ§\àA·▯ézäø½▯S▯?<^h▯î▯h▯▯ÖÔè▯Ïf-ºå9xÜ▯▯hù þYøD6iê}▯!v÷▯▯§DþÊÕÀen$%9
*àþ§º▯mB¼-▯+▯▯ð¡¾:ªÛ*▯i°M▯▯▯▯´]▯å%Î?¥Õ;▯¡ù,ß▯▯ö*Ò_Nº Ø▯(▯²zuà{©▯▯/▯`©%ö▯MÈ6y°ñ▯C²û g▯G°^y:f|C-e▯r▯¨õÅß.EK▯ÉVø<ïL▯:¬▯.▯s¢7▯Â&
Ä`!Ñ8▯Æ~▯▯Ñ▯+▯¨▯âî.°▯µG²ÛJ`▯+ð²▯▯â▯ £:▯]¼Îí[0e;¸▯óP▯sg6¶▯Cd¬åä÷M!▯Y|=ë0Y¾▯«W~Ð▯)°^.▯▯Ü&▯ï<3▯Ô▯³▯]Å¸▯▯¡Å¡Â


Retrieved plain:
 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor
in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proiden
t, sunt in culpa qui officia deserunt mollit anim id est laborum.000


--- 0.03497815132141113 seconds ---
```

4.3. CBC

```
Masukkan key: abcdefgh
Plain:
 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor
in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proiden
t, sunt in culpa qui officia deserunt mollit anim id est laborum.000


Cipher:
6▯      .▯ú&õÜ×Þ>2?Ñ▯QY¤ÌV▯▯▯uÇö▯▯0]▯ôç▯Pèý»©?ku)▯uö þ¾óQíwV6▯▯▯▯k▯x▯&þS÷▯¨▯Ö;▯|H▯[¿dXOi▯£é×[▯L'LÙZÉxÀ▯Ë▯ßR▯»▯r#▯·Ã;▯ð>{ges▯ÕëÆ#
=+▯7ö¿dqw0È .MR ¯▯▯Þ▯Q®º▯Aw▯▯J¸K¬▯▯ÓÁ÷▯▯▯?q@{N@W▯▯{▯▯â§ü▯ ▯¿@"▯▯▯P▯▯<▯@▯Eà▯ù▯ìúÅN▯ægF©


Retrieved plain:
 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor
in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proiden
t, sunt in culpa qui officia deserunt mollit anim id est laborum.000


--- 0.042975664138793945 seconds ---
```

4.4. Perubahan satu kunci pada CBC

Penggantian satu kunci dari “abcdefgh” menjadi “zbcdefgh” menghasilkan cipherteks yang sangat berbeda dari sebelumnya.

```
Masukkan key: zbcdefgh
Plain:
 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor
in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proiden
t, sunt in culpa qui officia deserunt mollit anim id est laborum.000


Cipher:
¿X?ýôrÄþ¾.Myh³Á□dj¿K□Nì□ô□□□ã□n)â□>ÔN:4cÕ_¥h2kÄ□□□X□±£G-1µ³õctaQ)□ë□þ□~ã□ß±.ãbÈb×â□□¯ê¿¦|ôÎÂi±
6□□.□ú&õÇxÞ>)$Ñ□QB¤×M□□□nÇí□0F□ôç□Kóý ©$pu2□nö å§óJíwM6□□□□p□c□&åHì□³□Ö □gH□[¤dXTi□¸ò×[□W<WÂZÉxÀ□Ë□ÄR□»□r#□·Ã;□ë>`g~s□Õëæ8&0□7ö
¤□dqw+Ó 5MR      ¯□□Þ□Jµº□Al□□Q¸P·□□Óáì□□□?j@{u□U@W□□{□□à§ç□ □¿["□□□P□□<□[□^à□ù□÷áÅU□ý|F²


Retrieved plain:
 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor
in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proiden
t, sunt in culpa qui officia deserunt mollit anim id est laborum.000


--- 0.034998178482055664 seconds ---
```

4.5. Perubahan satu bit plainteks pada CBC

Bit pada karakter pertama plainteks diubah dari “01001100”(L) menjadi “01001101”(M). Sama halnya dengan penggantian satu kunci, dihasilkan juga cipherteks yang jauh berbeda dibandingkan sebelum perubahan, meskipun hanya diubah satu bit.

```
Masukkan key: abcdefgh
Plain:
 Morem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor
in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proiden
t, sunt in culpa qui officia deserunt mollit anim id est laborum.000


Cipher:
¬       ~«\Å±tù*Ò*□ªFÝµ¢¥î.¦ÎÐiãE÷õ4Wy(³ÛQ,p¿KÂN¾\¼□³□£\n{ªÜ$Ô□ f+□□¿ hkÄ□NÚXÑ£ë□eyµá¼#tÿ3□)□¹
mýær□□×w□±Ô□*□□Äêë□¥□à□áµèYE□*ÙéÝú)Õmv
d1w1□h/MRI¯×Æ□□P¯ú]A>ÉÃK¸;L□£□j4bÉ□¼¾ÍÅ5.,R\ò\E6ÝHfÚ²nõ□□Ö>{v□□Q□ä□W□ÛltÇÿ□x\L¾¯ÕYéµºáv"uh□<¾è·õó□í?W~Æ□□Újyàn·□¶Ã©□□:Ô}@W
[ö,□□!
ÐÈ□©öÈÙ□?3uA□Í\{ÁÙ¢§µÉè□¿       *Ñ□□□Í<ÃALD¨U±Àû□OI¯.F


Retrieved plain:
 Morem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor
in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proiden
t, sunt in culpa qui officia deserunt mollit anim id est laborum.000


--- 0.032981157302856445 seconds ---
```

4.6. Perubahan satu bit cipherteks pada CBC

Setelah dilakukan enkripsi, hasil cipherteks diubah satu bit pada karakter ke 6. Hal ini mengakibatkan eror pada dekripsi karakter ke 6 sampai 1. Karakter terakhir sampai ke 7 tidak terpengaruh karena dekripsi dilakukan mulai dari belakang.

```
Plain:
 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor
in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proiden
t, sunt in culpa qui officia deserunt mollit anim id est laborum.000


Cipher:
6□      .□ú&ðÜ×Þ>2?Ñ□QY¤ÌV□□□uÇö□0]□ôç□Pèý»©?ku)□uö þ¾óQíwV6□□□□k□x□&þS÷□¨□Ö;□|H□[¿dXOi□£é×[□L'LÙZÉxÀ□Ë□ßR□»□r#□·Ã;□ð>{ges□Õë£#
=+□7ö¿dqw0È .MR ¯□□Þ□Q®º□Aw□□J¸K¬□□Óá÷□□□?q@{N@W□□{□□â§ü□ □¿@"□□□P□□<□@□Eà□ù□ìúÅN□ægF©


Retrieved plain:
 LwÆPm"ltsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor
in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proiden
t, sunt in culpa qui officia deserunt mollit anim id est laborum.000


--- 0.03997349739074707 seconds ---
```

5. Kesimpulan dan Saran Pengembangan

Feistel Network merupakan algoritma sederhana yang sangat bermanfaat, karena algoritma dekripsi sangat mirip dengan enkripsinya. Serumit apapun fungsi dan banyaknya round tidak masalah karena sifat reversibel dari feistel tersebut.
Penggantian bit pada kunci atau plainteks pada EBC tidak berpengaruh banyak pada cipherteks yang dihasilkan karena cipherteks diproses secara independen per bloknya, sedangkan CBC sebaliknya karena satu blok dengan blok sebelumnya dan sesudahnya berhubungan.

6. Referensi

[1] Zebua T, Ndruru E. (2017) "Pengamanan Citra Digital Berdasarkan Modifikasi Algoritma RC4". *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK)*. *4*. 275-282
[2] Sidik AP. (2019). "Teknik XOR pada Mode Operasi Algoritma Cipher Block Chaining (CBC) dengan Kunci Acak Blum Blum Shub dalam Meningkatkan Keamanan Data". Jurnal Mantik Penusa. 3. 2580-9741
[3] Setyaningsih E., Iswahyudi C., Widyastuti N. (2012). "Image Encryption on Mobile Phone using Super Encryption Algorithm". Telkomnika. 10. 837-845
[4] Sa'diyah H. (2018) "Implementasi Keamanan Pesan pada Citra Steganografi Menggunakan Modifikasi Cipher Block Chaining (CBC) Vigenere". Telematika. 13. 44-55
[5] Singh S. (2014). "Block Ciphers in Cryptography". *International Journal of Advanced Research in Computer Science*. 5