IMPLEMENTASI FEISTEL *BLOCK CIPHER* DALAM ENKRIPSI FILE BERBENTUK TEKS

[1] *Ety Sutanty*, [2] *Meilani B. Siregar*, [3]*Esti Setiyaningsih*
[1,2,3]*Fakultas Teknologi Industri Universitas Gunadarma*
Jl. Margonda Raya 100, Depok 16424, Jawa Barat
[1,2,3]*{ety_s, meilani, esti}@staff.gunadarma.ac.id*

Abstrak

Salah satu cara mengamankan file digital yang dikirimkan melalui jaringan komputer adalah kriptografi. Kriptografi melakukan pengamanan terhadap file digital yang dikirimkan dengan melakukan pengacakan informasi asli (plaintext) menggunakan kunci (key). Pada penelitian ini akan diimplementasikan salah satu metode dalam melakukan enkripsi terhadap file berbentuk Teks (txt, pdf, docx dan rtf) dengan maksimal input file digital berukuran 80 MB menggunakan Feistel Block Cipher dengan metode 3DES. Proses penyandian (enkripsi) algoritma DES diproses sebanyak 3 kali menggunakan 3 kunci dengan total kunci yang dibangkitkan menjadi 168 bit. Ketiga kunci yang digunakan dapat bersifat saling bebas, atau hanya 2 kunci saja yang saling bebas tergantung penggunaan yang dibutuhkan. Hasil pengujian menunjukkan ukuran file berpengaruh pada waktu yang dibutuhkan dalam proses enkripsi maupun dekripsi, selain itu proses yang berjalan pada sistem operasi juga mempengaruhi waktu pengenkripsian suatu file. Tabel hasil uji coba menunjukkan bahwa waktu yang dibutuhkan dalam proses enkripsi tidak jauh berbeda dengan waktu yang dibutuhkan dalam proses dekripsi pada file yang sama, yaitu memiliki selisih sebesar 0.03% dimana proses dekripsi lebih cepat daripada proses enkripsi.

***Kata Kunci**: Dekripsi, DES, Enkripsi, Feistel, Kunci*

Abstract

One way of sending digital files that are sent over a computer network is cryptography. Cryptography protects digital files sent by scrambling the original information (plaintext) using a key. In this study, one method will be implemented to encrypt files in the form of Text (txt, pdf, docx dan rtf) with a maximum input of 80 MB digital files using Feistel Block Cipher with the 3DES method. The DES algorithm's encryption (encryption) process becomes 3 times using 3 keys with a total key generated of 168 bits. The third key used can be independent, or only 2 keys are used depending on the required use. The test results show that the file size affects the time needed in the encryption and decryption process, besides that the process running on the operating system also affects the encryption time of a file. The table of test results shows that the time required for the encryption process is not much different from the time required for the decryption process on the same file, which has a difference of 0.03% where the decryption process is faster than the encryption process.

***Keywords**: Decrypt, DES, Encrypt, Feistel, Key, Video*

PENDAHULUAN

Salah satu metode dalam melakukan pengamanan terhadap file digital yang dapat dikirimkan melalui jalur komunikasi [1] adalah metode kriptografi. Metode ini tidak mencegah terjadinya pencurian data [2] melainkan membuat data yang dicuri menjadi

data yang tidak dapat dimengerti oleh pencurinya [3]. Tingkat keamanan kriptografi dalam mengamankan sebuah data bergantung pada kerahasiaan kuncinya [4]. Dalam kriptografi terdapat berbagai algoritma dengan tingkat komputasi dan aturan yang berbeda-beda yang dapat diimplementasikan pada file digital berbentuk teks, salah satunya dengan menggunakan Feistel Block Cipher. Jaringan Feistel membutuhkan inputan 64-bit hasil permutasi awal dan 16 sub-kunci yang masing-masing berukuran 48-bit yang biasanya terdapat pada algoritma 3DES (*Data Encryption Standart*).. Algoritma 3DES menggunakan 3 kunci dengan total kunci yang dibangkitkan menjadi 168 bit. Ketiga kunci yang digunakan dapat bersifat saling bebas, atau hanya 2 kunci saja yang saling bebas tergantung penggunaan yang dibutuhkan. Penelitian terkait implementasi Jaringan Feistel dalam melakukan enkripsi terhadap file digital dilakukan peneliti terdahulu. Penelitian [5] menggunakan pemrograman VHDL (VHSIC *Hardware Description Language*) untuk mengimplementasikan Feistel Block Cipher dengan algoritma 3DES dalam chip FPGA. Implementasi 3DES dalam sebuah chip FPGA membutuhkan banyak gerbang-gerbang digital, terutama karena operasinya yang 64 bit. Fungsi utama dari FPGA sendiri adalah dalam pengkodean dan pendekodean pesan yang dikirim atau diterima oleh mikrokontroller. Algoritma 3DES dimplementasikan ke dalam FPGA dengan cara dibagi-bagi menjadi modul-modul yang dibuat menggunakan VHDL. Penelitian [5] menggunakan tiga modul utama yang membutuhkan *resource* FPGA cukup besar antara lain *rounding* untuk implementasi F-*function*, penjadwalan kunci (*key schedulling*) pada tahap *preprocessing* dan control (termasuk *multiplexer*) [6]. Penelitian dengan mengimplementasikan penggunaan jaringan Feistel pada algoritma 3DES digunakan peneliti [7] dengan menggunakan verilog yang merupakan salah satu bahasa HDL (*Hardware Description Language*). Implementasi jaringan Fesital dengan penggunaan algoritma 3DES dilakukan peneliti [8] dengan mengimplementasikan pada suatu aplikasi yang dapat mengenkripsi sekaligus mendekripsikan pesan atau file. Aplikasi dibuat dengan mengunakan alat bantu Matlab, Aplikasi tersebut dioperasikan untuk mengetahui waktu yang dibutuhkan untuk melakukan proses enkripsi dan dekripsi. Penelitian ini juga melakukan uji keamanan algoritma 3DES dari serangan *brute force*. Peneliti lain mengkombinasikan 3DES dengan Base64 sebagai algoritma pengamanan pada proses *handshaking* suatu aplikasi Delphi yang diimplementasikannya pada suatu CV tertentu [9]. Selain itu, jaringan Feistel yang diimplementasikan pada algoritma 3DES tidak hanya dimplementasikan namun juga diuji oleh beberapa peneliti dengan cara membandingkan algortima 3DES dengan algoritma simetri lainnya [10].

Pada penelitian ini, diimplementasikan penggunaan Feistel Block Cipher untuk melakukan enkripsi pada File Teks. Proses enkripsi dilakukan setelah proses inisialisasi permutasi selesai. Plaintext berupa file digital berbentuk Teks dibagi menjadi dua bagian yaitu L0 dan R0 dimana kedua bagian berukuran 32-bit akan diproses sebanyak 16 kali/putaran. Implementasi jaringan feistel pada enkripsi file teks ini dapat dikembalikan seperti semula (dekripsi) walaupun diproses dengan banyak iterasi sehingga pihak penerima pesan dapat memahami pesan asli.

METODE PENELITIAN

Pada penelitian ini digunakan File teks, seperti : .txt, .pdf, .docx dan .rtf. Penelitian ini menggunakan data (pesan) dengan ukuran minimal 64-bit dan sebuah kunci simetri berukuran 56-bit. Ukuran data minimal tersebut merupakan satu blok data [11] yang nantinya akan diproses dengan menggunakan Feistel Block

A. Pembangkitan Kunci (*Key Generation*)

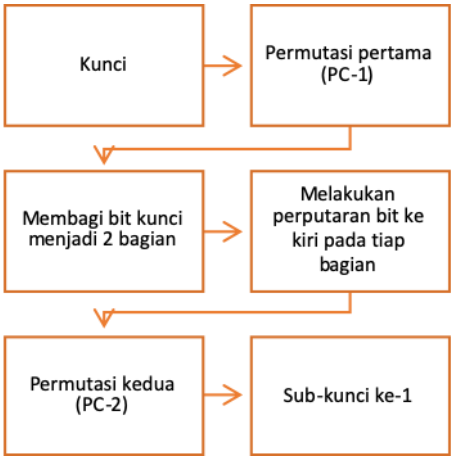
Data *plaintext* diinput bersama dengan kunci enkripsi. Kunci enkripsi yang terdiri 8 karakter akan dikonversi ke dalam satuan biner. Delapan karakter jika dikonversikan ke dalam byte menjadi 8 byte dan dalam bit menjadi 64-bit (yang dibutuhkan sebesar 56-bit). Ukuran minimal *plaintext* yang dapat dienkripsi sebesar 64-bit atau sebanyak 8 karakter. Ukuran minimal ini merupakan satu blok data yang dapat dienkripsi dimana jika ukuran plaintext lebih besar maka plaintext akan dibagi menjadi beberapa blok data [12], sedangkan jika ukuran *plaintext* lebih kecil maka *plaintext* akan ditambah dengan bit kosong hingga berukuran 64-bit. Penambahan karakter "spasi" dilakukan untuk menyesuaikan ukuran plaintext. Jumlah karakter "spasi" yang ditambahkan sesuai dengan jumlah kekurangan karakter (jkk) dari kelipatan 8, jika dinotasikan seperti persamaan 1.

$$jkk = 8 - lenP \ mod 8 \quad (1)$$

Penambahan karakter dilakukan dengan menggunakan karakter "spasi" karena karakter "spasi" dianggap tidak menimbulkan banyak perubahan pada file hasil dekripsi.

B. Pembentukan Sub-Kunci

Kunci enkripsi akan dibagi menjadi 16 sub-kunci dengan metode key schedule yang dapat dilihat pada Gambar 1.



Gambar 1. Bagan Pembuatan Sub-kunci

Kunci 64-bit akan diubah menjadi 16 sub-kunci yang masing-masing berukuran 48-bit. Penjelasan pembuatan sub-kunci sebagai berikut:

1. Pada setiap bit ke-8 dan kelipatannya pada kunci yang merupakan *odd parity* bit tidak diproses oleh algoritma, sehingga kunci yang digunakan sebesar 56-bit.
2. Selanjutnya sejumlah 56-bit dimasukkan pada tabel permutasi pertama(PC-1). Permutasi dilakukan untuk menukar bit dengan bit lain sesuai dengan tabel permutasi.

   pc1 = [56, 48, 40, 32, 24, 16, 8,
   0, 57, 49, 41, 33, 25, 17,
   9, 1, 58, 50, 42, 34, 26,
   18, 10, 2, 59, 51, 43, 35,
   62, 54, 46, 38, 30, 22, 14,
   6, 61, 53, 45, 37, 29, 21,
   13, 5, 60, 52, 44, 36, 28,
   20, 12, 4, 27, 19, 11, 3]

   Tabel permutasi menunjukan bahwa bit pertama (ke-0 jika index dimulai dari 0) pada hasil PC-1 adalah bit ke-56 pada kunci sebelum dilakukan permutasi pertama, begitu juga seterusnya.
3. Hasil PC-1 dibagi menjadi 2 bagian, bagian kiri dan kanan yang masing-masing berukuran 28 bit.
4. Melakukan proses perputaran bit ke kiri sejumlah 1 atau 2 bit pada tiap bagian sesuai dengan putaran sub-kuncinya [13]. Aturan perputaran bit ke kiri sebagai berikut:
   a. Pada putaran ke-i = 1,2,9,16 ; kedua bagian tersebut melakukan perpindahan bit ke kiri sebanyak satu bit.
   b. Pada putaran lainnya (selain i = 1,2,9,16) kedua bagian tersebut melakukan perpindahan bit ke kiri sebanyak dua bit.

   Atau dapat dituliskan sesuai dengan urutannya:

   left_rotations = [1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1]
5. Melakukan permutasi kedua (PC-2) pada hasil perputaran bit dengan tabel permutasi

kedua untuk mendapatkan nilai sub-kunci yang berukuran 48-bit.

pc2 = [13, 16, 10, 23, 0, 4,
2, 27, 14, 5, 20, 9,
22, 18, 11, 3, 25, 7,
15, 6, 26, 19, 12, 1,
40, 51, 30, 36, 46, 54,
29, 39, 50, 44, 32, 47,
43, 48, 38, 55, 33, 52,
45, 41, 49, 35, 28, 31]

6. Proses diatas terus dilakukan sebanyak 16 kali, sehingga menghasilkan 16 sub-kunci yang nantinya akan digunakan pada proses enkripsi pada iterasi masing-masing sub-kunci.

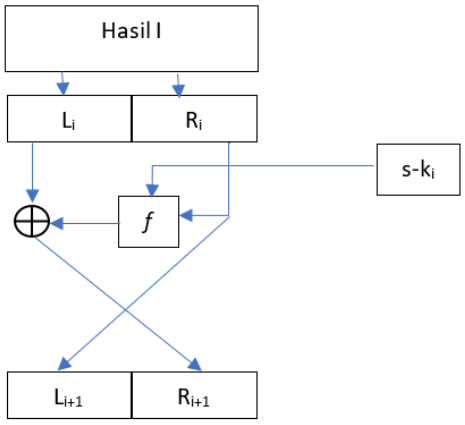
C. Permutasi Awal

Proses permutasi kembali dibutuhkan untuk menyelesaikan algoritma DES. Perbedaan proses permutasi awal atau biasa disebut initial permutation (IP) berada pada tabel pertukaran bit [14] yang digunakan dan IP tidak melakukan permutasi pada kunci melainkan pada plaintext yang ingin dienkripsi. Inputan pada proses IP sebesar 64-bit dengan keluaran yang sama yaitu 64-bit.

ip = [57, 49, 41, 33, 25, 17, 9, 1,
59, 51, 43, 35, 27, 19, 11, 3,
61, 53, 45, 37, 29, 21, 13, 5,
63, 55, 47, 39, 31, 23, 15, 7,
56, 48, 40, 32, 24, 16, 8, 0,
58, 50, 42, 34, 26, 18, 10, 2,
60, 52, 44, 36, 28, 20, 12, 4,
62, 54, 46, 38, 30, 22, 14, 6]

D. Enkripsi dengan Jaringan Feistel

Enkripsi merupakan proses merubah *plaintext* menjadi *ciphertext.* Penelitian ini mengimplementasikan penggunaan jaringan feistel agar file yang telah dienkripsi dapat dikembalikan seperti semula [15] walaupun diproses dengan banyak iterasi seperti dapat dilihat pada Gambar 2.



Gambar 2. Bagan Jaringan Feistel

Jaringan Feistel membutuhkan inputan 64-bit hasil permutasi awal dan 16 sub-kunci yang masing-masing berukuran 48-bit. Berikut langkah penyelesaian dengan menggunakan jaringan feistel:

1. Membagi hasil permutasi awal menjadi 2 bagian, bagian kiri (L_i) dan kanan (R_i) yang masing-masing berukuran 32-bit.
2. Menyalin bagian kanan untuk menjadi bagian kiri iterasi berikutnya (L_{i+1}).
3. Menyisipkan bagian kanan bersamaan dengan sub-kunci kedalam fungsi f pada DES.
4. Melakukan operasi XOR pada hasil fungsi f dengan bagian kiri (L_i) untuk menjadi bagian kanan pada iterasi berikutnya (R_{i+1}).
5. Mengulangi langkah ke-2 sampai ke-4 hingga iterasi ke-16.

Tahap penyelesaian fungsi f pada DES sebagai berikut:

1. Melakukan *expansion* pada plaintext bagian kanan (R_i) 32 bit menjadi 48 bit. Expansion dilakukkan dengan melakukan permutasi sesuai dengan tabel permutasi E.

```
expansion_table = [  31,  0,  1,  2,  3,  4,
                     3,  4,  5,  6,  7,  8,
                     7,  8,  9, 10, 11, 12,
                     11, 12, 13, 14, 15, 16,
                     15, 16, 17, 18, 19, 20,
                     19, 20, 21, 22, 23, 24,
                     23, 24, 25, 26, 27, 28,
                     27, 28, 29, 30, 31,  0      ]
```

2. Hasil permutasi E di operasikan dengan sub-kunci dengan menggunakan operasi XOR.
3. Hasil operasi XOR akan dibagi menjadi 8 bagian yang masing-masing berukuran 6-bit.
4. Tiap bagian akan dilakukan substitusi sesuai dengan tabel S-box-nya masing-masing sehingga setiap bagian menghasilkan 4-bit data.

```
sbox =
 [# S1
[14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0,
7,
 0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3,
8,
 4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5,
0,
 15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6,
13],
# S2
[15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5,
10,
 3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11,
5,
 0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2,
15,
 13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14,
9],
      ..
      ..
      ..
# S8
[13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12,
7,
 1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9,
2,
```

7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8,
2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11],
]

5. Kedelapan bagian yang ukurannya telah menjadi 4-bit digabungkan kembali menjadi 32 bit lalu dilakukan permutasi P.

p = [15, 6, 19, 20, 28, 11,
27, 16, 0, 14, 22, 25,
4, 17, 30, 9, 1, 7,
23,13, 31, 26, 2, 8,
18, 12, 29, 5, 21, 10,
3, 24]

E. Permutasi akhir

Permutasi akhir (IP^{-1}) dilakukan setelah jaringan Feistel diproses sebanyak 16 iterasi. Permutasi dilakukan sesuai dengan tabel permutasi akhir (*Final Permutation*).

fp = [39, 7, 47, 15, 55, 23, 63, 31,
38, 6, 46, 14, 54, 22, 62, 30,
37, 5, 45, 13, 53, 21, 61, 29,
36, 4, 44, 12, 52, 20, 60, 28,
35, 3, 43, 11, 51, 19, 59, 27,
34, 2, 42, 10, 50, 18, 58, 26,
33, 1, 41, 9, 49, 17, 57, 25,
32, 0, 40, 8, 48, 16, 56, 24]

Hasil permutasi akhir berupa 64-bit data yang telah diacak dalam bentuk biner, sehingga ketika data biner tersebut diubah menjadi bentuk karakter maka karakter yang didapat merupakan karakter yang berbeda.

HASIL DAN PEMBAHASAN

Studi kasus dilakukan sebagai bahan uji penelusuran alur proses menggunakan Feistel Block Cipher. Sebagai contoh aplikasi kriptografi akan mengenkripsi plaintext “pandemi corona” dengan 3 kunci simetri “enkripsi”, “dekripsi”, “shenozar”.

1. Aplikasi akan menghitung jumlah karakter plaintext. Jumlah karakter pada plaintext “pandemicorona” adalah 13 karakter. Plaintext ini tidak berukuran kelipatan 8 karakter/64-bit. Algoritma tidak dapat memproses plaintext ini, maka aplikasi menambahkan karakter spasi sesuai dengan jumlah karakter yang kurang. Jumlah kekurangan karakter (jkk) didapat dari notasi dibawah ini.

   jkk = 8 – jumlah_karakter modulus 8

   = 8 – 13 mod 8 = 8 – 5 = 3 karakter

   Sehingga plaintext saat ini menjadi “pandemicorona ”
2. Aplikasi akan mengkonversi inputan baik plaintext maupun kunci ke dalam satuan biner.

   Plaintext “pandemicorona” = 01110011 01101000 01100001 01110011 01100001 00100000 01110010 01100001 01100111 01100001 01111010 01111010 01101001 00100000 00100000 00100000

   Kunci-1 “enkripsi” = 01100101 01101110 01101011 01110010 01101001 01110000 01110011 01101001

Kunci-2 “dekripsi” = 01100100 01100101 01101011 01110010 01101001 01110000 01110011 01101001

Kunci-3 “shenozar” = 01110011 01101000 01100101 01101110 01101111 01111010 01100001 01110010

3. Aplikasi melakukan menentukan sub-kunci pertama hingga ke-16 untuk kunci-1 “enkripsi”
   a. Permutasi pertama (PC-1)

      Permutasi awal membuat 64-bit kunci menjadi 56-bit.

      Hasil PC-1 = [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1]
   b. Membagi bit menjadi 2 bagian

      Bagian L 28-bit = [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0]

      Bagian R 28-bit = [0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0]
   c. Perputaran bit

      Perputaran bit ke kiri untuk sub-kunci pertama berjumlah 1 putaran.

      Bagian L 28-bit = [0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0]

      Bagian R 28-bit = [1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0]
   d. Permutasi kedua (PC-2)

      Kedua bagian disatukan kembali lalu dilakukan permutasi ke-2.

      Hasil PC-2 = [1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1] = sub-kunci pertama.
   e. Perputaran bit kembali dilakukan untuk mendapatkan sub-kunci lainnya. Berikut sub-kunci ke-2 hingga ke-16.

      Sub-kunci ke-2 = [1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1]

      Sub-kunci ke-3 = [1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0]

      ….

      …..

      Sub-kunci ke-16 = [1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0]
4. Setelah sub-kunci didapat akan mulai mengenkripsi plaintext. sebelum plaintext dipermutasi dengan IP, plaintext akan dibagi menjadi per-blok yang tiap bloknya berjumlah 8 karakter = “pandemic”

   “pandemic” = 01110011 01101000 01100001 01110011 01100001 00100000 01110010 01100001

a. Plaintext dipermutasi dengan IP

   IP = [1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1]

b. Hasil IP dibagi menjadi 2 bagian

   Bagian L 32-bit = [1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1]

   Bagian R 32-bit = [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1]

c. Bagian R_i menjadi L_{i+1}.

   Bagian L_2 32-bit = [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1]

d. Bagian R_i bersama sub-kunci ke-i dimasukkan ke dalam fungsi *f*.

   i. Bagian R dipermutasi dengan tabel e-box (expansion table)

      Hasil e-box 48-bit = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0]

   ii. Bagian R XOR sub-kunci ke-1

      Hasil XOR 48-bit = [0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1]

   iii. Hasil XOR dibagi 8 bagian, yang setiap bagian dimasukkan ke dalam tiap 8 s-box

      Hasil XOR dibagi 8 bagian = [0, 1, 1, 0, 0, 0], [0, 0, 1, 0, 1, 0], [1, 0, 0, 1, 0, 0], [0, 1, 1, 0, 0, 0], [1, 1, 0, 0, 1, 0], [0, 0, 0, 1, 1, 0], [1, 1, 1, 0, 1, 1], [0, 0, 1, 0, 0, 1]

   iv. Hasil S-box disatukan kembali lalu dilakukan permutasi P menjadi hasil fungsi f

      Hasil S-box = [5] [11] [4] [11] [9] [15] [2] [10]

      Dalam biner = [0101], [1011], [0100], [1011], [1001], [1111], [0010], [1010]

      Hasil permutasi P 32-bit = [1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0]

e. Hasil fungsi f dioperasikan dengan bagian L_i menggunakan operasi XOR menjadi R_{i+1}.

   Bagian R_2 32-bit = [0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1]

f. L2 dan R2 telah didapatkan. Proses terus diulang hingga iterasi ke-16 menggunakan sub-kunci sesuai iterasinya.

g. L16 dan R16 disatukan untuk dilakukan permutasi akhir (IP^{-1})

   Hasil IP^{-1} = [1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0]

5. Ciphertext untuk plaintext “pandemic” telah ditemukan (Hasil IP^{-1}).
6. Blok plaintext lainnya “orona” lalu diproses dengan menggunakan langkah-langkah *enciphering* yang sama.

   Ciphertext blok-2 = [0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0]
7. Proses enkripsi ke-1 telah selesai diproses. Kini aplikasi akan mengenkripsi hasil enkripsi ke-1 ke dalam proses dekripsi ke-2 dengan menggunakan kunci ke-2 “dekripsi”(proses pembentukan sub-kunci juga kembali dilakukan).

   Ciphertext ke-2 blok ke-1 = [0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1]

   Ciphertext ke-2 blok ke-2 = [1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1]
8. Hasil enkripsi ke-2 lalu dienkripsi kembali menggunakan kunci ke-3 “shenozar”.

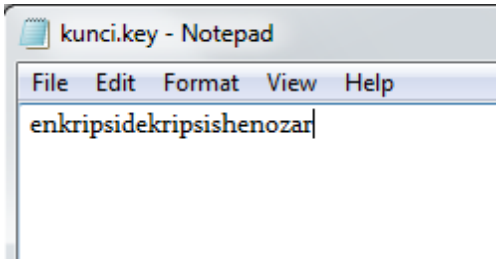
   Ciphertext ke-3 blok ke-1 = [1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0]

   Ciphertext ke-3 blok ke-2 = [0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1]
9. Ciphertext implementasi Feistel Block Cipher telah didapatkan. Mode operasi yang digunakan adalah mode ECB (*Electronic Codebook Mode*) maka tiap blok hanya perlu disatukan tanpa melalui proses tertentu.

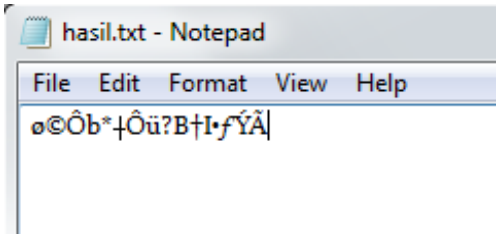
   Plaintext = “pandemicorona ”

   Ciphertext dalam biner = [1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1]

Keluaran *ciphertext* dikonversi ke dalam biner agar bisa dicocokkan dengan hasil perhitungan dapat dilihat pada Gambar 3 dan Gambar 4. Gambar 4 menunjukkan hasil ciphertext pada studi kasus dengan Teks: pandemicorona. Gambar 5 menunjukkan hasil konversi cipherteks kedalam notasi biner. Implementasi terhadap Feistel Block Cipher juga diuji dengan cara menginput beberapa macam ekstensi file teks yang berbeda, mulai dari dari file *plaintext*(.txt), file *format-text*(.docx, .pdf, .rtf) seperti dapat dilihat pada Tabel 1.



Gambar 3. Kunci Feistel Block pada Studi Kasus



Gambar 4. *Ciphertext* studi kasus

```
IDLE 2.6.6      ==== No Subprocess ====
>>> a='ø©Ôb*†Ôü?B†I•ƒÝÃ'
>>> ' '.join(format(ord(i),'b').zfill(8) for i in a)
'11111000 10101001 11010100 01100010 00101010 00010000 110101
00 11111100 00111111 01000010 10000110 01001001 10010101 1000
0011 11011101 11000011'
>>>
```

Gambar 5. Konversi *Ciphertext* ke dalam Biner

Tabel 1. Hasil Uji Coba Aplikasi pada Berbagai File Digital

No	Nama file	Ukuran file	Waktu Proses Enkripsi (s)	Waktu Proses Dekripsi (s)
1.	nama.txt	1 kb (13 karakter)	0.00058	0.00048
2.	teks2.txt	1 kb (455 karakter)	0.00077	0.00075
3.	teks3.txt	3 kb (2716 karakter)	0.00212	0.00213
4.	doc1.docx	13 kb	0.00834	0.00830
5.	jarkom1.docx	494 kb	0.29117	0.28510
6.	pweb7.docx	1.301 kb	0.81959	0.74977
7.	ebook4.pdf	625 kb	0.35708	0.36676
8.	ebook5.pdf	2.503 kb	1.43324	1.41381
9.	ebook9.pdf	4.213 kb	2.44992	2.39780
10.	rtf1.rtf	45 kb	0.02525	0.02729

Seperti dapat dilihat pada Tabel 1 hasil ujicoba enkripsi terhadap beberapa input file digital, terdapat perbedaan waktu enkripsi pada file yang ukurannya berbeda. Semakin besar file, maka waktu proses enkripsi berlangsung akan semakin lama. Terdapat selisih waktu 0.03% antara proses dekripsi

menjadi pesan asli dengan proses enkripsi pesan asli.

KESIMPULAN DAN SARAN

Implementasi Feistel Block Cipher berhasil melakukan enkripsi terhadap file plainteks berbentuk teks. Ukuran file berpengaruh pada waktu yang dibutuhkan dalam proses enkripsi maupun dekripsi, selain itu proses yang berjalan pada sistem operasi juga mempengaruhi waktu pengenkripsian suatu file. Tabel hasil uji coba menunjukkan bahwa waktu yang dibutuhkan dalam proses enkripsi tidak jauh berbeda dengan waktu yang dibutuhkan dalam proses dekripsi pada file yang sama, yaitu memiliki selisih sebesar 0.03% dimana proses dekripsi lebih cepat daripada proses enkripsi.

Pengembangan dapat dilakukan dengan menambahkan fitur/fungsi baru seperti pemberitahuan ketika file input tidak sesuai (ukuran file melebihi 70MB) serta penggabungan antara metode simetris dan asimetris untuk mendapatkan ciphetext yang benar-benar aman.

DAFTAR PUSTAKA

[1] L.Y. Zhang, Y. Liu, F. Pareschi, Y. Zhang, K.W. Wong, R. Rovatti, and G. Setti, "On the security of a class of diffusion mechanisms for image encryption," IEEE Transactions on Cybernetics, vol. 48, no. 4, pp. 1-13, 2017

[2] X. Zhang, Y. Niu, C. Shen, and G. Cui, "Fluorescence resonance energy transfer-based photonic circuits using single-stranded tile self-assembly and DNA strand displacement," *Journal of Nanoscience and Nanotechnology*, vol. 17, no. 2, pp. 1053–1060, 2017

[3] J. P. L. Cox, "Long-term data storage in DNA," *Trends in Biotechnology*, vol. 19, no. 7, pp. 247-250, 2001

[4] X. Zhang, Y. Wang, G. Cui, Y. Niu, and J. Xu, "Application of a novel IWO to the design of encoding sequences for DNA computing," *Computers and Mathematics with audioApplications*, vol. 57, no. 11-12, pp. 2001–2008, 2009

[5] Y. Erlich, and D. Zielinski, "DNA fountain enables a robust and efficient storage architecture," Science, vol. 355, no. 6328, pp. 950-954, 2016

[6] S.P. Venigalla, M N. Babu, S. Boddu, and G.S.S. Vemana, "Implementation Of The Triple-Desblock Cipher Using Vhdl", *International Journal Of Advances In Engineering & Technology (IJAET*), India. ISSN: 22311963117. Vol. 3, Issue 1, Pp. 117-128, 2012

[7] M. S.Narula dan S. Singh, "Implementation Of Triple Data Encryption Standard Using Verilog", *International Journal Of Advanced Research In Computer Science And*

Software Engineering (IJARCSSE), Page 667 Volume 4 , Issue 1. ISSN: 2277 128X, 2014.

[8] A. Hidayat, *Enkripsi dan Dekripsi Data dengan Algoritma 3DES*, Bandung: Universitas Padjadjaran, 2008.

[9] Rahim, M. Abdul, *Implementasi Kombinasi Algoritma 3DES dan Algoritma Base64 pada Sistem Keamanan Handshaking Animation Store di CV*. Edukreasi, Semarang: Universitas Dian Nuswantoro, 2013

[10] H.O. Alanazi, B.B.Zaidan, A.A.Zaidan, H.A. Jalab, M.S., and Y.Al-Nabhani. "New Comparative Study Between DES, 3DES, and AES within Nine Factors", *Journal Of Computing*. Page 152 Volume 2, Issue 3. ISSN: 2151-9617, 2010

[11] W. Stallings, *Cryptography and Network Security*, Principles and Practice, Third Edition, Pearson, 2003.

[12] H. Feistel, "Cryptography and Computer Privacy", *Scientific American*, Vol. 228, No.5, pp. 15-23, 1973.

[13] V.U.K. Sastry and K. Anup Kumar, "A Modified Feistel Cipher involving a key as a multiplicant on both the sides of the Plaintext matrix and supplemented with Mixing, Permutation and XOR Operation", *International Journal of Computer Technology and Applications*, ISSN 2229-6093,Vol 3 (1), pp, 23-31 , 2012.

[14] V.U.K. Sastry and K. Anup Kumar, "A Modified Feistel Cipher involving a key as a multiplicant on both the sides of the Plaintext matrix and supplemented with Mixing, Permutation and Modular Arithmetic Addition", *International Journal of Computer Technology and Applications*, ISSN 2229-6093,Vol 3 (1), pp, 32-39 , 2012.

[15] V.U.K. Sastry and K. Anup Kumar, "A Modified Feistel Cipher Involving a Key as a Multiplicant on Both the Sides of the Plaintext Matrix and Supplemented with Mixing, Permutation, and Modular Arithmetic Addition", *International Journal of Computer Science and Information Technologies* , ISSN 0975 – 9646. Vol. 3 (1) , 2012, pp, 3133 – 3141,2012.