Algoritma *Cipher* Blok YDIHYLFSIPFIF

Asif Hummam Rais[1], **Juniardi Akbar**[2].

Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika (STEI), Institut Teknologi Bandung (ITB), Jalan Ganesha 10, Bandung 40132
[1] E-mail: 13517099@std.stei.itb.ac.id, hashshura@gmail.com
[2] E-mail: 13517075@std.stei.itb.ac.id, juniardiakbar@gmail.com

Abstrak. Masalah keamanan informasi merupakan isu penting di era berkembangnya teknologi informasi. Kriptografi merupakan ilmu yang memiliki peran penting pada bidang keamanan teknologi informasi. Pada perkembangannya, algoritma kriptografi juga turut mengalami perbaikan dari yang sebelumnya proses manipulasi dilakukan pada karakter menjadi manipulasi pada bit. Proses manipulasi ini dapat dilakukan dengan membagi pesan ke dalam blok-blok bit yang disebut dengan *cipher* blok. Pada makalah ini, penulis akan menjelaskan sebuah algoritma *cipher* blok baru yang disebut YDIHYLFSIPFIF (baca: *yedih-yelf-sip-fif*)—*Yo Dawg I Heard You Like Feistel, So I Put Feistel Inside Feistel*. Algoritma ini pada dasarnya akan mengubah struktur jaringan feistel dengan menambahkan sebuah jaringan feistel ke dalam sebuah komponen pada jaringan feistel. Hal ini bertujuan agar kata dalam plainteks dapat semakin teracak. **Keywords**: kriptografi, *cipher* blok, kunci simetri, *block cipher variation*.

1. Pendahuluan

Kriptografi adalah studi tentang penerapan teknik matematika dalam yang berhubungan terhadap aspek keamanan informasi seperti *confidentiality* (kerahasiaan), *data integrity* (kebenaran), *authentication* (identifikasi), dan *non-repudiation* (anti penyangkalan). Tujuan mendasar dari kriptografi adalah menerapkan keempat bidang ini secara memadai baik dalam teori maupun praktik. Seiring perkembangan, fungsi dari algoritma kriptografi kian meluas sehingga tidak hanya berfungsi sebagai metode untuk menyembunyikan pesan, tetapi juga upaya pencegahan dan deteksi kecurangan atau aktivitas jahat lainnya[1].

Pada perkembangannya, algoritma kriptografi dapat dibedakan menjadi dua era yaitu kriptografi klasik dan kriptografi modern. Kriptografi klasik melakukan manipulasi pada basis karakter, yang artinya proses enkripsi dan dekripsi dilakukan terhadap karakter dalam pesan. Adapun kriptografi modern melakukan manipulasi pada basis biner, yang artinya informasi dalam bentuk apapun dapat dienkripsi asalkan direpresentasikan dalam bentuk biner.

Pada kriptografi modern, karena beroperasi dengan mode bit, maka semua data dalam informasi (kunci, plainteks, maupun cipherteks) akan dinyatakan dalam rangkaian bit biner. Sehingga rangkaian bit yang menyatakan plainteks akan dienkripsi menjadi cipherteks dalam bentuk rangkain bit, begitu pula sebaliknya. Operasi manipulasi yang akan banyak digunakan pada algoritma kriptografi modern adalah XOR dan *shift register*.

Pada algoritma kriptografi modern, proses enkripsi dan dekripsi terhadap data digital dapat dikelompokkan menjadi dua kategori yaitu *cipher* alir (*stream cipher*) dan *cipher* blok (*block cipher*).

Perbedaan yang mendasar dari keduanya adalah bentuk bit yang dioperasikan. *Cipher* alir beroperasi pada bit tunggal sedangkan *cipher* blok beroperasi pada blok bit.

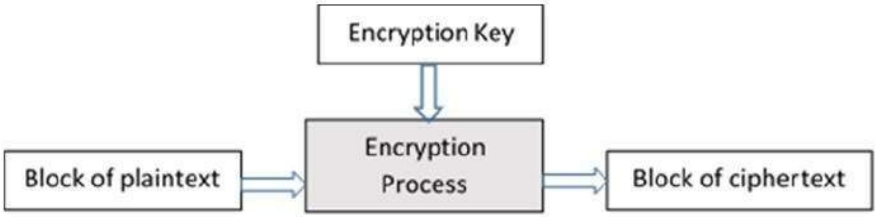
Implementasi algoritma *cipher* alir dan *cipher* blok sudah banyak dipublikasikan. Tentu saja setiap algoritma memiliki kelebihan dan kekurangan masing-masing. Bahkan beberapa algoritma sudah berhasil untuk di-kriptanalisis. Beberapa contoh algoritma *cipher* blok adalah DES dan GOST. DES termasuk algoritma cipher yang paling populer walaupun kini sudah tidak aman karena berhasil di-kriptanalisis. DES merupakan algoritma kriptografi kunci simetri yang akan membagi pesan menjadi blok-blok dengan ukuran blok sebesar 64 bit. Algoritma DES mengenkripsi pesan dengan menggunakan 56 bit junci *internal* yang dibangkitkan dari 64 bit kunci eksternal.

2. Dasar Teori

2.1. Cipher Blok (Block Cipher)

Pada *cipher* blok, rangkaian bit-bit pesan yang ingin dimanipulasi akan dibagi menjadi blok-blok bit dengan panjang yang sama terlebih dahulu. Setiap blok bit pesan akan dienkripsi atau didekripsi dengan bit-bit kunci yang panjangnya sama dengan panjang blok. Proses enkripsi dengan *cipher* blok akan menghasilkan blok cipherteks yang berukuran sama dengan ukuran plainteks, begitu juga dengan proses dekripsi.

Misalkan terdapat blok plainteks (P) dan cipherteks (C) yang berukuran n bit dinyatakan sebagai $P = (p_1, p_2, p_3, \ldots, p_n)$ dan $C = (c_1, c_2, c_3, \ldots, c_n)$ dengan p_i dan $c_i \in \{0, 1\}$. Apabila plainteks dibagi menjadi m buah blok, maka barisan blok plainteks dapat dinyatakan sebagai $(P_1, P_2, P_3, \ldots, P_m)$. Untuk setiap blok P_i akan disusun dari kumpulan vektor bit yang dinyatakan sebagai $P_i = (p_{i1}, p_{i2}, p_{i3}, \ldots, p_{in})$. Skema enkripsi dengan *cipher* blok dapat dilihat pada gambar berikut.



Gambar 1. Ilustrasi *cipher* blok

2.2. Teknik Kriptografi

Untuk melakukan manipulasi bit pada *cipher* blok, diperlukan beberapa teknik kriptografi klasik untuk enkripsi dan dekripsi. Beberapa teknik tersebut antara lain.

a. Substitusi

Teknik ini akan mengganti sebuah atau sekumpulan bit pada blok plainteks tanpa mengubah urutannya. Secara matematis, teknik ini dapat dinyatakan sebagai berikut.

$c_i = E(p_i),\ i = 1, 2, \ldots$

dengan c_i adalah bit cipherteks, p_i adalah bit plainteks, dan E adalah fungi matematis atau dapat merupakan tabel substitusi.

b. Transposisi

Teknik ini akan memindahkan posisi bit pada blok plainteks dengan aturan tertentu. Secara matematis, teknik ini dapat dinyatakan sebagai berikut.

$C = PM$

dengan C adalah blok cipherteks, P adalah blok plainteks, dan M adalah fungsi transposisi yang diguankan

2.3. *Mode Operasi pada Block Cipher*

Berdasarkan cara blok plainteks atau blok cipherteks dioperasikan dalam *cipher* blok, terdapat beberapa mode operasi pada *cipher* blok seperti *Electronic Code Book, Cipher Block Chaining, Cipher Feedback, Output Feedback,* dan *Counter Mode*.

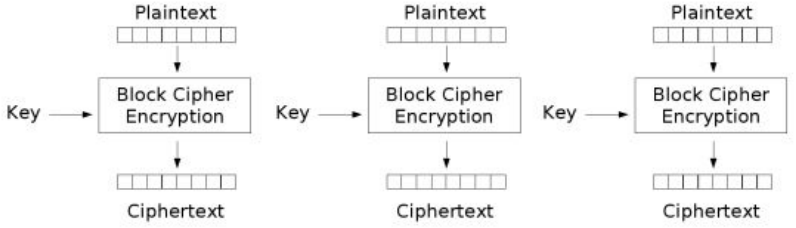
a. *Electronic Code Book* (ECB)

Pada mode ini, setiap blok plainteks P_i dienkripsi secara individual dan independen dari blok lain menjadi blok cipherteks C_i. Secara matematis, proses enkripsi dan dekripsi mode ECB dengan m buah blok dapat dinyatakan sebagai berikut

$C_i = E_k(P_i)$ dan $P_i = D_k(C_i)$

Pada persamaan di atas, K adalah kunci yang digunakan dan P_i dan C_i masing-masing adalah blok plainteks dan cipherteks ke-i.

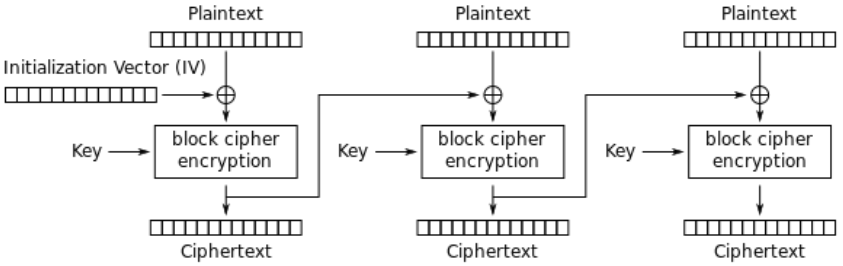
Mode ECB memiliki kelemahan yaitu blok plainteks yang sama akan selalu dienkripsi menjadi blok cipherteks yang sama. Hal ini dikarenakan setiap blok plainteks yang sama setiap blok diproses secara independen. Dengan kelemahan ini, potensi kriptanalis untuk memecahkan cipherteks menjadi mudah dengan melakukan analisis distribusi kemunculan blok-blok.



Gambar 2. Ilustrasi algoritma enkripsi dengan ECB

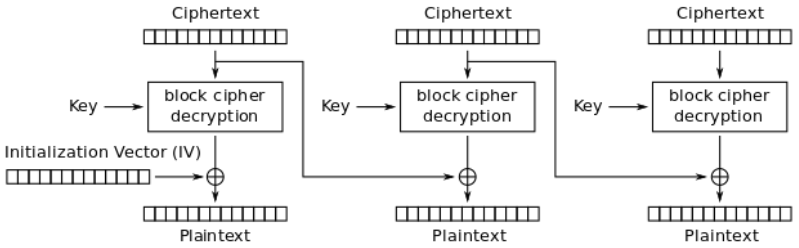
b. *Cipher Block Chaining* (CBC)

Pada mode ini, hasil enkripsi blok sebelumnya akan di-umpan-balikkan (dijadikan parameter) ke proses enkripsi blok yang sekarang. Hal ini bertujuan untuk memberikan ketergantungan dengan merantaikan setiap blok. Mekanisme yang digunakan pada metode ini adalah blok plainteks yang sekarang di-XOR-kan terlebih dahulu dengan blok cipherteks sebelumnya. Selanjutnya hasil operasi XOR akan masuk dalam fungsi E. Dengan mode ini, setiap blok cipherteks tidak hanya bergantung pada blok plainteksnya saja, melainkan bergantung pada seluruh blok plainteks sebelumnya.



Gambar 3. Ilustrasi algoritma enkripsi dengan CBC

Proses dekripsi dengan metode ini dilakukan dengan memasukkan blok cipherteks sekarang ke fungsi dekripsi. Kemudian, hasil dari fungsi dekripsi akan dioperasikan secara XOR dengan blok cipher sebelumnya. Pada proses dekripsi, blok cipherteks sebelum nya akan berfungsi sebagai umpan-maju pada akhir proses dekripsi.

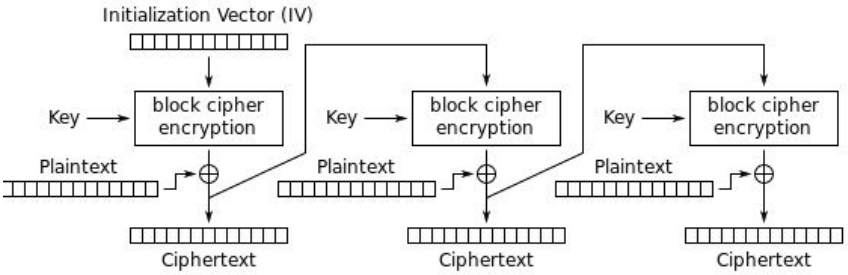


Gambar 4. Ilustrasi algoritma dekripsi dengan CBC

Karena blok cipherteks yang dihasilkan bergantung dengan blok-blok plainteks sebelumnya, maka kesalahan satu bit pada sebuah blok plainteks akan berakibat fatal karena akan merambat pada semua blok cipherteks setelahnya.

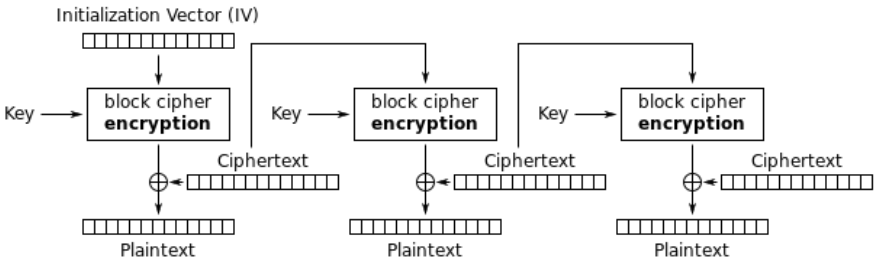
c. Cipher Feedback (CFB)

Metode CFB adalah alternatif dari metode CBC yang digunakan untuk mengatasi masalah ketika plainteks yang diterima belum lengkap membentuk satu blok. Pada metode ini, data akan dienkripsi dalam unit yang lebih kecil daripada ukuran blok. Secara umum, CFB p-bit mengenkripsi plainteks sebanyak p bit setiap kalinya dengan $p \leq n$ (n = ukuran blok). Dengan kata lain, CFB mengenkripsi *cipher* blok seperti enkripsi pada *cipher* alir.



Gambar 5. Ilustrasi algoritma enkripsi dengan CFB

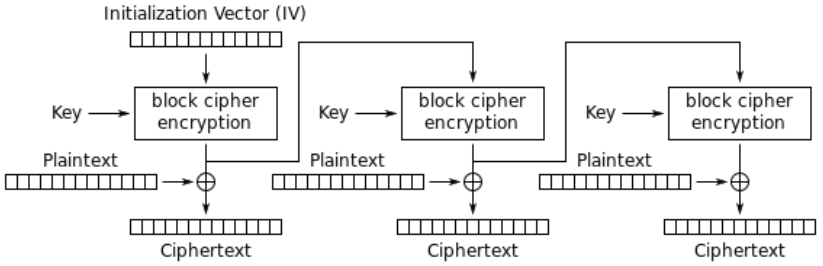
Sama seperti metode CBC, kesalahan 1 bit pada blok plainteks akan merambat pada blok-blok cipherteks yang berkoresponden dan blok cipherteks berikutnya pada proses enkripsi. Namun, pada proses dekripsi kesalahan 1 bit pada blok cipherteks hanya akan mempengaruhi blok plainteks yang saat itu dan satu blok plainteks setelahnya.



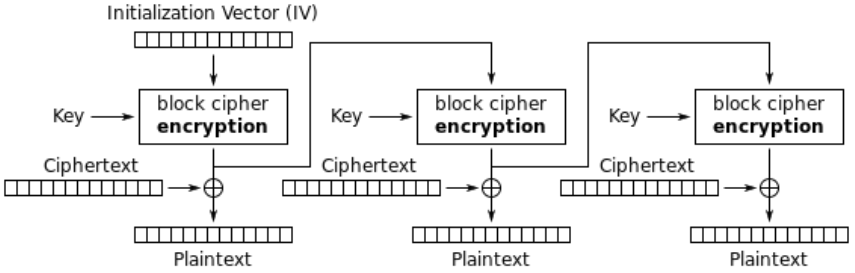
Gambar 6. Ilustrasi algoritma dekripsi dengan CFB

d. Output Feedback (OFB)

Mode OFB mirip dengan mode CFC, hanya saja p-bit hasil enkripsi akan disalin menjadi elemen dengan posisi paling kanan dalam antrian. Dengan begitu, kesalahan 1bit pada blok plainteks hanya mempengaruhi blok cipherteks yang berkoresponden saja. Begitu pula pada proses dekripsi, kesalahan 1bit pada blok cipherteks hanya mempengaruhi blok plainteks yang bersangkutan saja.



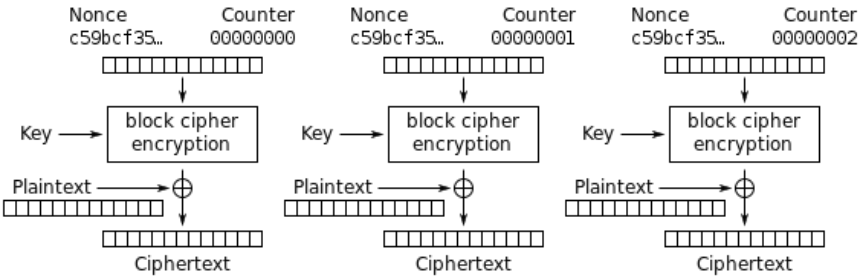
Gambar 7. Ilustrasi algoritma enkripsi dengan OFB



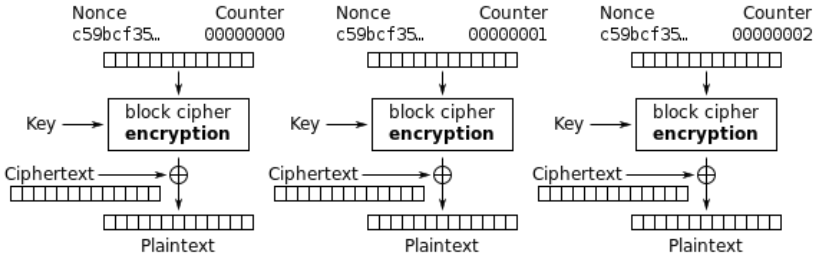
Gambar 8. Ilustrasi algoritma dekripsi dengan OFB

e. Counter Mode

Mode *counter* cukup berbeda dengan mode sebelumnya. Mode ini tidak melakukan proses perantaian seperti pada CBC dan sekilas akan mirip ECB karena setiap blok diproses secara independen. Perbedaan mode ini dengan ECB adalah terdapat penambahan blok baru yang dinamakan *counter*. Pada mulanya, counter diinisialisasi dengan sebuah nilai yang selanjutnya untuk enkripsi blok berikutnya, nilai dari counter akan dinaikkan satu.



Gambar 9. Ilustrasi algoritma enkripsi dengan *counter mode*



Gambar 10. Ilustrasi algoritma dekripsi dengan *counter mode*

2.4. Prinsip-prinsip Perancangan Cipher Blok

Dalam merancang sebuah *cipher* blok baru, terdapat beberapa prinsip penting yang perlu diperhatikan. menurut Schneier (1996) perancangan algoritma kriptografi *cipher* blok harus mempertimbangkan beberapa prinsip sebagai berikut.

a. Prinsip *Confusion* dan *Diffusion*

Prinsip *confusion* bertujuan untuk menyembunyikan hubungan apapun yang terdapat antara cipherteks, plainteks, dan kunci. Prinsip ini akan menyulitkan kriptanalis untuk mencari pola-pola statistik dari hubungan antara plainteks, cipherteks, dan kunci. Cara paling sederhana untuk menciptakan efek *confusion* adalah dengan melakukan proses substitusi pada blok bit.

Prinsip *diffusion* akan menyebarkan pengaruh satu bit plainteks atau kunci ke sebanyak mungkin cipherteks. Pada prinsipnya, perubahan 1 bit plainteks akan menciptakan perubahan yang berarti pada cipherteks. Prinsip ini tentu saja akan menyulitkan kriptanalis untuk mencari hubungan antara cipherteks, plainteks, dan kunci, Cara paling mudah untuk menciptakan efek *diffusion* adalah dengan melakukan proses transposisi pada blok bit.

b. Iterated Cipher

Secara mudahnya, proses ini akan mengulang proses modifikasi bit pada plainteks sebanyak beberapa kali. Proses ini bertujuan untuk mendapatkan efek *confusion* dan *diffusion* yang sangat kuat. Secara matematis, proses ini dapat dinyatakan dengan

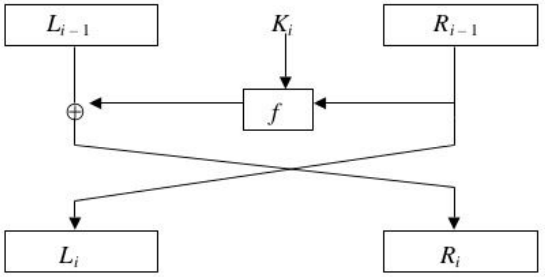
$$C_i = f(C_{i-1}, K_i)$$

dengan i adalah jumlah putaran, K_i adalah upa-kunci pada putaran ke-i, dan f adalah fungsi modifikasi bit yang didalamnya mungkin terdapat fungsi substitusi, permutasi, ekspansi, maupun kompresi.

c. Jaringan Feistel

Jaringan Feistel adalah sebuah model manipulasi blok bit yang membagi blok menjadi dua buah bagian. Model ini ditemukan oleh Horst Feistel pada tahun 1970. Model jaringan ini dapat dijelaskan sebagai berikut.

1. Sebuah blok bit akan dibagi menjadi dua bagian yaitu kiri (L) dan kanan ® yang sama panjang.
2. Definisikan *cipher* blok berulang yang mana hasil putarahn ke-i akan ditentukan oleh hasil putaran sebelumnya dengan aturan:
   $L = R_{i-1}$
   $R_i = L_{i-`1} \oplus fi(R_{i-1}, K_i)$



Gambar 11. Ilustrasi jaringan feistel

d. Kotak-S

Kotak-S adalah matriks yang berisi substitusi sederhana yang memetakan satu atau lebih bit dengan satu atau lebih bit yang lain. Perancangan kotak-S menjadi sebuah isu yang penting karena dapat menentukan keluaran dari algoritma kriptografi yang bagus dan mudah diimplementasikan.

Misalkan terdapat kotak-S sebagai berikut yang akan men substitusi 6 bit b_1, b_2, b_3, b_4, b_5, b_6

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

Nomor baris dari tabel akan ditunjukkan oleh string bit b_1b_6 sedangkan nomor kolom akan ditunjukkan oleh string bit $b_2b_3b_4b_5$

Sehingga jika terdapat masukan 110100 (52), maka nomor baris tabel adalah 10 (2) dan nomor kolom tabel adalah 1010 (10). Sehingga substitusi bit 110100 akan ditunjukkan pada elemen tabel;l pada baris ke-2 dan kolom ke-10 yaitu 4.

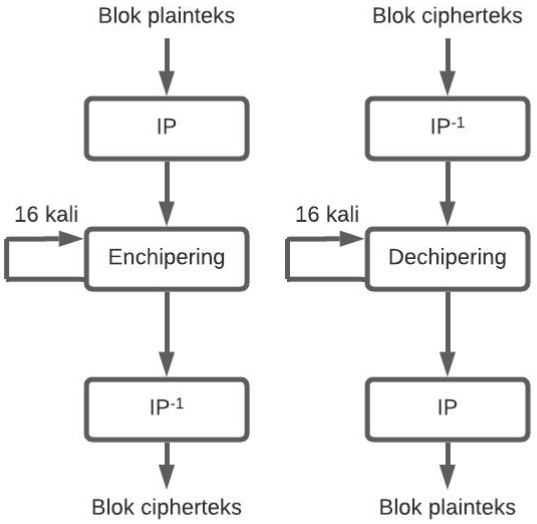
3. Proposed Block Cipher

Cipher blok yang kami rancang beroperasi dengan ukuran blok 64-bit. Pada algoritma ini, pesan akan dibagi menjadi blok-blok berukuran 64-bit dengan panjang kunci *internal* 256-bit. Kunci *internal* akan dibangkitkan dari kunci *eksternal* yang diberikan oleh pengguna. Fungsi *internal* akan dibangkinkan dengan menggunakan fungsi sha256 dan menerima parameter fungsi eksternal ditambah dengan *secret message* yang menjadi *string* konstan dalam program.

Sebuah plainteks akan dibagi menjadi beberapa blok 64-bit, dengan setiap blok dilakukan proses *ciphering*. Proses ini dilakukan sebanyak 16 kali untuk setiap blok, dan setiap proses *enciphering* didefinisikan dengan arsitektur jaringan Feistel yang kami usulkan. Arsitektur ini kami beri nama YDIHYLFSIPFIF (baca*: yedih-yelf-sip-fif*), sebuah singkatan dari *Yo Dawg I Heard You Like Feistel, So I Put Feistel Inside Feistel*, yang namanya cukup merepresentasikan struktur arsitektur model *cipher* usulan kami dimana komponen atau noda *right* (kanan) dari jaringan Feistel biasa dimodifikasi agar memiliki jaringan Feistel sendiri.

Skema global dalam algoritma YDIHYLFSIPFIF adalah sebagai berikut.

- Blok plainteks dipermutasi dengan matriks permutasi awal (*initial permutation*)
- Hasil permutasi awal kemudian akan dimanipulasi dengan proses *enciphering* sebanyak 16 putaran. Setiap putaran menggunakan kunci internal yang berbeda.
- Hasil *enciphering* kemudian dipermutasi dengan matriks permutasi balikan (*inverse initial permutation*).



Gambar 12. Skema global algoritma enkripsi (kiri) dan dekripsi (kanan) YDIHYLFSIPFIF

3.1. Permutasi Awal

Permutasi awal dilakukan dengan menggunakan *list* berisi posisi setiap *byte* pada plainteks maupun cipherteks, yang direpresentasikan dengan angka 0..63 dikarenakan setiap blok memiliki 64-bit. *List* tersebut dilakukan *shuffling* dengan menggunakan *seed* yang berasal dari *key* sesuai masukan pengguna. Urutan *bit* pada masukan plainteks maupun cipherteks akan di-*shuffle* berdasarkan *list* yang telah diacak tersebut menjadi urutan sembarang yang tidak bermakna.

3.2. Enciphering

Dalam proses *enciphering*, setiap blok plainteks akan dibagi menjadi dua bagian yaitu sub-blok kiri (L) dan sub-blok kanan (R) dengan panjang yang sama. Sub-blok kanan akan dibagi lagi menjadi dua bagian yang akan dinamai dengan LR dan RR. Bagian-bagian ini akan masuk ke dalam 16 putaran proses manipulasi bit. Pada setiap manipulasi, sub-blok R dan sub-blok RR merupakan masukan untuk fungsi transformasi dengan nama *scramble* yang akan mengkombinasikan sub-blok masukan dengan kunci internal K_i. Luaran dari fungsi ini, akan dioperasikan secara XOR dengan sub-blok L atau sub-blok LR untuk menghasilkan sub-blok baru R dan RL. Sedangkan sub-blok baru LL akan langsung didapatkan dari sub-blok RR sebelumnya.

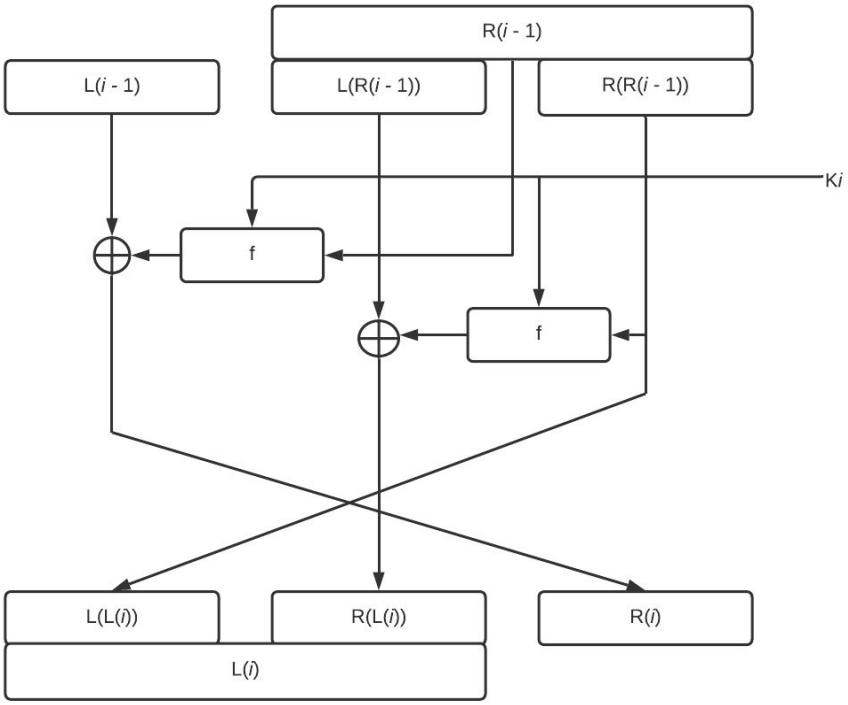
Dapat diperhatikan bahwa algoritma ini merupakan modifikasi dari jaringan feistel biasa. Pada jaringan feistel biasa, blok L yang baru akan langsung didapatkan dari sub-blok R yang lama. Sedangkan pada algoritma ini, sub blok L yang baru dan sub blok R yang lama, yang telah dibagi menjadi dua bagian, akan diproses pada sebuah jaringan feistel lagi. Secara matematis, satu putaran YDIHYLFSIPFIF dapat dinyatakan sebagai berikut.

$$LL_i = RR_{i-1}$$

$$RL_i = LR_{i-1} \oplus f_i(RR_{i-1}, K_i)$$

$$L_i = LL_i \Vert RL_i$$

$$R_i = L_{i-1} \oplus f_i(R_{i-1}, K_i)$$



Gambar 13. Arsitektur dari modifikasi jaringan feistel pada proses *cipher* setiap blok algoritma YDIHYLFSIPFIF

Pada metode ini, untuk menyembunyikan hubungan apapun yang terdapat antara cipherteks, plainteks, dan kunci, digunakan substitusi bit pada blok dengan menggunakan kotak-S. Setiap blok yang panjangnya 32-bit, akan dibagi menjadi 8 buah bagian yang masing-masing panjangnya 4-bit. Algoritma ini memiliki 8 buah kotak S yang setiap kotaknya berisi 16 buah nilai. Setiap 4-bit ini akan masuk ke kotak-S untuk proses substitusi. Setiap bit pada potongan blok ke-i akan diproses dengan kotak S ke-i.

Setiap elemen pada kotak-S diindex dari 0 sampai 15. masukan untuk kotak S akan berasosiasi dengan index elemen sedangkan keluarannya akan berasosiasi dengan nilai di dalam elemen dengan indeks tersebut. Misal terdapat 4 bit masukan 0001 pada suatu blok kotak dengan susunan sebagai berikut. Maka luarannya adalah nilai dengan elemen ke-1 yaitu 3 atau dalam biner 0011.

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

Setelah proses substitusi, selanjutnya akan dilakukan proses transposisi berupa pergeseran pada bit hasil substitusi. Hal ini bertujuan untuk menyebarkan pengaruh satu bit plainteks sehingga hasil *enciphering* akan semakin teracak. Proses transposisi yang dilakukan adalah dengan menggeser blok bit hasil substitusi sebesar 4 bit ke kiri.

Adapun fungsi *scramble* atau fungsi F_i yang digunakan untuk "mengacak" nilai sebelum dimasukkan ke operasi XOR dapat berupa fungsi satu arah dan tidak reversibel (tidak seperti AES). Hal ini disebabkan karena algoritma YDIHYLFSIPFIF yang digunakan mengadopsi landasan berupa struktur Feistel di mana F_i hanya dipakai sebelum XOR, dan nilai dari (F XOR x) XOR F pasti bernilai x untuk F sembarang. Dengan kata lain, fungsi F_i ini pada esensinya bertujuan untuk membangkitkan nilai sembarang yang baik pada hasil cipherteks. Untuk meningkatkan *confusion*, kami mengusulkan fungsi pengacak sembarang sederhana menggunakan kombinasi perkalian dan perpangkatan berupa

$$f_i(x,\ k)\ =\ (x\ \times\ k)^i$$

Seluruh proses ini akan dilakukan sebanyak 16 kali melalui istilah *round*, yang didefinisikan sebagai nilai i pada ilustrasi arsitektur dan persamaan di atas. Nilai *key* yang digunakan pada *round* ke-i, dinyatakan dengan K_i, diperoleh dengan pembangkit kunci setiap awal mulai dari suatu *round* atau putaran. Pembangkit ini cukup sederhana, dibangkitkan menggunakan *subkeygen* yang membutuhkan parameter i dan *initial key* untuk kemudian dimasukkan ke dalam fungsi bawaan `hashlib.sha256` sebagai konkatenasi antara i dengan *initial key*, menghasilkan *key* baru berbentuk SHA256 untuk kemudian digunakan sebagai K pada *round* yang bersangkutan (ke-i).

3.3. Deciphering

Dalam proses *deciphering*, prosesnya pada dasarnya adalah mengembalikan tanda anak panah pada proses *enciphering*. Setiap blok cipherteks akan dibagi menjadi dua bagian yaitu sub-blok kiri (L) dan sub-blok kanan (R) dengan panjang yang sama. Sub-blok kiri akan dibagi lagi menjadi dua bagian yang akan dinamai dengan LL dan RL. Bagian-bagian ini akan masuk ke dalam 16 putaran proses manipulasi bit.

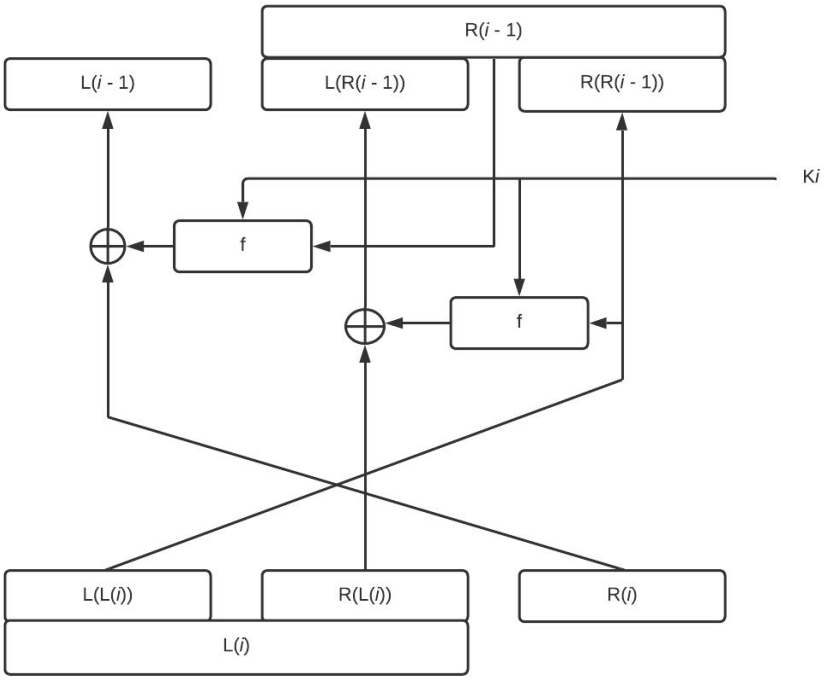
Pada setiap manipulasi, sub-blok LL merupakan masukan untuk fungsi transformasi dengan nama *scramble* yang akan mengkombinasikan sub-blok masukan dengan kunci internal K_i. Luaran dari fungsi ini, akan dioperasikan secara XOR dengan sub-blok RL untuk menghasilkan sub-blok baru LR. Sedangkan sub-blok baru RR akan langsung didapatkan dari sub-blok LL sebelumnya. Nilai LR dan RR ini akan digabung menjadi sub-blok baru R, yang akan dilakukan fungsi *scramble* dengan K_i yang sama lalu di-XOR dengan subblok R menjadi nilai L yang baru. Secara matematis, satu putaran YDIHYLFSIPFIF dapat dinyatakan sebagai berikut.

$$LR_{i-1} = RL_i \oplus f_i(LL_i, K_i)$$

$$RR_{i-1} = LL_i$$

$$R_{i-1} = LR_{i-1} \Vert RR_{i-1}$$

$$L_{i-1} = R_i \oplus f_i(R_{i-1}, K_i)$$



Gambar 14. Arsitektur dari modifikasi jaringan feistel pada proses *decipher* setiap blok algoritma YDIHYLFSIPFIF

3.4. Permutasi Akhir

Permutasi akhir dilakukan dengan menggunakan *list* berisi posisi setiap *byte* pada plainteks maupun cipherteks setelah dilakukan *shuffle* menggunakan *seed* dari *key* masukan pengguna, lalu dilakukan *unshuffle* dengan mekanisme pencarian posisi semula dari *list* yang telah di-*shuffle*, yaitu melalui pembalikkan nilai indeks dan nilai isi *list*.

4. Eksperimen dan Analisis Hasil

Untuk metode eksperimen, kami bertujuan untuk menganalisis kasus antara enam hal yaitu

- Uji coba *encrypt* dan *decrypt* plainteks dengan kata sembarang. Potongan data plainteks yang kami gunakan berasal dari **Alice's Adventures in Wonderland** (Lewis Carroll, 1865). Berikut adalah potongan data plainteks yang dimaksud.

```
CHAPTER I. Down the Rabbit-Hole

Alice was beginning to get very tired of sitting by her sister on the
bank, and of having nothing to do: once or twice she had peeped into the
book her sister was reading, but it had no pictures or conversations in
it, 'and what is the use of a book,' thought Alice 'without pictures or
conversations?'
```

- Bentuk cipherteks dari plainteks dengan kata yang berulang (prinsip *confusion*). Potongan data untuk eksperimen poin ini adalah sebagai berikut.

```
abcdefghabcdefghabcdefghabcdefghabcdefghabcdefghabcdefghabcdefghabcdefghabc
```

```
defghabcdefghabcdefghabcdefghabcdefghabcdefghabcdefghabcdefghabcdefghabcdef
ghabcdefghabcdefghabcdefghabcdefghabcdefghabcdefghabcdefghabcdefghabcdefgha
bcdefghabcdefghabcdefghabcdefghabcdefghabcdefghabcdefghabcdefghabcdefghabc
```

- Bentuk plainteks yang didekripsi dari cipherteks yang rusak sebagian (prinsip *diffusion*). Potongan data untuk eksperimen poin ini sama seperti potongan data pada poin (1).
- Bentuk plainteks yang didekripsi dari cipherteks tertentu dengan satu *bit* kunci yang diganti. Potongan data eksperimen ini sama seperti poin (1).
- Perbedaan hasil cipherteks apabila plainteks diubah dengan satu *bit*. Potongan data yang digunakan sembarang, pendek.
- Statistika tingkat kemunculan setiap *byte* pada plainteks, dibandingkan dengan statistika tingkat kemunculan setiap *byte* pada cipherteks.

Keenam poin yang disebutkan di atas akan diuji-coba menggunakan arsitektur yang telah kami buat dengan tiga mode berbeda yaitu *electronic codebook* (ECB), *cipher block chaining* (CBC), dan mode *counter*. *Key* yang kami pakai menggunakan perangkat lunak yang sudah dibangun untuk melakukan uji coba adalah "`password`".

4.1. Uji coba encrypt-decrypt

Pada bagian ini, hasil uji coba akan dituliskan dalam bentuk tabel 1x2 di mana baris pertama adalah hasil enkripsi plainteks yang didefinisikan sebelumnya, sedangkan baris kedua adalah hasil dekripsi dari hasil enkripsi plainteks yang ada pada baris pertama. Setelah tabel, dituliskan analisis mengenai hasil enkripsi dan dekripsi dari mode yang disebutkan.

Mode *Electronic Codebook*

`m^N^K7````^X^UÊ`o``P_I`<^B`w`^BR/``GZ3`^@0`dW```z```````^KX`````J ``^[`^A3^?^C<^E]w2a``o`e`c``$`N*``&`tH``v_`MɑW>`^W^?`kN0a`Fw`^B'IH`p Y^?````^U`K_Uf`Btg@``?0`^\V`Q````t`^Zƀs^^v````2W`/(x_@`□^H``^Y6D{`p ` `I```fi `v`^KKK |*깢[d:`п``^O`Qy``K`V````^Z``c`C`^W_Ia`z`^D`x``^X>``````6````ꓖBSz ```=`(`hd`x``hk4Q`^G>``^SH1`^V}Y`#x `U``ↇ N`:~`]k`\^[```~`K`}9`=5^U``^P`````^_`^GmO^D`+1`
`CHAPTER I. Down the Rabbit-Hole Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, 'and what is the use of a book,' thought Alice 'without pictures or conversations?'`

Pada mode ECB menggunakan arsitektur YDIHYLFSIPFIF yang sudah dirancang, terlihat bahwa perangkat lunak sudah berhasil melakukan enkripsi dan dekripsi, dengan aplikasi prinsip *confusion* terlihat pada kemunculan *byte* unik yang lebih banyak dibandingkan plainteks. Hasil dekripsi memiliki tiga tambahan *byte* berbentuk spasi di akhir teks dikarenakan jumlah *byte* pada plainteks semula memiliki panjang teks mod 8 sebesar -3 (141).

Mode *Cipher Block Chaining*

```
m^N^K7)o4H^I^D.'/^D^T"TE0f^K3Pj|S>iW7~
X^T  c4^K<^F^K^K^N^bqm^E^J^pk`dr/^EE$^X^^[L^^E`t
O}6^X^
cT^_F ^Ey
M;|^n>^Hi$?^Bg#1rU^D^h^'p^^^^T3$d^4R[^Qa^^N^,W^?^l4^m^^^V
^\}^R^Hú^{~^CJN^a^X^c^
^bD^m^^B^^^8^F^^^U?^3^E-U^^^dQ$^QL^^^Z^.^^V^^4^@^F^G^
U^^T^#cM^.^^~)S^_x^Pr)^~^T^^#^^^C^QyJN^v^Q^^^D^^^H^^S/p^
```

```
CHAPTER I. Down the Rabbit-Hole

Alice was beginning to get very tired of sitting by her sister on the
bank, and of having nothing to do: once or twice she had peeped into the
book her sister was reading, but it had no pictures or conversations in
it, 'and what is the use of a book,' thought Alice 'without pictures or
conversations?'
```

Pada mode CBC menggunakan arsitektur YDIHYLFSIPFIF yang sudah dirancang, terlihat bahwa perangkat lunak sudah berhasil melakukan enkripsi dan dekripsi, dengan aplikasi prinsip *confusion* terlihat pada kemunculan *byte* unik yang lebih banyak dibandingkan plainteks. Dapat terlihat juga pada hasil enkripsi CBC, 8 *byte* (64 bit) pertama pada cipherteks yaitu `m^N^K7` sama seperti 8 *byte* pertama pada enkripsi menggunakan mode ECB dikarenakan *initial values* hanya berubah dan berpengaruh pada indeks blok kedua dan seterusnya (dipengaruhi oleh blok-blok sebelumnya).

Mode *Counter*

```
m^N^K7^^^Q^H^oa
^^^^^^^^^B^R^!l^Q^shx^^$Zt^m^^Rp^[\>^^^O^^^^U^X.^!^^v^Z0^^,<u
^Q^S^_^Z!^H^^a^[^^U^^y^.^NQ^.U^^X^F
^D^60^^^\^Q^^<^^^^F(^/^OT^^V^^F^^Ef6^^^@5n^^P^U^(^^H!^^R^^@^
^^^^)^R^^^^n}qF^^J^A^%(^^^EQR^g^^^^^^^^^J^W^^^^O^P+
O^^^CE^b^V^(4^R^^^^w^Y^N^Y^24^W>^E^^{<^"^'H^0^^^<^{^^^^jJ^<^G^^^
Y^n^^i^^^T^?^^^s^^^!|A#^R^Tz\8^^;^Y^ ^o^JIvo^^^^T^^N^G^Y
^D^^P^^Qm
```

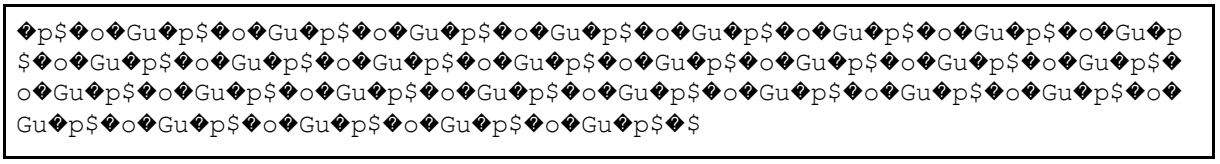
```
CHAPTER I. Down the Rabbit-Hole

Alice was beginning to get very tired of sitting by her sister on the
bank, and of having nothing to do: once or twice she had peeped into the
book her sister was reading, but it had no pictures or conversations in
it, 'and what is the use of a book,' thought Alice 'without pictures or
conversations?'
```

Pada mode *counter* menggunakan arsitektur YDIHYLFSIPFIF yang sudah dirancang, terlihat bahwa perangkat lunak sudah berhasil melakukan enkripsi dan dekripsi, dengan aplikasi prinsip *confusion* terlihat pada kemunculan *byte* unik yang lebih banyak dibandingkan plainteks. Dapat terlihat juga pada hasil enkripsi *counter*, 8 byte (64 bit) pertama pada cipherteks yaitu `m^N^K7` sama seperti 8 *byte* pertama pada enkripsi menggunakan mode ECB dan CBC dikarenakan *counter* untuk indeks pertama adalah `00000000` (tidak mempengaruhi nilai *xor*).

4.2. Plainteks dengan kata berulang

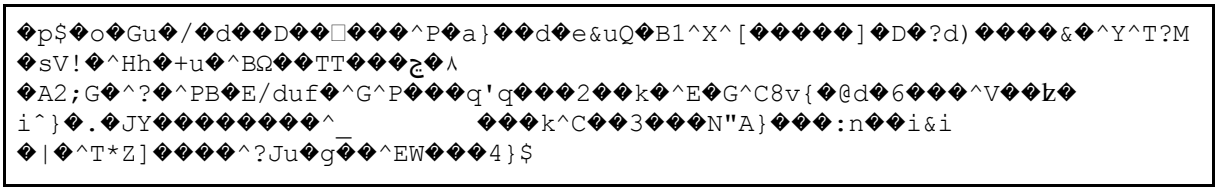
Pada bagian ini, hasil uji coba akan dituliskan dalam bentuk tabel 1x1 (kotak) di mana kotak itu menampilkan hasil enkripsi dari *cipherteks* menggunakan mode tertentu. Setelah tabel, dituliskan analisis mengenai hasil enkripsi (terutama pada prinsip *confusion*) dari mode yang disebutkan.

Mode *Electronic Codebook*



Pada mode ECB dengan plainteks kata berulang, terlihat bahwa cipherteks yang dihasilkan memiliki nilai *confusion* yang rendah dikarenakan masih terlihat secara kasat mata bahwa ada perulangan. Hal ini masuk akal karena mode ECB tidak memiliki keterkaitan antar blok, sehingga setiap blok pada plainteks akan dilakukan *cipher* dengan *initial key* untuk enkripsi & *shuffling* yaitu *key* yang diberikan di awal (tidak mengalami perubahan ketika berpindah blok). Dapat terlihat bahwa hasil cipherteks akan berulang setiap 8 *byte*, sama seperti bentuk plainteks semula.

Mode *Cipher Block Chaining*



Pada mode CBC, plainteks dengan kata berulang tidak menghasilkan cipherteks dengan kata berulang. Hal ini membuat statistika tidak dapat memecahkan cipherteks, sehingga nilai *confusion* akan lebih tinggi dibandingkan ECB. Penyebab dari fenomena ini berhubungan dengan sifat alamiah dari CBC sendiri, di mana setiap vektor yang menjadi *output* dari proses *cipher* akan dilakukan *xor* dengan *input* untuk blok selanjutnya.

Mode *Counter*

Pada mode *counter*, plainteks dengan kata berulang tidak menghasilkan cipherteks dengan kata berulang. Sama seperti metode CBC, hal ini membuat statistika tidak dapat memecahkan cipherteks, sehingga nilai *confusion* akan lebih tinggi dibandingkan ECB. Penyebab dari fenomena ini adalah bahwa setiap blok pada mode *counter* akan dilakukan *xor* dengan *counter* yang nilainya terus bertambah setiap pergantian blok, sehingga menyebabkan blok berbeda dengan konten yang sama akan dienkripsi menjadi cipherteks yang berbeda.

4.3. Cipherteks dengan sebagian kerusakan

Pada bagian ini, hasil uji coba akan dituliskan dalam bentuk tabel 1x2 di mana baris pertama adalah hasil enkripsi plainteks yang dilakukan perusakan, ditandai dengan *highlight* berwarna kuning, sedangkan baris kedua adalah hasil dekripsi dari cipherteks "rusak" seperti pada baris pertama. Setelah tabel, dituliskan analisis mengenai hasil enkripsi dan dekripsi dari mode yang disebutkan.

Mode *Electronic Codebook*

`m^N^K7````^X^UÊ`o``P_I`<^B`w`^BR/``GZ3`^@0`dW```z```````^KX``````J ``^[`^A3^?^C<^E]w2a``o`e`c``$`N*``&`tH``v_`MɑW>`^W^?`k`N0a`Fw`^B'IH` pY^?````^U`K_Uf`Btg@``?0`^\V`Q````t`^Zƀs^^v````2W`/(x_@`□^H``^Y6D{` p` `I```fi `v`^KKK |*깢[d:`п``^O`Qy``K`V````^Z``c`C`^W_Ia`z`^D`x``^X>``````6````ЧBSz ```=`(`hd`x``hk5Q`^G>``^SH1`^V}Y`#x`U``ↄ N`:~`]k`\^[```~`K`}9`=5^U``^P````^_`^GmO^D`+1
CHAPTER I. Down the Rabbit-Hole Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, 'and what is thk`L`ڱ`` a book,' thought Alice 'without pictures or conversations?'  `W``w``r

Pada mode ECB menggunakan arsitektur YDIHYLFSIPFIF yang sudah dirancang, terlihat bahwa untuk cipherteks dengan sedikit kerusakan (1 bit dari 00000100 menjadi 00000101 pada label kuning) hanya akan merusak blok yang terkait saja. Hal ini masuk akal dikarenakan mode ECB tidak memiliki keterkaitan antar blok, sehingga blok yang rusak tidak akan "menjalar" ke blok lain. Teks setelahnya masih dapat dibaca, namun perhatikan juga, terlihat pada bagian terakhir terdapat 8 *byte* tambahan yang tidak diinginkan. 8 *byte* tambahan itu muncul akibat kesalahan sistem dalam mencari *length* dari plainteks, dikarenakan sistem mengharapkan *unicode* yang sesuai pada hasil dekripsi.

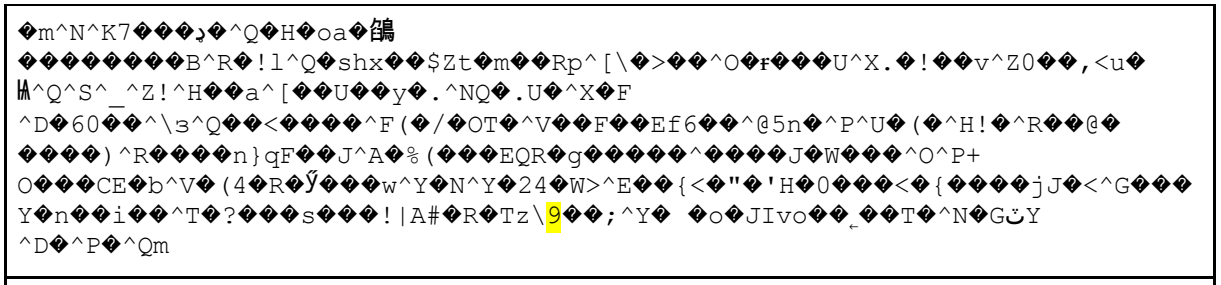
Mode *Cipher Block Chaining*

`m^N^K7````)``o`4ʂH`I^D```˜``.`'`/^D`T"T`È`0````f^K3`Pj`|S>`iW`7~` ɻ□X`^T``  c``4^K<`^F````^K`N``bqm`E``J`pk`` `dr/`EE$^X`^[L`^E` ```t` O}6`^X``厂`^]-` cT^_F ^Ey`` `MŰ``;\|^n>^Hi$?`^Bg#1rU^D```h``'p``````T3$d`5R[^Qa``N`,W^?`l4`m```^V `\}^R^Hú`{~^CJNڿќa`X`c`` `bD̥m``^B```ȣ^F``^U?`3^E-U```dQ$^QL``^Z`.``V``4^@`F^G`ڏ U``T`#cM`.``~)S`_x^Pr)`~^T``#```^C^QyJN`v`Q`^^D``ɤ`H`^S/p`
CHAPTER I. Down the Rabbit-Hole Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book`<``7`%`0X````t'@PY$``ȮE`V`b0``□`````````&_`n`iR`J`f``N```n` `nW`"````6`J*``W`` us`:``"$``n?J```qd`Y`ImW`m +``m`1`N`5```\~`x``'````)We``````z`0`

Pada mode CBC, terlihat bahwa kerusakan pada satu bit saja (00000100 menjadi 00000101) pada cipherteks mengubah keseluruhan isi pada hasil dekripsi untuk blok sendiri dan blok-blok setelahnya. Hal ini disebabkan vektor yang menjadi *xor* untuk blok selanjutnya dipengaruhi oleh hasil cipherteks

blok sebelumnya, sehingga kesalahan pada cipherteks suatu blok akan merusak seluruh blok yang ada di belakangnya. Hal ini berarti bahwa *diffusion* pada mode CBC lebih baik dibandingkan dengan menggunakan mode *ECB* (dan *counter*, yang akan dijelaskan selanjutnya).

Mode *Counter*

```

```

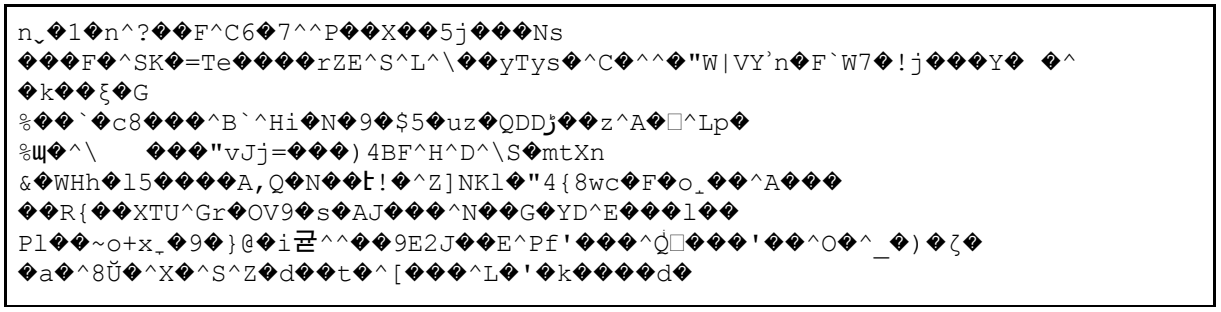
```
CHAPTER I. Down the Rabbit-Hole

Alice was beginning to get very tired of sitting by her sister on the
bank, and of having nothing to do: once or twice she had peeped into the
book her sister was reading, but it had no pictures or conversations in
it, 'and what is the use of a book,' thought Alice
q
 pictures or
conversations?'
```

Pada mode *counter*, terlihat bahwa kerusakan pada satu *bit* (00001000 menjadi 00001001) pada cipherteks hanya mengubah blok yang berkaitan saja. Hal ini berarti bahwa *diffusion* dari mode *counter* cenderung lebih lemah dibandingkan pada mode CBC. Pahami bahwa pada "or" dan "conversations" di plainteks memang ada *newline*, sehingga tetap muncul pada hasil dekripsi. Selain itu, seperti pada kasus ECB, muncul bit-bit yang tidak bermakna di akhir hasil dekripsi dikarenakan sistem mengharapkan *unicode* yang sesuai sehingga dapat mengalami kesalahan dalam menghitung panjang plainteks.

4.4. Kunci dengan sebagian kerusakan

Pada bagian ini, cipherteks yang digunakan adalah cipherteks hasil dari eksperimen (4.1). Hasil diberikan dalam tabel 1x1 (kotak), dengan setiap kotak menandakan hasil dekripsi dari cipherteks dengan mode sesuai, namun memiliki *key* yang berbeda satu bit: "pbssword" (*key* awal: "password").

Mode *Electronic Codebook*

```

```

Pada mode ECB dengan *key* yang diubah satu *bit*, hasil dekripsinya menjadi tidak sesuai sama sekali (bahkan pada bit-bit pertama). Hal ini dikarenakan pembangkitan *key* dilakukan menggunakan *hash* MD5 yang bersumber dari *key* sebelumnya (ditambahkan *i* yaitu indeks *round*), sehingga kesalahan pada *key* akan menyebabkan tidak terdefinisinya pesan yang diinginkan.

Mode *Cipher Block Chaining*

```
n◡�1�n^?#�^^(^W/#,:�b-^\9��^?��-����^H�^[Ek^_�o���)^K{^AUS��H�vr��}.
����^F�
��^Oi̇��(^$�^F30n��*`*�^X�e�15L�����^D'���^S��`��L.�Ny�V��G�VySBM�
�����C~u�^Y�1��^@+1�^X��
�nI�;9ǫZ�!'�M]`�_e���EM�[A*�^[!^@���w/N�^Z���F<��^��9�^A&s
�e»^?��^B�^T���(7Y�Y¥^RK^V��p��3�R#�_7��
�^[,�^@Yj%fX̧��Ġh{DT:^Y��g,���`��=����^N�^Dt^@{^U���¬���^Ao^H����
�^?r>9^A^B��□��O���^R��&o���^G9���^_^X�M��I^F
```

Pada mode CBC, sama seperti mode ECB, *key* yang mengalami perubahan sebanyak 1 *bit* memiliki pengaruh yang sangat besar dalam dekripsi (menjadi tidak bermakna).

Mode *Counter*

```
n◡�1�n^?i\[d'�^?}��k�f�b�  ^]�6�^G�m�E�/S��w���^H�G�^L�1^
^W�^Zo��^X�������=�^E���dX,��g�:��VX�hLRHd�ъ�!�W�^U□'ɹ�x$�m�j�
�������^K.L^Hl���R����l^Ke�J��$`□
=^W,?>p�^U?#��>^A*��]`�&>A?m^U��L�^Z[r^���.�}<w
��
S[0Zj o K[8^U
2uf~MB^B¼)^L7U,J?/!^L A-2y4 8W^_6&□|A ỸI ^
Xj1aep= _^K J^_a ^S
:)"z8u     9`4(      q6y
```

Pada mode *counter*, sama seperti mode ECB dan CBC, *key* yang mengalami perubahan sebanyak 1 *bit* memiliki pengaruh yang sangat besar dalam dekripsi (menjadi tidak bermakna). Hal ini membuktikan bahwa mem-*bruteforce key* untuk mode apapun pada arsitektur YDIHYLFSIPFIF yang diusulkan akan lebih sulit dikarenakan *key* yang mirip tidak memberikan informasi yang bermakna.

4.5. Plainteks dengan sebagian kerusakan

Pada bagian ini, plainteks yang digunakan adalah plainteks berikut: “qwertyuiopasdfghjklzxcvbnm”. Hasil diberikan dalam tabel, di mana untuk setiap mode akan digunakan enkripsi dengan plainteks yang diubah menjadi “qwestyuiopasdfghjklzxcvbnm” (perubahan 1 *bit* r 114 menjadi s 115). *Key* yang digunakan adalah “password”.

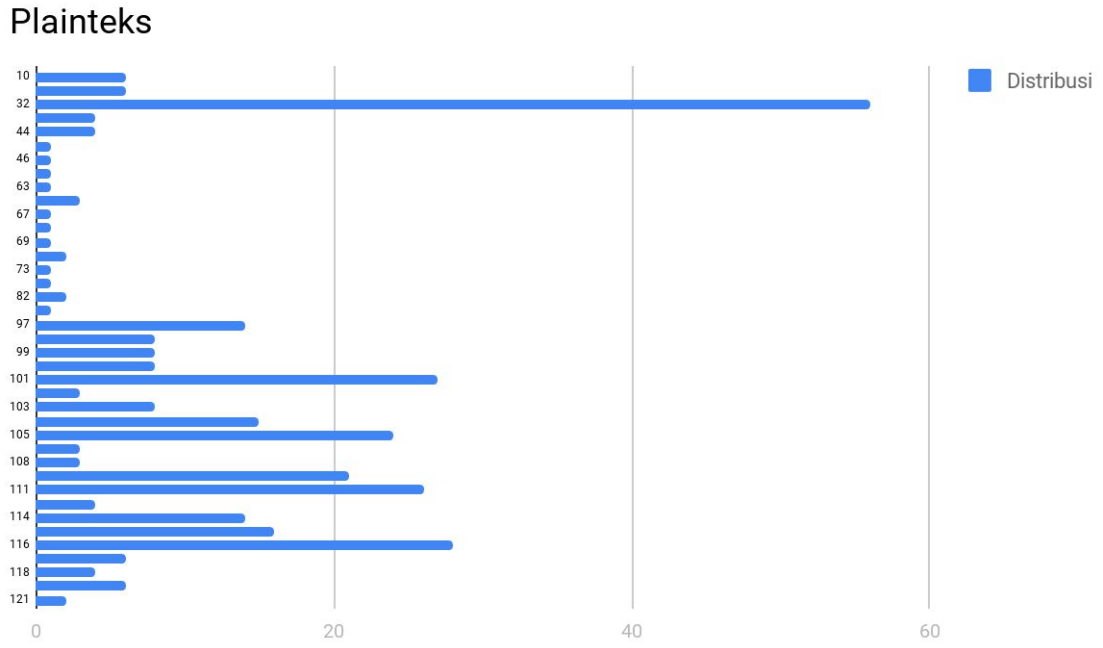
Metode yang digunakan	**Hasil enkripsi “qwertyuiop...”**	**Hasil enkripsi “qwestyuiop...”**
ECB	`^[;\X^GAJ`^Z^Pm6^Y^N x e5BB8`	`^X fe Aj`^Z^Pm6^Y^N x e5BB 8`
CBC	`^[;\X^GS}^?.0V^T )^_DE v^[f(n^F`	`^X fe ]^R7'G^AU^?^B^AA. 9`
Counter	`^[;\X^G&r!w , ù^?^[(A^V`	`^X fe &r!w , ù^?^[(A^V`

Pada tabel di atas, ditemukan bahwa untuk ECB dan *counter*, hasil enkripsi plainteks yang mengalami perubahan 1 *bit* hanya mengubah blok yang bersesuaian (*highlight* kuning). Karena proses *cipher* menggunakan *round* berulang dan pergeseran *bit*, maka satu blok penuh akan terpengaruh oleh kesalahan ini. Pada *counter*, dikarenakan perubahan dari 1 *bit* pada plainteks tidak mempengaruhi *counter* untuk blok-blok selanjutnya (yang hanya secara *strict* naik 1), cipherteks blok selanjutnya tidak terpengaruh.

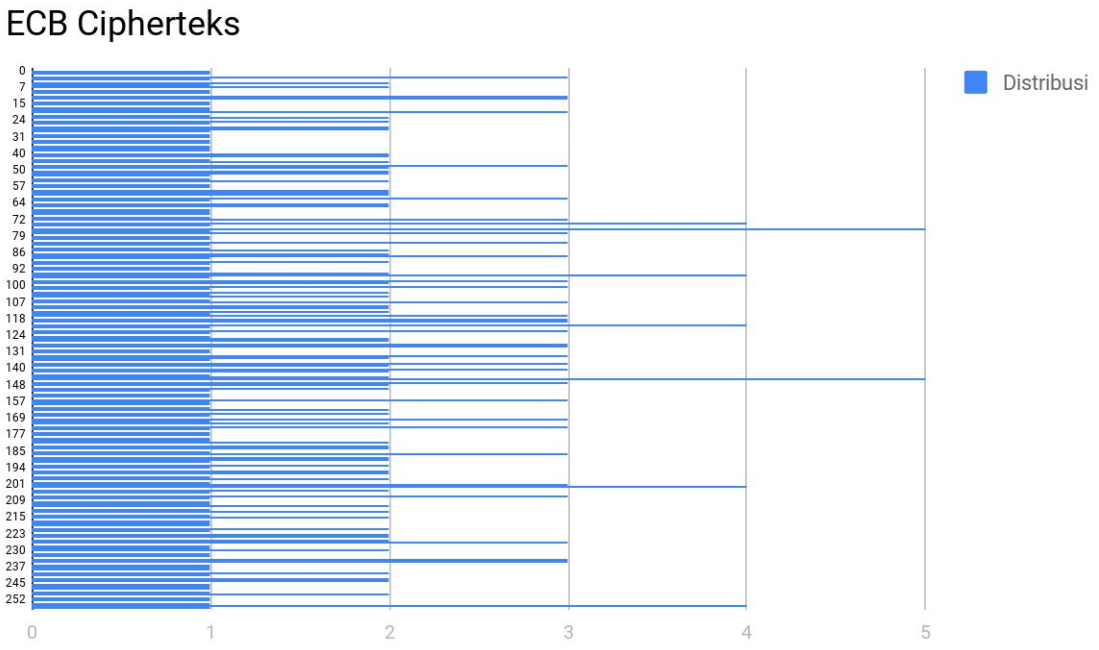
Pada CBC, dikarenakan nilai vektor yang mempengaruhi cipherteks pada blok selanjutnya dipengaruhi oleh hasil cipherteks pada blok sebelumnya, maka keseluruhan cipherteks pada blok sendiri dan seterusnya akan mengalami perubahan apabila plainteks memiliki sedikit perbedaan (difusi tinggi).

4.6. Statistika dari setiap byte pada plainteks dan cipherteks

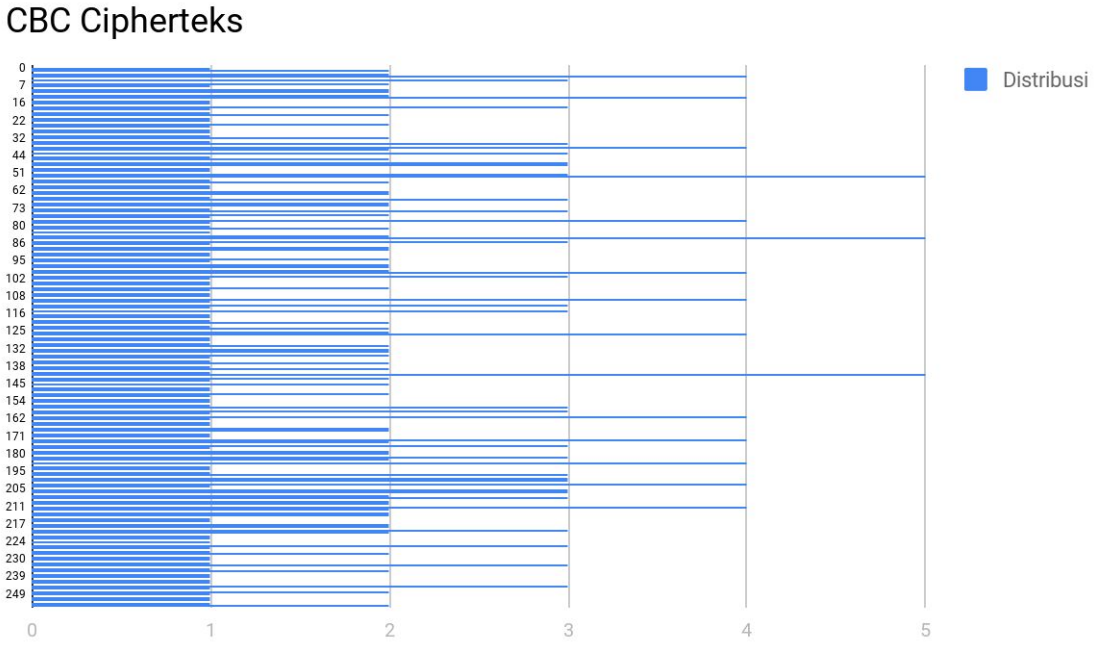
Untuk membuktikan penerapan prinsip *confusion* lebih lanjut terhadap arsitektur YDIHYLFSIPFIF yang diusulkan, kami melakukan uji coba menggunakan statistika berupa kemunculan setiap *byte* pada plainteks dan cipherteks. Pada bagian ini akan diberikan empat grafik (sumbu *x* adalah jumlah kemunculan, sumbu *y* adalah *byte*), masing-masing menjelaskan statistika kemunculan *byte* pada plainteks, cipherteks dengan ECB, cipherteks dengan CBC, dan cipherteks dengan *counter*.



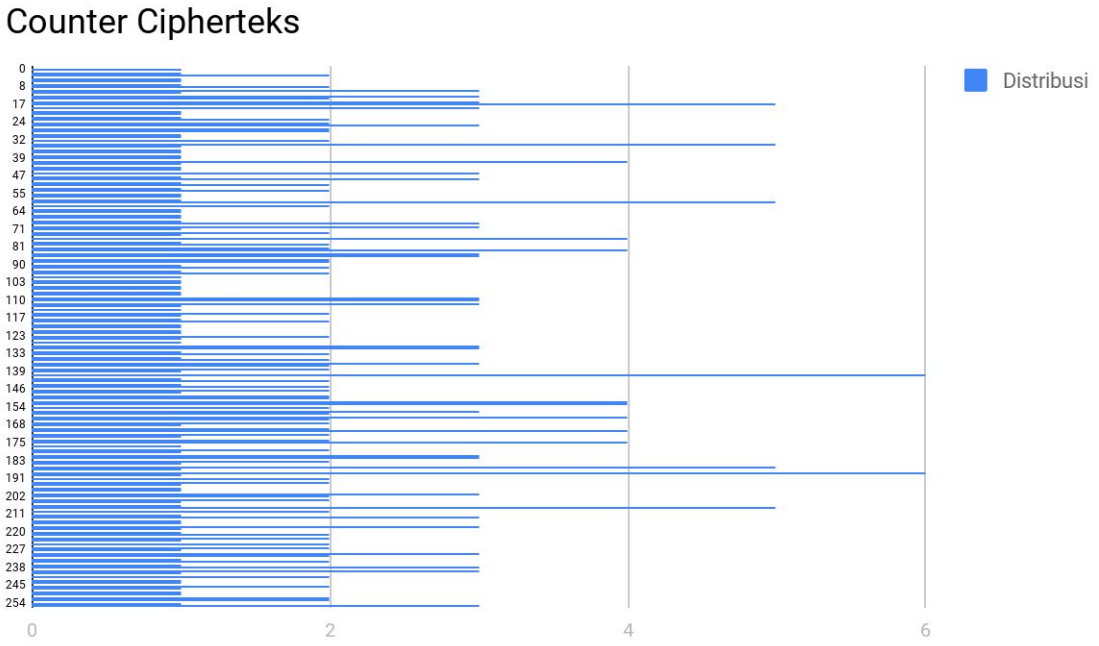
Gambar 15. Distribusi byte pada plainteks



Gambar 16. Distribusi byte pada cipherteks dengan mode ECB



Gambar 17. Distribusi byte pada cipherteks dengan mode CBC



Gambar 18. Distribusi byte pada cipherteks dengan mode *counter*

Dapat dilihat bahwa ketiga mode *cipher* dapat dikatakan memiliki nilai *confusion* yang baik dikarenakan tipe *byte* yang digunakan sangat banyak dan jumlah *byte* yang digunakan sudah tidak merepresentasikan statistika seperti bentuk data pada plainteks, di mana untuk ECB dan CBC, *occurrence* terbesar dari *byte* tertentu adalah 5 buah, untuk *counter* 6 buah, sedangkan untuk plainteks semula dapat mencapai 56 buah.

4.7. Analisis Keamanan

Dengan 16 jumlah putaran, proses substitusi, dan proses transposisi, algoritma ini sudah sangat aman dari proses kriptanalisis. Dapat dilihat dari analisis statistik sebelumnya bagaimana algoritma YDIHYLFSIPFIF mampu menyamarkan hubungan antara plainteks, cipherteks, dan kunci dengan sangat baik. Kriptanalisis akan kesulitan untuk memecahkan cipherteks karena distribusi dari setiap bit sudah tersebar. Dengan adanya proses transposisi dan perulangan pada *enciphering* juga semakin mengacak cipherteks yang dihasilkan.

Namun walaupun begitu, ancaman *brute force* terhadap kunci masih dapat terjadi. Walaupun hal tersebut sangat sulit untuk dilakukan karena panjang kunci internal yang digunakan oleh algoritma ini adalah sebesar 256-bit. Sehingga, akan terdapat 2^{256} kombinasi kemungkinan untuk kunci *internal*. Jika sebuah komputer dapat mencoba 2^{32} kemungkinan kunci per detik, maka dibutuhkan 2^{224} detik (8.5×10^{59} tahun) untuk mencoba semua kemungkinan kunci. Dengan hal ini, dapat dikatakan bahwa saat ini algoritma YDIHYLFSIPFIF sangat aman untuk digunakan.

5. Kesimpulan dan Saran Pengembangan

Algoritma YDIHYLFSIPFIF telah mampu untuk memenuhi prinsip *confusion* dan *diffusion* yang kuat sehingga akan menyulitkan proses kriptanalisis. Kemudian, perubahan (perusakan) satu bit pada plainteks akan mengubah cipherteks. Selain itu proses enkripsi dengan perubahan (perusakan) satu bit pada cipherteks juga akan mempengaruhi proses *deciphering* menjadi plainteks. Namun tentu saja perubahan akan bergantung dengan mode operasi yang digunakan. Apabila dengan ECB dan *counter*,

sebagian plainteks masih dapat dipecahkan dengan baik. Sedangkan pada mode CBC, tentu saja kerusakan akan menjalar karena adanya keterkaitan antar blok.

Saran pengembangan dari penelitian mengenai algoritma yang diusulkan, YDIHYLFSIPFIF, adalah untuk melakukan pengujian juga terhadap fungsi *scramble*, apakah persebarannya sudah cukup acak, dengan metode metrik yang sesuai. Selain itu, pengujian menggunakan metode kriptanalisis yang biasa digunakan untuk mengevaluasi metode cipher blok pada umumnya dapat diberikan pada bab eksperimen agar dapat menghasilkan nilai praktis mengenai tingkat evaluasi yang dapat dibandingkan dengan metode cipher blok populer sebagai bentuk studi analisis komparatif.

6. Daftar Pustaka

[1] E. Biham, A. Shamir. (1993). *Differential Cryptanalysis of the Data Encryption Standard*. Springer Verlag, New York.

[2] Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. (2018). *Handbook of applied cryptography*.

[3] National Bureau of Standards, U.S. (1977). *Data Encryption Standard~ Federal Information Processing Standard (FIPS)*, Publication 46, Department of Commerce, Washington D.C.

[4] Munir, Rinaldi. (2019). *"Kriptografi"*. Bandung: Informatika.

Acknowledgments

Pertama-tama penulis mengucapkan terima kasih dan puji syukur kepada Allah SWT atas limpahan rahmat dan karunianya berupa kemudahan sehingga penulis dapat menjalani kuliah di Institut Teknologi Informasi Bandung hingga semester 7 ini dengan selamat. Selanjutnya penulis mengucapkan terima kasih kepada orang tua dan keluarga penulis yang selalu mendoakan dan mendukung kesuksesan penulis. Penulis juga mengucapkan terimakasih atas melimpahnya ilmu yang selalu diberikan oleh dosen Teknik Informatika ITB khususnya kepada Bapak Rinaldi Munir yang mendorong penulis dalam penyusunan makalah ini.