ICBC

Inverse Circular Block Cipher

Andreas Dwi Nugroho (13511051)
Teknik Informatika
Institut Teknologi Bandung
Bandung, Indonesia
andreasdwin@gmail.com

Erwin (13511065)
Teknik Informatika
Institut Teknologi Bandung
Bandung, Indonesia
erwin9huang@gmail.com

***Abstrak*—Melalui makalah ini, diusulkan sebuah algoritma enkripsi dan dekripsi berbasis *block cipher* baru yang bernama <NAMA ALGO>. *Block cipher* ini menggunakan blok sepanjang 64-bit dan kunci yang panjangnya 128-bit. Dalam rancangan *block cipher* ini, untuk membuat algoritma sekompleks mungkin sehingga sukar dipecahkan maka digunakan operasi seperti permutasi, substitusi, pergeseran bit, perulangan, dan XOR untuk meningkatkan kompleksitas. Di dalam rancangan ini, dijelaskan pula pembentukan kunci internal dan fungsi dalam sebuah *cipher* berulang**

Kata kunci—block cipher; enkripsi; dekripsi; kunci

I. PENDAHULUAN

Seiring berkembangnya teknologi komputasi, banyak informasi sensitif yang harus dijaga kerahasiaannya. Untuk itu, kriptografi mengambil peranan penting untuk melindungi informasi-informasi tersebut. Salah satu kualitas kriptografi adalah ditentukan dari kekuatan enkripsi-dekripsi dari algoritma kriptografi terkait. Kekuatan yang dimaksud adalah seberapa susahnya sebuah *ciphertext* dapat dipecahkan kembali menjadi *plaintext*.

Algoritma enkripsi konvensional sangat mudah dipecahkan dengan metode seperti serangan frekuensi kemunculan huruf. Hal ini menuntut perkembangan algoritma yang lebih kuat seperti penanganan dalam sekelompok bit. Cara ini juga dapat dipecahkan dengan melakukan serangan frekuensi terhadap kemunculan kelompok bit yang sama. Oleh karena itu, diperlukan algoritma yang dapat membuat relasi antara *plaintext* dan *ciphertext* serumit mungkin.

Salah satu cara yang sering digunakan untuk meningkatkan kekuatan enkripsi pada *block cipher* adalah metode *confusion* dan *diffusion* dari Shannon. Metode ini tidak dapat diserang dengan metode konvensional karena perubahan 1 bit pada *plaintext, ciphertext,* ataupun *key* akan menyebabkan perubahan drastis secara keseluruhan.

Untuk menambah kekuatan enkripsi, digunakan jaringan Feistel. Jaringan Feistel menggunakan prinsip XOR dan membagi *plaintext* ke dalam 2 blok yang seimbang. Kedua blok tersebut saling mempengaruhi hasil yang satu dengan yang lain dalam setiap bit yang dioperasikan yang menyebabkan jaringan ini mempunyai tingkat kerumitan yang tinggi.

II. DASAR TEORI

A. *Confusion dan Diffusion Shannon*

Confusion dimaksudkan untuk membuat relasi antara *plaintext, ciphertext,* dan *key* sekompleks mungkin. Salah satu cara yang mendasari prinsip *confusion* adalah dengan menggunakan fungsi kompleks yang tidak linear. Hal ini akan membuat hubungan antara ketiganya menjadi sulit dihubungkan antara satu dengan yang lain.

Diffusion mengandung arti bahwa setiap bit dari *ciphertext* bergantung kepada beberapa bit dalam *plaintext*. Salah satu cara yang mudah untuk mengaplikasikan *diffusion* adalah seperti pada persamaan di bawah.

$$y1 = x1 + x2 + x3 + x4 + k1 + k2 + k3 + k4$$

$$y2 = x2 + x3 + x4 + x5 + k2 + k3 + k4 + k5$$

$$\vdots$$

$$y8 = x8 + x1 + x2 + x3 + k8 + k1 + k2 + k3$$

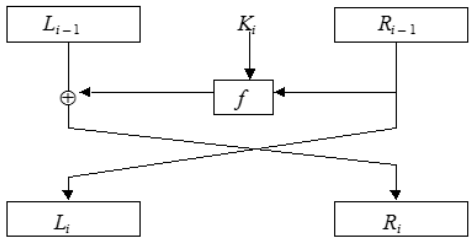
dengan x = *plaintext*, y = *ciphertext*, k = key (8 bit). Persamaan ini menjamin setiap bit pada *ciphertext* dipengaruhi setengah dari *plaintext* dan setengah dari *key*. Hal ini emngakibatkan perubahan sedikit pada *plaintext* ataupun *key* sangat mempengaruhi *ciphertext* sehingga prinsip difusi *ciphertext* ke dalam *plaintext* tercapai.

Sebuah metode yang cukup efektif untuk menjamin *confusion* dan *diffusion* adalah metode subsitusi-permutasi. Metode ini menjamin *diffusion* karena jika terjadi perubahan 1 bit pada *plaintext* maka perubahan ini akan dimasukkan ke dalam S-box (*Subsitution box*) beberapa kali sehingga *ciphertext* akan berubah drastis. Metode ini juga menjamin *confusion* karena sedikit perubahan pada kunci akan menyebabkan banyak perubahan pada bit-bit *cipher* dalam kondisi yang sangat kompleks akibat permutasi yang dikenakan berulang kali. Metode ini akan digunakan dalam makalah ini.

B. *Jaringan Feistel*

Jaringan *Feistel* adalah sebuah struktur simetris yang digunakan dalam *block cipher* . Jaringan ini banyak digunakan dalam *block cipher* masa kini. Oleh karena strukturnya yang

simetris, pengembangan enkripsi-dekripsi hampir sama yang menyebabkan waktu pengembangan menjadi lebih singkat.



Gambar 1 – Jaringan Feistel

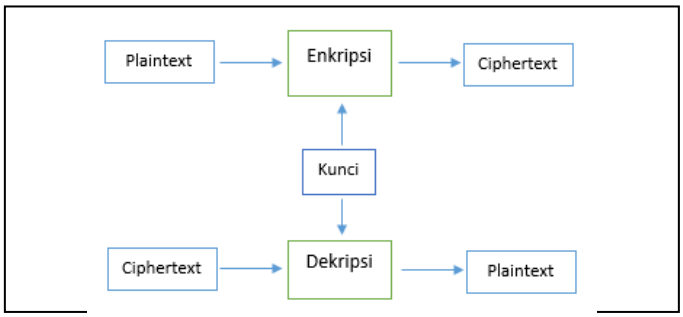
C. *Circular Shift*

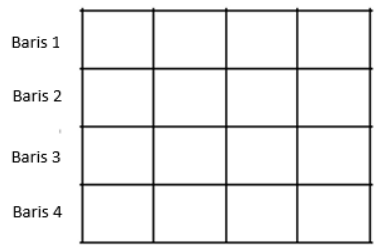
Circular Shift adalah sebuah metode yang digunakan untuk mengacak bit. Tujuan dilakukan *circular shift* ini adalah untuk membuat bit awal menjadi berbeda (besarnya perbedaan tergantung dari jumlah *shift* yang diterapkan) dengan bit hasil sehingga membuat algoritma enkripsi menjadi lebih rumit.

Circular shift terbagi atas 2 yaitu *circular left shift* dan *circular right shift*. *Circular left shift* akan melakukan pergeseran sejumlah bit ke arah kiri dengan bit paling kiri dipindah ke paling kanan, misalnya pergeseran sejumlah 1 bit dari 11001010 menjadi 10010101. Sedangkan *circular right shift* akan melakukan pergeseran sejumlah bit ke arah kanan dengan bit paling kanan dipindah ke paling kiri, misalnya pergeseran 1 bit dari 11001010 menjadi 01100101.

III. RANCANGAN *BLOCK CIPHER*

Secara umum, skema enkripsi dan dekripsi pada *block cipher* ditunjukkan pada Gambar 2. Rancangan algoritma *block cipher* yang diusulkan ini akan menggunakan prinsip diffusion dan confusion melalui proses substitusi, permutasi, pergeseran, dan perulangan. Serta dalam rancangan *block cipher* ini menggunakan jaringan feistel untuk mempermudah pembuatan algoritma dekripsi. Untuk penjelasan lebih detil dari rancangan *block cipher* yang diusulkan akan dijabarkan dalam subbab-subbab berikut.





A.

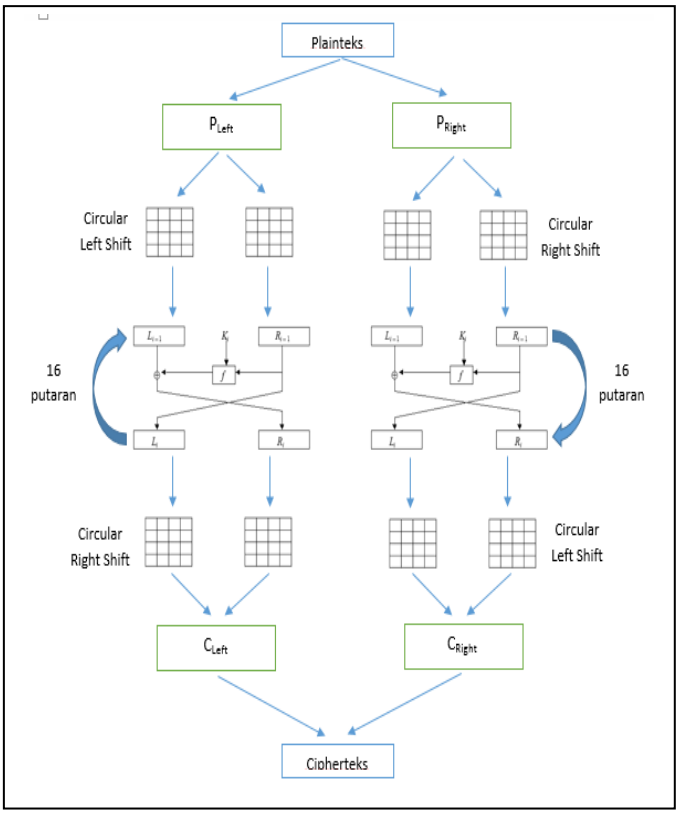
ci
al
bit

k
m
4-
g

diperlukan yaitu 128-bit. Selain itu, jumlah putaran yang dilakukan yaitu sebanyak 16 putaran.

Berdasarkan skema ini, awalnya plainteks yang berupa blok pesan berukuran 64-bit dibagi menjadi 2 bagian masing-masing panjangnya 32-bit. Kedua bagian tersebut kemudian dibagi menjadi 2 bagian blok lagi yang panjangnya sebanyak 16-bit dan disusun dalam sebuah matriks berukuran 4x4 sehingga ada 4 buah matriks blok. Untuk setiap matriks blok bagian kiri dimodifikasi dengan *circular left shift*. Sedangkan matriks blok bagian kanan dilakukan *circular right shift*. *Circular shift* yang digunakan untuk matriks ini menggunakan salah satu proses yang digunakan dalam algoritma Rijndael yaitu setiap baris dalam matriks tersebut dilakukan pergeseran ke kiri atau kanan sebanyak nomor baris tersebut. *Circular left shift* dapat dilihat pada Gambar 4, sedangkan untuk *circular right shift* prosesnya sama dengan *circular left shift* hanya arah pergeserannya ke arah kanan.

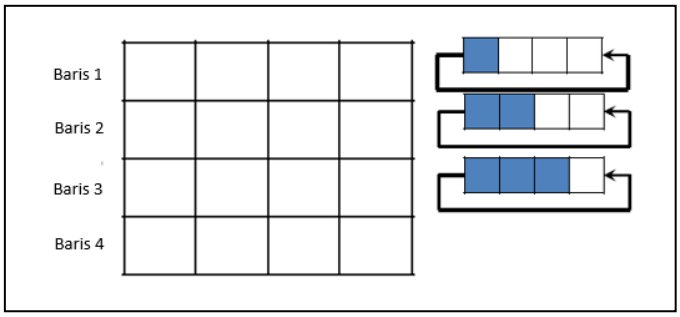
Kemudian matriks blok berukuran 16-bit tersebut akan menjadi masukan dalam jaringan Feistel. Skema ini menggunakan jaringan Feistel karena bersifat *reversible* untuk proses enkripsi dan dekripsi sehingga tidak perlu membuat algoritma baru untuk melakukan dekripsi. Proses dalam jaringan Feistel tersebut dilakukan berulang-ulang sebanyak 16 putaran yang tiap putaran menggunakan kunci internal sepanjang 16-bit yang dibangkitkan dari kunci eksternal dari masukan pengguna. Hasil dari proses ini yaitu dua buah blok yang berukuran 16 bit-bit.



Gambar 3 – Skema algoritma

Hasil dari jaringan Feistel kemudian direpresentasikan dalam bentuk matriks berukuran 4x4. Untuk masing-masing bagian kiri dan kanan terdapat 2 buah matriks yang dihasilkan. Kedua matriks blok bagian kiri dilakukan proses *circular right shift*, sedangkan untuk dua matriks blok bagian kanan dilakukan proses *circular left shift*.

Setelah dilakukan proses pergeseran dalam matriks blok, selanjutnya kedua matriks baik di bagian kiri maupun kanan digabung menjadi sebuah blok pesan yang panjang 32-bit. Diperoleh 2 buah blok pesan yang masing-masing panjangnya 32-bit. Lalu kedua blok tersebut digabung sehingga membentuk sebuah pesan berukuran 64-bit yang merupakan cipherteks dari proses enkripsi menggunakan skema ini. Untuk proses dekripsi dalam algoritma ini juga menggunakan skema yang sama dengan skema enkripsi namun urutan penggunaan kunci internal dalam proses perputaran dengan jaringan Feistel dibalik.



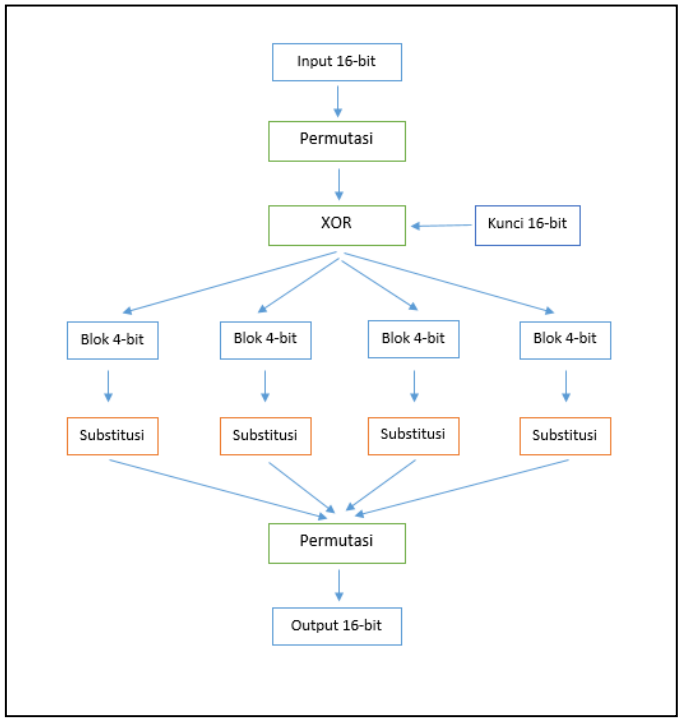
Gambar 4 – *Circular left shift*

B. *Round Function*

Round function merupakan fungsi f yang digunakan dalam jaringan Feistel. Fungsi ini akan menerima masukan berupa blok pesan yang panjangnya 16-bit. Kemudian dilakukan permutasi terhadap blok tersebut menggunakan sebuah P-Box (tabel permutasi). Hasil dari proses permutasi ini yaitu blok berukuran 16-bit. Hasil ini akan di-XOR-kan dengan kunci internal yang panjangnya 16-bit.

Setelah itu, blok berukuran 16-bit tersebut dibagi menjadi 4 bagian masing-masing berukuran 4-bit. Setiap blok 4-bit akan dilakukan proses substitusi dengan menggunakan sebuah S-Box (tabel substitusi) sehingga dalam fungsi f ini terdapat 4 buah S-Box yang berbeda. Setiap S-Box berisi permutasi angka 0 sampai 15.

Selanjutnya hasil substitusi dari semua S-Box digabung menjadi sebuah blok sepanjang 16-bit dan dilakukan proses permutasi lagi menggunakan P-Box yang sama dengan sebelumnya. Hasil dari fungsi f ini yaitu sebuah blok pesan yang panjangnya 16-bit. Fungsi f yang digunakan dapat dilihat pada Gambar 5.



Gambar 5 – *Round function*

C. *Pembangkitan Kunci Internal*

Dalam jaringan Feistel yang digunakan untuk algoritma *block cipher* ini dibutuhkan kunci sebanyak 16 dengan panjang setiap kuncinya 16-bit karena jumlah perputaran yang dilakukan juga 16 kali. Kunci-kunci tersebut akan dibangkitkan dari kunci eksternal yang panjangnya 128-bit. Kunci eksternal yang berukuran 128-bit dibagi menjadi 16 bagian sehingga diperoleh 16 bagian kunci yang masing-masing panjangnya 8-bit. Enam belas kunci tersebut dinotasikan dengan K_i, dengan *i* adalah nomor kunci.

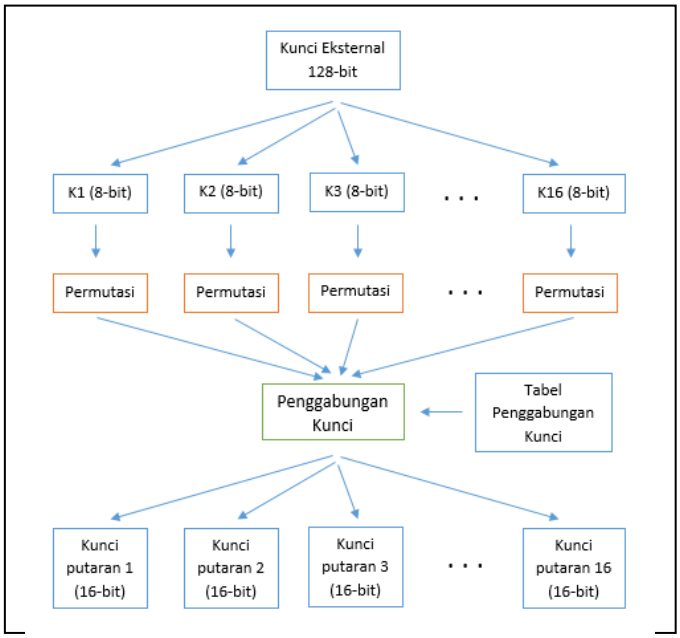
Selanjutnya setiap kunci dilakukan permutasi menggunakan sebuah P-Box yang sama dan menghasilkan kunci dengan panjang 8-bit. Setelah itu, karena setiap putaran membutuhkan kunci dengan panjang 16-bit maka dilakukan penggabungan 2 buah kunci dari 16 kunci sebelumnya sehingga menghasilkan sebuah kunci dengan panjang 16-bit. Untuk setiap putaran, penggabungan kunci-kunci tersebut disesuaikan dengan sebuah tabel penggabungan kunci yang ditunjukkan pada Tabel I.

Proses pembangkitan kunci internal ini dapat digambarkan pada Gambar 6.

TABLE I. PENGGABUNGAN KUNCI

Putaran	Gabungan Kunci
1	K1 ‖ K8
2	K2 ‖ K7
3	K3 ‖ K6
4	K4 ‖ K5
5	K5 ‖ K4
6	K6 ‖ K3
7	K7 ‖ K2
8	K8 ‖ K1

Putaran	Gabungan Kunci
9	K9 ‖ K16
10	K10 ‖ K15
11	K11 ‖ K14
12	K12 ‖ K13
13	K13 ‖ K12
14	K14 ‖ K11
15	K15 ‖ K10
16	K16 ‖ K9



Gambar 6 – Pembangkitan kunci internal

IV. EKSPERIMEN DAN PEMBAHASAN HASIL

Berdasarkan rancangan algoritma *block chipper* yang diusulkan, dilakukan implementasi algoritma tersebut menggunakan bahasa pemrograman Java. Dan selanjutnya dilakukan uji coba terhadap implementasi tersebut. Pengujian dilakukan dengan melakukan enkripsi dan dekripsi dari suatu pesan dan melihat hasilnya untuk mengetahui apakah rancangan algoritma *block cipher* ini benar-benar bisa mengenkripsi suatu pesan kemudian mendekripsinya menjadi pesan asli sehingga dapat digunakan untuk mengamankan pesan. Hasil pengujian proses enkripsi dan dekripsi dapat dilihat pada Gambar 7 dan Gambar8 .

```
Plaintext : i want to encrypt this message with nbc.

Key : nbcalgorithmkeynbcalgorithmkey

Ciphertext : □~□□z□□grl  <□ □v□8□□lqq □R□t□E □N ?j□
```

Gambar 7 - Proses enkripsi

```
Ciphertext : □~□□z□□grl  <□ □v□8□□lqq □R□t□E □N ?j□

Key : nbcalgorithmkeynbcalgorithmkey

Plaintext: i want to encrypt this message with nbc.
```

Gambar 8 - Proses dekripsi

Untuk pengujian enkripsi digunakan pesan dengan panjang 40-byte dan kunci dengan panjang 128-bit dan pengujian dekripsi menggunakan pesan cipherteks yang panjangnya juga 40-byte dari hasil proses enkripsi serta kunci yang sama dalam proses enkripsi.

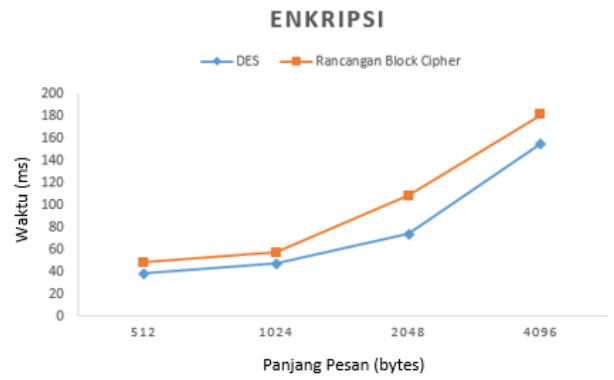
Hasil pengujian tersebut menunjukkan bahwa enkripsi menggunakan rancangan *block cipher* ini dapat mengubah bentuk pesan plainteks menjadi cipherteks yang susah untuk dibaca oleh seseorang. Dan berdasarkan hasil dekripsi juga menunjukkan bahwa dekripsi dalam rancangan *block cipher* ini dapat mengubah sebuah cipherteks menjadi plainteks yang dapat dipahami.

Pengujian selanjutnya dilakukan dengan melakukan enkripsi dan dekripsi suatu pesan dengn berbagai ukuran panjangnya dan menghitung waktu yang dibutuhkan untuk melakukan hal tersebut. Pengujian tersebut juga dilakukan dengan menggunakan salah satu algoritma yang sudah ada yaitu DES dan kemudian membandingkan hasil yang diperoleh dari kedua pengujian tersebut. Hasil pengujian ditunjukkan pada Tabel.

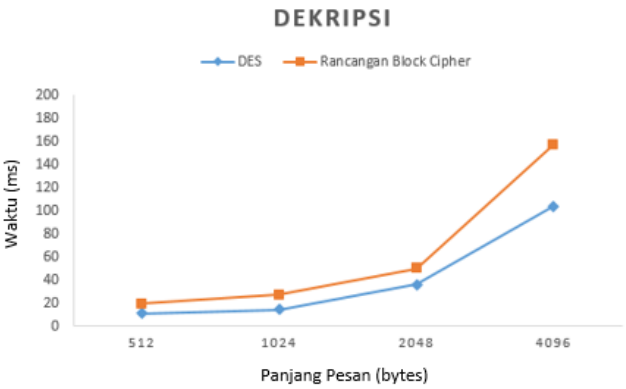
TABLE II. HASIL PENGUJIAN

Panjang Pesan	Waktu Enkripsi		Waktu Dekripsi	
	DES	Rancangan *Block cipher*	DES	Rancangan *Block cipher*
512 bytes	38 ms	48 ms	11 ms	19 ms
1024 bytes	47 ms	57 ms	14 ms	27 ms
2048 bytes	74 ms	108 ms	36 ms	50 ms
4096 bytes	155 ms	181 ms	104 ms	157 ms

Berdasarkan tabel hasil pengujian tersebut, berikut adalah grafik perbandingan untuk proses enkripsi dan dekripsi antara kedua algoritma.



Gambar 9 - Perbandingan waktu enkripsi



Gambar 10 - Perbandingan waktu dekripsi

Semakin panjang pesan yang akan diproses maka semakin banyak waktu yang dibutuhkan. Dari hasil pengujian, waktu yang digunakan untuk melakukan enkripsi dan dekripsi menggunakan rancangan *block cipher* ini ternyata lebih banyak dibandingkan dengan algoritma DES. Namun tidak ada keterkaitan antara rancangan *block cipher* dengan DES dalam perbandingan waktu yang digunakan. Waktu yang digunakan untuk enkripsi dan dekripsi bergantung pada proses komputasi dalam algoritma. Rancangan *block cipher* ini menggunakan proses komputasi dalam bentuk proses modifikasi bit atau byte seperti permutasi dan substitusi yang jumlahnya tidak sedikit sehingga waktu yang diperlukan pun juga sebanding dengan banyaknya komputasi yang dilakukan. Banyaknya penggunaan proses komputasi ini bertujuan untuk membuat algoritma menjadi sekompleks mungkin supaya algoritma ini sangat sukar untuk dipecahkan.

Keamanan *block cipher* juga bergantung pada panjang kunci yang digunakan. Semakin panjang kunci yang digunakan dalam *block cipher* maka semakin sukar untuk memecahkannya. Jika dalam perbandingan waktu sebelumnya rancangan *block cipher* ini lebih banyak menggunakan waktu enkripsi dan dekripsi namun dari segi keamanan berdasarkan panjang kunci maka kompleksitas rancangan algoritma ini lebih tinggi dibandingkan dengan DES karena panjang kuncinya lebih panjang yaitu 128-bit sedangkan panjang kunci DES yaitu 64-bit. Namun selain ini, ada lebih banyak algoritma yang kompleksitasnya di atas kompleksitas rancangan *block cipher* ini jika dilihat dari segi keamanan berdasarkan panjang kunci yang digunakan.

V. ANALISIS KEAMANAN

Analisis keamanan dari rancangan algoritma *block cipher* ini dilakukan berdasarkan kompleksitas dari algoritma itu sendiri. Keamanan *block cipher* bergantung pada panjangnya kunci yang digunakan. Jika kunci yang digunakan semakin panjang maka kompleksitas untuk memecahkan algoritma tersebut semakin besar sehingga semakin sukar. Jika menggunakan metode brute-force attack untuk menemukan kunci yang tepat untuk mendekripsi suatu cipherteks maka perlu dilakukan percobaan sebanyak 2^{128} kali karena panjang kunci yang digunakan dalam rancangan *block cipher* ini adalah 128-bit. Hal itu tentunya membutuhkan waktu yang sangat lama untuk melakukan semua percobaan tersebut.

Untuk membuat algoritma ini semakin kompleks sehingga algoritma enkripsi manjadi sangat sukar dipecahkan maka dalam rancangan *block cipher* ini digunakan prinsip diffusion dan confusion dari Shannon. Efek diffusion dan confusion tersebut diberikan melalui operasi substitusi, permutasi, serta perulangan. Selain itu, untuk menambah kerumitan, digunakan juga struktur jaringan Feistel pada proses enkripsi maupun dekripsi blok pesan

VI. KESIMPULAN DAN SARAN

Kesimpulan yang diperoleh yaitu melalui makalah ini telah diusulkan rancangan algoritma *block cipher* bernama <NAMA ALGO> yang dapat melakukan proses enkripsi dan dekripsi. Rancangan algoritma tersebut menerapkan prinsip diffusion dan confusion dengan melakukan proses substitusi, permutasi, pergeseran dengan circular shift, dan perulangan. Rancangan ini juga menggunakan jaringan Feistel untuk menambah kerumitan serta mempermudah pembuatan algoritma untuk dekripsi. Berdasarkan rancangan tersebut, telah diimplementasikan algoritma *block cipher* ini menggunakan bahasa pemrograman Java dan dilakukan pengujian terhadap implementasi tersebut. Hasil pengujian menunjukkan rancangan algoritma ini bisa melakukan enkripsi dan dekripsi. Selain itu waktu yang diperlukan untuk enkripsi dan dekripsi bergantung dari jumlah proses komputasi yang dilakukan.

Saran terkait dengan rancangan algoritma *block cipher* ini yaitu sebaiknya dilakukan pengembangan lebih lanjut sehingga keamanan dari algoritma ini semakin tinggi serta dapat memperkecil jumlah waktu yang dibutuhkan untuk proses enkripsi dan dekripsi. Semakin banyak panjang pesan maka semakin lama untuk mengenkripsi maupun mendekripsi. Oleh karena itu waktu merupakan hal yang penting untuk diperhatikan. Selain itu, sebaiknya dilakukan juga analisis dan pengujian keamanan lebih lanjut untuk menguji dan mengetahui seberapa dalam tingkat keamanan yang diberikan dari rancangan algoritma *block cipher* ini.

DAFTAR REFERENSI

[1] Rijndael Algorithm (Advanced Encryption Standard) AES: https://www.lri.fr/~fmartignon/documenti/systemesecurite/5-AES.pdf, tanggal akses: 11 Maret 2015.

[2] L., Debra.2004,*Elastic Block Ciphers*: *The Feistel Cipher Case*: Colombia University

[3] Hsiao, Jeffrey. 2012. *MARC – A New Block Cipher Algorithm*: Contemporary Engineering Sciences

Lampiran A – Gambar skema *block cipher*