3RF Block Cipher

Mokhammad Ferdi Ghozali - 13515014
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13515014@std.stei.itb.ac.id

Muhammad Umar Fariz Tumbuan - 13515050
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13515050@std.stei.itb.ac.id

Kode sumber tersedia di: https://github.com/agent-whisper/3RF_cipher

***Abstrak*—Kriptografi adalah sebuah konsep yang telah diterapkan lama di dalam sejarah. Di zaman modern sekarang, kriptografi semakin diperlukan karena volume proses pertukaran informasi, baik privat maupun publik, terus bertambah. Namun, sebuah algoritma cipher suatu saat bisa terpecahkan, sehingga upaya membuat dan mengembangkan algoritma cipher perlu terus dilakukan. Algoritma 3RF merupakan sebuah algoritma cipher yang berusaha berinovasi dengan menggunakan jaringan Feistel termodifikasi serta dengan menggunakan lebih dari satu fungsi dalam melakukan satu iterasi pada jaringan Feistel.**

Keywords—Kriptografi; Block cipher; Feistel; Hash; P-Box; S-Box; Transformasi.

I. Pendahuluan

Kriptografi merupakan sebuah algoritma untuk menghilangkan makna sebuah pesan. Penggunaan kriptografi paling awal yang tercatat dalam sejarah dilakukan oleh Julius Caesar pada masa kekaisaran Romawi [3]. Pada masa itu, Julius Caesar membuat sebuah algoritma kriptografi bernama *Caesar Cipher*. Pada masa itu, algoritma sederhana seperti *Caesar Cipher* masih cukup aman untuk digunakan. Namun pada masa digital saat ini di mana informasi tersebar luas melalui Internet, algoritma kriptografi baru yang lebih mutakhir perlu dikembangkan.

Berbagai algoritma kriptografi modern telah dibuat dan beberapa juga telah dijadikan standar. Salah satu algoritma yang paling dikenal adalah algoritma *Data Encryption Standard* (DES) [2]. DES merupakan algoritma *block cipher* dengan *symmetric-key* dan pada pertengahan tahun 1970 diakui sebagai standar dunia. Namun DES pada akhirnya dapat dipecahkan, sehingga akhirnya tergantikan dengan standar terbaru yaitu *Advanced Encryption Standard* [4]. Fenomena tersebut menunjukkan bahwa selalu ada potensi suatu saat algoritma kriptografi yang digunakan akan terpecahkan. Karena itulah algoritma kriptografi, baik yang sudah ada maupun yang baru, perlu terus dikembangkan.

Berbagai usaha untuk mengembangkan teknik kriptografi-baru telah dikembangkan. Salah satu usaha yang dilakukan adalah melakukan modifikasi terhadap jaringan Feistel yang digunakan. R. Anderson dan E. B. Iham mengembangkan algoritma "Bear and Lion" [6] yang menggunakan *unbalanced feistel network* dan memberikan hasil yang bagus. Usaha serupa juga dilakukan oleh R. Hartanto dan D. Novanto dengan algoritma NH2 [7] yang menambah kompleksitas dengan melakukan pemanggilan fungsi kedua. Hal ini menunjukkan bahwa modifikasi terhadap jaringan Feistel memiliki potensi untuk menghasilkan algoritma cipher yang baik.

Algoritma 3RF terinspirasi terutama oleh pekerjaan [7]. Jaringan feistel pada [7] dimodifikasi menjadi lebih kompleks dengan melakukan rotasi kedua. Selain itu, algoritma 3RF menggunakan tiga jenis fungsi dalam sebuah iterasi. Oleh karena itu, kontribusi dari makalah ini adalah:

1) Mengembangkan jaringan Feistel yang lebih kompleks dengan menambah jumlah rotasi yang dilakukan.
2) Membuat kerangka kerja untuk menggunakan tiga jenis fungsi dalam sebuah iterasi.

Untuk selanjutnya, makalah akan disusun dengan susunan sebagai berikut. Bagian dua akan membahas dasar teori yang relevan dengan pekerjaan yang dilakukan. Bagian tiga akan membahas rancangan detail dari algoritma 3RF. Bagian empat akan membahas hasil eksperimen penggunaan algoritma 3RF. Bagian kelima akan mengestimasi keamanan algoritma 3RF dari beberapa sudut pandang. Bagian terakhir akan membuat kesimpulan dan hasil eksperimen serta membahas bagaimana penelitian ini bisa dikembangkan lebih lanjut.

II. Dasar Teori

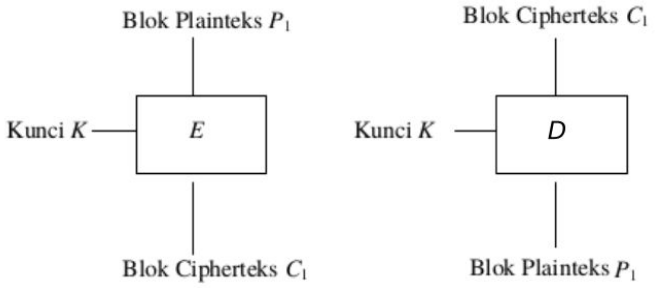
A. Block Cipher

Block cipher adalah sebuah fungsi yang memetakan blok plainteks sepanjang *n*-bit ke blok cipherteks sepanjang *n*-bit [2]. Secara garis besar, block cipher dapat dianggap sebagai sebuah algoritma substitusi terhadap sekumpulan karakter yang berjumlah banyak. Hal ini menjadi perbedaan utama dengan *stream cipher* yang beroperasi dengan unit yang lebih kecil.

Dalam mengembangkan sebuah *block cipher*, terdapat lima jenis mode operasi yang dapat digunakan [3, 5]:

1) Electronic Codebook (ECB)

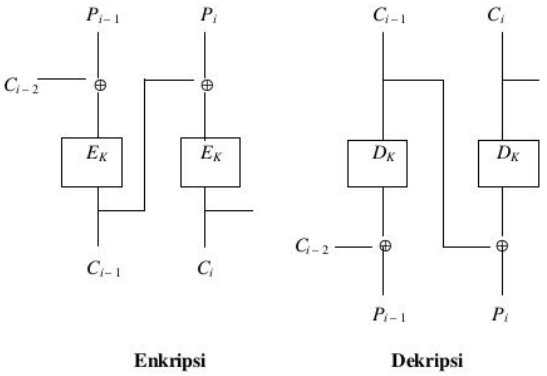
Pada mode operasi ECB, setiap blok plainteks diproses secara independen [3, 4]. Skema mode operasi ECB diberikan pada Gambar 1. Sebutan "*codebook"* diberikan kepada mode operasi ini karena sebuah blok plainteks secara teori akan selalu terenkripsi menjadi blok cipherteks yang sama. Karakteristik ini merupakan kelemahan utama dari mode operasi ECB. Namun kelemahan tersebut dapat diatasi dengan menggunakan ukuran blok yang besar, sehingga ukuran *"codebook"* yang dihasilkan terlalu besar untuk dianalisis.



Gambar 1. Skema enkripsi dan dekripsi dari mode operasi ECB [3].

2) Cipher Block Chaining (CBC)

Mode operasi CBC menambahkan mekanisme *feedback* dalam penerapan *block cipher* [3, 5]. Hal ini dilakukan dengan melakukan operasi XOR terhadap blok plainteks yang akan dienkripsi dengan cipherteks hasil enkripsi blok sebelumnya. Proses enkripsi baru dilakukan setelah operasi XOR selesai dilakukan. Cipherteks yang dihasilkan kemudian digunakan untuk mengenkripsi blok plainteks selanjutnya. Untuk melakukan dekripsi dilakukan proses sebaliknya. Skema dari mode operasi CBC diberikan pada Gambar 2. Untuk mengenkripsi blok plainteks pertama, sebuah *initialization vector (IV) block* digunakan untuk melakukan operasi XOR. Blok tersebut dapat diberikan oleh pengguna atau dibuat secara random.

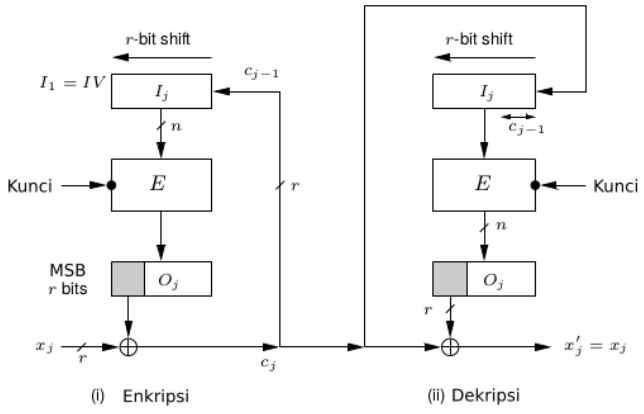


Gambar 2. Skema enkripsi dan dekripsi dari mode operasi CBC

3) Cipher Feedback (CFB)

Mode operasi CBC mengenkripsi plainteks sebanyak *n*-bit pada satu waktu. Namun beberapa aplikasi *block cipher* terkadang butuh mengenkripsi dan mengirim data tanpa jeda yang terlalu lama [2]. Mode operasi CFB memungkinkan fungsi *block cipher* mengenkripsi *r*-bit pada satu waktu dengan $r < n$ (umumnya $r = 1$). Skema mode operasi CFB diberikan pada Gambar 3. Komponen yang digunakan sebagai *feedback* adalah cipherteks hasil enkripsi sepanjang *r*.

Mekanisme *feedback* pada mode operasi CFB menyebabkan proses dekripsi dependen terhadap keutuhan cipherteks [2]. Jika terdapat bagian cipherteks yang salah atau tertukar, maka sisa hasil dekripsi akan menjadi salah. Penggunaan ukuran cipherteks yang kecil sebagai *feedback* memperkuat dependensi lebih jauh.

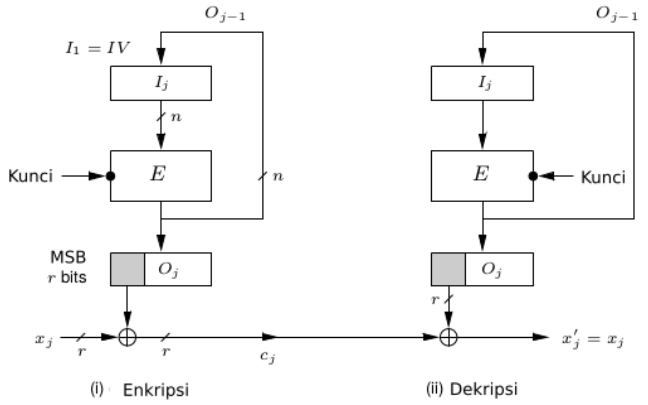


Gambar 3. Skema (i) enkripsi dan (ii) dekripsi dari mode operasi CFB [2].

4) Output Feedback (OFB)

Mode operasi OFB kurang lebih sama dengan mode operasi CFB. Hal yang membedakan keduanya adalah komponen yang digunakan sebagai *feedback*. Alih-alih

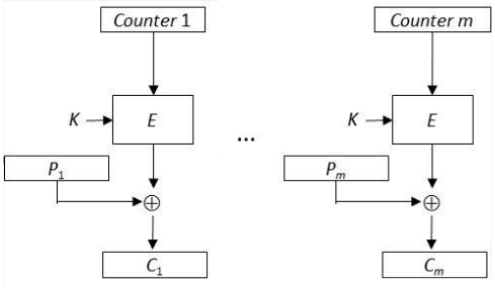
menggunakan cipherteks untuk mengubah *IV block*, OFB menggunakan MSB dari keluaran (*output*) enkripsi *IV block* itu sendiri sebagai komponen *feedback* [2]. Skema mode operasi OFB diberikan pada Gambar 4. OFB cocok digunakan jika perambatan kesalahan seperti yang terjadi proses enkripsi CBC dan proses dekripsi CFB ingin diminimalisir [2].



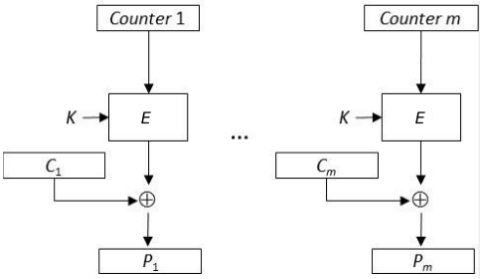
Gambar 4. Skema (i) enkripsi dan (ii) dekripsi dari mode operasi OFB [2].

5) Counter Mode (CM)

Mode operasi CM merupakan sebuah simplifikasi dari mode operasi OFB [2]. Mode operasi CM menggunakan sebuah *counter block* sebagai pengganti *IV block*. Untuk setiap proses enkripsi selanjutnya, nilai *counter block* diubah nilainya secara berkala (contohnya ditambah dengan nilai 1). Konsekuensi dari perubahan ini adalah CM tidak lagi menggunakan mekanis *feedback*. Namun CM mendapatkan *random-access property*, sehingga cipherteks tidak perlu didekripsi secara keseluruhan. Skema mode operasi CM diberikan pada Gambar 5.



(i) Enkripsi



(ii) Dekripsi

Gambar 5. Skema (i) enkripsi dan (ii) dekripsi mode operasi CM .

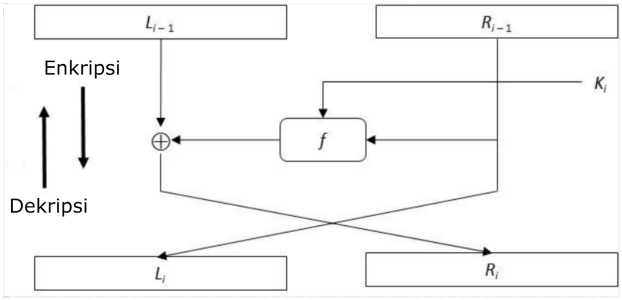
B. Confusion dan Diffusion

Menurut Shannon, *confusion* dan *diffusion* merupakan dua teknik dasar untuk mengurangi redudansi sebuah plainteks [1, 3]. *Confusion* berperan dalam upaya menghilangkan relasi antara plainteks dan cipherteks. Dengan menerapkan *confusion* pada algoritma cipher, usaha untuk mencari redundansi dan pola statistikal akan menjadi lebih sukar untuk dilakukan. Salah satu contoh teknik *confusion* adalah proses substitusi seperti pada *caesar cipher*. Pada algoritma cipher modern, proses substitusi yang dilakukan jauh lebih kompleks. Namun teknik *confusion* cenderung tidak dapat terlalu diandalkan jika tidak disertai dengan teknik lain.

Teknik *diffusion* adalah teknik yang berupaya menghilangkan redudansi pada sebuah plainteks dengan menyebarkan bagian-bagian plainteks pada cipherteks [1, 3]. Salah contoh teknik *diffusion* adalah dengan melakukan permutasi pada plainteks. Algoritma cipher modern umumnya menggunakan lebih dari satu jenis teknik untuk mendapatkan tingkat *diffusion* yang lebih tinggi.

C. Jaringan Feistel

Jaringan Feistel adalah sebuah cipher iteratif yang memetakan *2t*-bit plainteks (L_0, R_0), untuk *t*-bit blok L_0 dan R_0, ke sebuah cipherteks (R_r, L_r), yang dilakukan sebanyak *r*-ronde dengan $r \geq 1$ [2]. Jaringan Feistel banyak dipakai pada algoritma kriptografi modern, seperti DES, OKI, GOST, FEAL, Lucifer, Blowfish, dan lain-lain [5]. Salah satu alasan jaringan Feistel sering digunakan adalah karena sifatnya yang reversible [5], sehingga dalam pembuatannya cukup membuat satu algoritma yang dapat digunakan pada enkripsi dan dekripsi. Gambar 6 menunjukkan skema dasar sebuah jaringan Feistel sederhana.



Gambar 6. Skema jaringan Feistel [5].

D. *Transformasi Substitusi*

Transformasi substitusi merupakan proses memetakan bytes pada pesan kedalam bentuk bytes lain, berdasarkan S-Box. Berikut bentuk S-box yang kami gunakan sama dengan S-Box yang digunakan pada algoritma Rijndael.

hex	y															
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

x

Gambar 7. S-Box Rijndael [5].

Penggunaan S-box ini adalah dengan mengubah 2 hex sebelum menjadi 2 hex baru berdasarkan tabel, karakter pertama hex sebelum sebagai indeks baris, dan hex kedua sebagai indeks kolom. Sebagai contoh, hex “fe” akan diubah dengan hex “bb” yang berada pada tabel S-box baris “F” dan kolom “E”.

E. *Transformasi Permutasi*

Transformasi permutasi merupakan proses mengacak urutan bit dari suatu *byte*. Proses ini sama seperti proses permutasi pada algoritma DES. Berikut tabel P-Box yang digunakan.

16	7	20	21	29	12	28	17	1	15	23	26	5	8	31	10
2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25

Gambar 8. P-Box DES [5].

Urutan bit dalam byte diacak berdasarkan tabel tersebut, sehingga akan mengubah nilai dari byte itu sendiri. bit ke-16 akan menjadi bit pertama, dan dilanjutkan dengan bit ke-7 dan seterusnya.

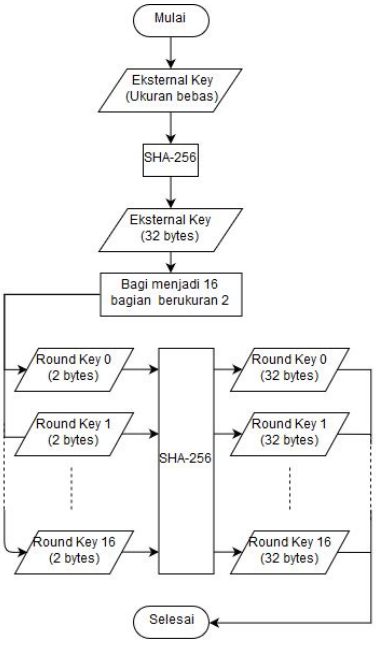
III. Rancangan Algoritma

A. *Ikhtisar Algoritma 3RF*

Algoritma 3RF adalah sebuah *block cipher* yang dibuat di atas sebuah jaringan Feistel yang telah dimodifikasi serta menggunakan algoritma SHA-256 dalam memproses kunci internal. Algoritma bekerja dalam satuan *byte* terhadap blok plainteks dan kunci eksternal yang berukuran 256-bit (32 *byte*). Pada sebuah iterasi di dalam jaringan Feistel, blok plainteks akan diproses menggunakan kombinasi tiga jenis *round functions* yang urutan penggunaan bergantung pada kunci internal yang berkoresponden dengan iterasi

B. *Key Scheduling*

Kunci yang dimasukkan oleh pengguna pertama-tama akan di-*hash* menggunakan algoritma SHA-256. Dengan melakukan *hashing*, pengguna bisa memasukkan kunci apapun yang mereka inginkan dan algoritma tetap akan bekerja. Hasil *hash* yang berukuran 32 *byte* kemudian dibagi menjadi 16 bagian. Setiap bagian tersebut kemudian di-*hash* kembali menggunakan algoritma SHA-256, menghasilkan *round key* berukuran 32 *byte*. Setiap *round key* akan digunakan pada iterasi dengan urutan yang sama. Gambar 9 menunjukkan skema *key scheduling* dari algoritma 3RF.



Gambar 9. Skema *key scheduling* dari Algoritma 3RF

C. Round Function

Round function dari Algoritma 3RF terdiri atas tiga jenis subfungsi. Dalam sebuah *round function*, ketiga jenis subfungsi bernama RF_A, RF_B, dan RF_C. Ketiga fungsi tersebut akan dijalankan dengan urutan yang dapat berbeda antara satu iterasi dengan iterasi lain.

Setiap subfungsi selalu diawali dengan operasi XOR antara plainteks dengan *round key* dan diakhiri dengan operasi permutasi. Perbedaan dari setiap sub-fungsi adalah operasi tambahan yang dilakukan di bagian tengah dirinya:

1) RF_A melakukan operasi transformasi dan operasi substitusi,
2) RF_B hanya melakukan operasi substitusi, dan
3) RF_C hanya melakukan operasi transformasi.

Penggunaan kombinasi subfungsi yang sebagian tidak melakukan operasi lengkap diharapkan dapat membuat hasil enkripsi yang didapatkan menjadi lebih kompleks dan tidak konsisten, sehingga sumber daya untuk melakukan kriptanalisis menjadi lebih tinggi.

Tabel 1 menunjukkan 6 permutasi urutan pemanggilan yang mungkin. Indeks permutasi ditentukan menggunakan persamaan berikut:

$$\textit{indeks permutasi} = \textit{sum(round key)} \bmod 6 \qquad (1)$$

Lokasi pemanggilan setiap subfungsi dalam sebuah iterasi akan dijelaskan pada bagian selanjutnya.

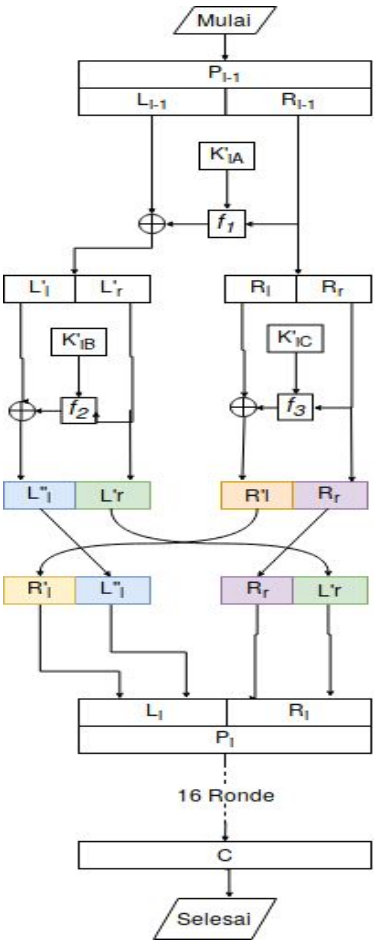
Tabel 1. Permutasi urutan pemanggilan subfungsi.

Indeks Permutasi	f_1	f_2	f_3
0	RF_A	RF_B	RF_C
1	RF_A	RF_C	RF_B
2	RF_B	RF_A	RF_C
3	RF_B	RF_C	RF_A
4	RF_C	RF_A	RF_B
5	RF_c	RF_B	RF_A

D. Struktur Jaringan Feistel

Jaringan Feistel yang digunakan algoritma 3RF diberikan pada Gambar 10. Panjang blok yang diproses adalah 256-bit atau sebanyak 32 *bytes*. Jaringan Feistel tersebut telah dimodifikasi untuk melakukan dua pemanggilan fungsi dan dua rotasi tambahan setelah dilakukan pembagian pertama. Proses sebuah iterasi pada jaringan Feistel adalah sebagai berikut:

1) *Round key* dibagi menjadi tiga bagian (K_A, K_B, K_C). K_A memiliki panjang 128-bit sedangkan K_B dan K_C memiliki panjang 64-bit. Pembagian *round key* pada Gambar 11.
2) Blok P dibagi dua menjadi blok L dan R yang masing-masing berukuran 128-bit.
3) Blok L dan R bersama dengan K_A digunakan sebagai masukan fungsi f_1 dan menghasilkan blok L' dan R.
4) Blok L' dibagi dua menjadi (L'_L, L'_R) yang masing-masing berukuran 64-bit. Kedua blok tersebut beserta K_B digunakan sebagai masukan f_2 dan menghasilkan (L''_L, L'_R).
5) Blok R dibagi dua menjadi (R_L, R_R) yang masing-masing berukuran 64-bit. Kedua blok tersebut beserta K_C digunakan sebagai masukan f_3 dan menghasilkan (R'_L, R_R).
6) Empat blok (L''_L, L'_R, R'_L, R_R) diposisikan menjadi (R'_L, L''_L, R_R, L'_R). Hasil perubahan posisi tersebut adalah keluaran dari sebuah iterasi.
7) Proses dilakukan sebanyak 16 iterasi.



Gambar 10. Jaringan Feistel yang digunakan Algoritma 3RF.

Gambar 11. Pembagian *round key*.

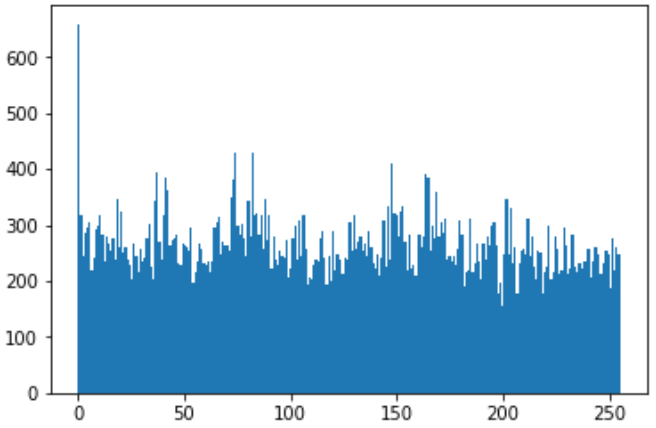
IV. Hasil dan Pembahasan Eksperimen

Eksperimen akan dilakukan menggunakan satu plainteks uji yaitu gambar digital berformat JPEG. Eksperimen pertama dilakukan untuk menganalisis statistika antara plainteks dengan cipherteks menggunakan kelima mode operasi. Eksperimen kedua dan ketiga dilakukan untuk menganalisis kualitas *confusion* dan *diffusion* dari algoritma 3RF. Plainteks yang digunakan adalah berkas gambar yang memiliki ukuran cukup besar untuk memudahkan analisis statistika.

A. Pengujian Statistika Hasil Enkripsi

Tampilan plainteks uji dan frekuensi kemunculan *byte*-nya ditunjukkan pada Gambar 12 dan Gambar 13. Gambar yang dijadikan plainteks memiliki ukuran 67,5 kB dengan dimensi 512x512 pixel. Hasil enkripsi plainteks tidak dapat ditampilkan karena proses enkripsi akan merusak *encoding* JPEG.

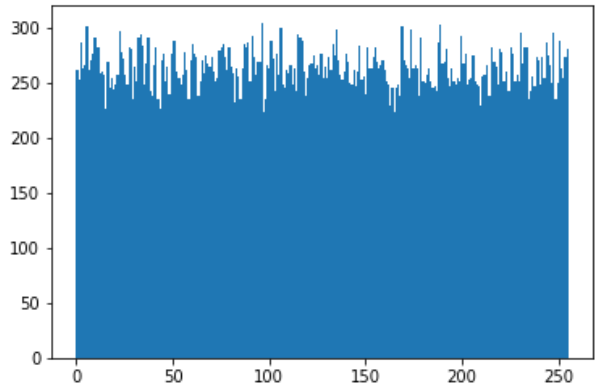
Gambar 12. Plainteks uji eksperimen.
(Sumber: https://4.bp.blogspot.com/-mLOwpEsNL4Y/UCu0wcVsPBI/AAAAAAAAA6s/7ECKTpxXr3o/s1600/lena.bmp)



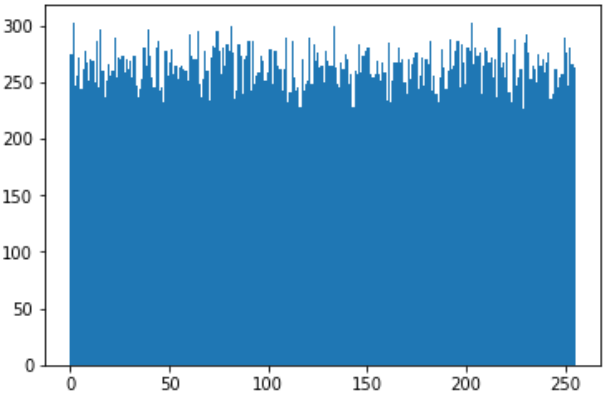
Gambar 13. Frekuensi kemunculan setiap nilai *bytes* pada plainteks.

Gambar 14 - 18 menunjukan hasil distribusi kemunculan byte dari file hasil enkripsi menggunakan metode ECB, CBC, Cipher Feedback, Output Feedback dan Counter Mode.

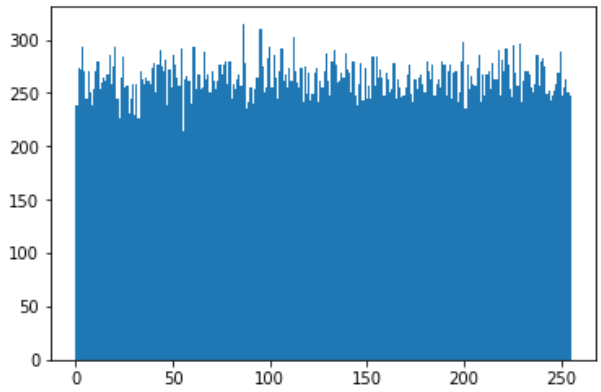
Dari gambar-gambar tersebut dapat dilihat bahwa distribusi kemunculan byte file cipher sudah berbeda dengan distribusi byte plain file. persebaran byte cipher juga lebih merata dibanding persebaran byte plain file. Hal ini menunjukan bahwa kualitas enkripsi tergolong baik.



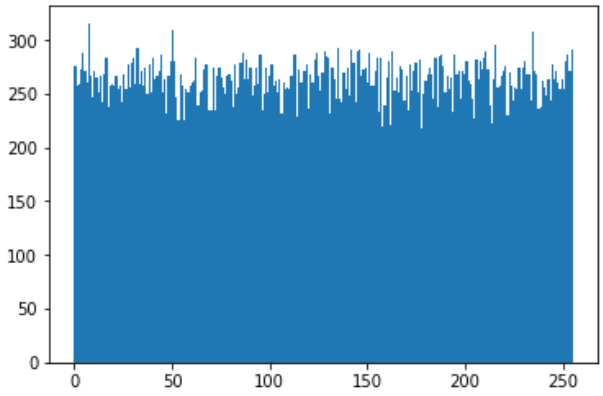
Gambar 14. Frekuensi kemunculan setiap nilai *bytes* setelah dienkripsi menggunakan ECB.



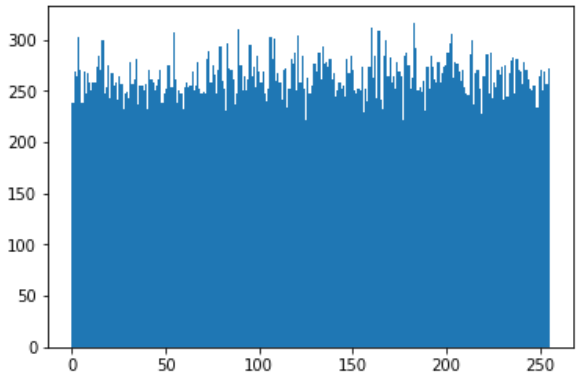
Gambar 15. Frekuensi kemunculan setiap nilai *bytes* setelah dienkripsi menggunakan CBC.



Gambar 16. Frekuensi kemunculan setiap nilai *bytes* setelah dienkripsi menggunakan CFB.



Gambar 17. Frekuensi kemunculan setiap nilai *bytes* setelah dienkripsi menggunakan OFB.



Gambar 18. Frekuensi kemunculan setiap nilai *bytes* setelah dienkripsi menggunakan CM.

B. Pengujian Confusion dan Diffusion Algoritma 3RF

Eksperimen yang kedua dilakukan untuk melihat dampak perubahan 1 bit pada plainteks dan kunci terhadap cipherteks yang dihasilkan. Pengujian *confusion* dilakukan dengan membandingkan hasil enkripsi berkas gambar menggunakan beberapa kunci yang berbeda satu bit dengan kunci awal. Kualitas *confusion* diukur dengan menghitung kemunculan nilai *byte* yang memiliki nilai dan posisi yang sama antara cipherteks pertama dan kedua. Kunci awal yang akan digunakan adalah "kriptografi". Kunci-kunci percobaan serta jumlah kesamaan dengan cipherteks awal diberikan pada Tabel 2. Hasil yang didapatkan menunjukkan bahwa algoritma 3RF memiliki *confusion* yang tinggi karena perubahan 1-bit pada kunci dapat menghasilkan dua cipherteks yang rata-rata memiliki perbedaan sebesar 99.61 persen.

Tabel 2. Hasil pengujian *confusion*.

Kunci	Jumlah Kesamaan *Bytes* pada Cipherteks	Persentase Perubahan
Iriptografi	231 / 67520	99,65
k**s**iptografi	269 / 67520	99,59
kr**h**ptografi	265 / 65720	99,60
kriptograf**j**	271 / 67520	99,59
kriptogra**g**i	264 / 65720	99,60
kriptogr**b**fi	248 / 65720	99,62
Rata-rata		99,61

Eksperimen yang ketiga ketiga dilakukan untuk melihat dampak perubahan 1 bit pada plaintext terhadap ciphertext. Pengujian *diffusion* dilakukan dengan membandingkan hasil enkripsi berkas gambar dengan hasil enkripsi berkas gambar yang sudah diubah 1 bit pada random byte. Kualitas *diffusion* diukur dengan menghitung kemunculan nilai *byte* yang memiliki nilai dan posisi yang sama antara cipherteks pertama dan kedua. Gambar pertama yang digunakan adalah gambar lena. Gambar kedua adalah gambar lena yang sudah diubah 1-bit pada posisi random. Percobaan dilakukan sepuluh kali dan hasil yang didapatkan menunjukkan bahwa algoritma 3RF dengan metode CBC memiliki *diffusion* yang tinggi karena perubahan 1-bit pada kunci dapat menghasilkan dua cipherteks yang rata-rata memiliki perbedaan sebesar 49.26 persen.

Tabel 3. Hasil pengujian *diffusion*.

no. uji	Jumlah Kesamaan *Bytes* pada Cipherteks	Persentase Perubahan
1	34334 / 67520	49,15
2	34523 / 67520	48,87
3	34283 / 65720	49,22
4	34310 / 67520	49,18
5	34350 / 65720	49,13
6	34235 / 65720	49,30
7	33948 / 65720	49,72
8	34115 / 65720	49,47
9	34094 / 65720	49,50
10	34377 / 65720	49,09
Rata-rata		49.26

Setelah itu, eksperimen dilakukan dengan metode ECB dengan menggunakan plainteks berukuran 1 blok, diketahui bahwa perubahan 1 bit pada separuh awal *plainteks* akan mengubah setengah blok dari cipherteks. Hal ini juga berlaku untuk perubahan 1 bit pada bagian paruh akhir plainteks. Dan jika bit paruh awal dan akhir diganti satu bit, maka akan mempengaruhi 1 blok cipherteks.

V. Analisis Keamanan

A. Serangan Brute Force

Serangan brute force adalah serangan yang dilakukan dengan mencoba satu per satu kemungkinan kunci yang digunakan untuk mengenkripsi suatu pesan. Sehingga waktu proses nya bergantung dengan banyak kemungkinan kunci yang dapat digunakan.

Kunci yang dimasukkan pengguna di-*hash* terlebih dahulu menggunakan algoritma SHA-256, sehingga panjang kunci yang digunakan dalam algoritma 3RF adalah 32 karakter atau 256 bit. Oleh karena itu, terdapat 256^{32} atau kira-kira setara dengan $115 * 10^{74}$ kemungkinan kunci. Apabila komputer dapat menguji 10^9 kunci dalam 1 detik, maka waktu yang dibutuhkan untuk menemukan satu kunci dalam kasus terburuk adalah $3.67x10^{60}$ tahun. Untuk kasus CBC, *initial vector* menggunakan hasil dari SHA-256 *round key*, sehingga tidak memperbanyak kompleksitas dalam algoritma brute force.

Algoritma 3RF dinilai cukup aman dari serangan bruteforce karena untuk menemukan sebuah kunci membutuhkan waktu sangat lama dan tidak mungkin dilakukan dengan satu komputer saja. Selain itu, apabila sistem terdistribusi dipakai untuk mencari kuncinya, kemungkinan biaya yang dibutuhkan untuk membangun sistemnya juga tidak sebanding dengan informasinya.

B. Serangan Analisis Frekuensi

Analisis frekuensi merupakan teknik memecahkan enkripsi ciphertext dengan memperhatikan frekuensi kemunculan huruf, bigram, atau kelompok huruf yang sering muncul pada plaintext, lalu menarik korelasi pemetaan plaintext ke ciphertext berdasarkan frekuensi tersebut. Sebagai contoh, pada bahasa Inggris, huruf yang sering muncul adalah E, T, A, dan O, sedangkan huruf yang jarang muncul adalah Z, Q, dan X.

Untuk ciphertext biner, frekuensi yang dianalisis merupakan frekuensi setiap byte. Sebagai contoh pada gambar 12 memiliki kemunculan byte 00 dua kali lebih banyak dari frekuensi byte yang lain. Info mengenai frekuensi kemunculan ini bisa dimanfaatkan oleh penyerang untuk membantu proses penemuan kunci seperti pada kasus pemecahan algoritma *vigenere.*

Algoritma 3RF dinilai sudah baik dalam menangani analisis frekuensi, hal ini ditunjukan dengan hasil persebaran frekuensi pada gambar 14 sampai gambar 18 yang menunjukan hasil distribusi yang lebih merata dibandingkan dengan distribusi dari plainteks awal (gambar 13).

C. Sifat Confusion dan Diffusion

Confusion menurut Ruadan ditunjukan dengan tidak ada kaitan antara plaintext dan cipherteks sehingga tidak dapat diserang dengan *frequency analysis*. Sedangkan *Diffusion* menurut Ruadan dapat ditunjukan dengan adanya perubahan signifikan pada hasil dekripsi cipherteks apabila cipherteks diubah sedikit.

Berdasarkan eksperimen pertama, dapat ditarik kesimpulan bahwa tidak ada kaitan antara cipher dan plain. Distribusi kemunculan byte setiap ciper dari berbagai metode berbeda dengan distribusi kemunculan byte pada plain file. Eksperimen kedua menunjukkan bahwa algoritma 3RF memiliki *confusion yang tinggi* dan *diffusion* yang cukup. Berdasarkan hasil tersebut, dapat algoritma 3RF dapat dikatakan cukup susah untuk dipecahkan secara statistik.

VI. Kesimpulan

Algoritma 3RF sudah dapat melakukan enkripsi dengan cukup baik. Hasil analisis statistik pada cipherteks menunjukkan bahwa algoritma dapat menghasilkan cipherteks

yang lebih merata relatif terhadap plainteks. Selain itu, pengujian lebih lanjut menunjukkan bahwa tingkat *confusion* dan *diffusion* dari algoritma cukup tinggi. Dengan mengubah 1-bit pada kunci, algoritma secara konsisten menghasilkan cipherteks dengan persentase perbedaan sebesar 99,61 persen. Kemudian, dengan mengubah 1-bit plainteks, algoritma secara konsisten menghasilkan cipherteks dengan persentase perbedaan sebesar 49,26 persen.

Algoritma 3RF memberikan hasil yang menjanjikan. Namun ada beberapa hal yang dapat dilakukan untuk memperbaiki algoritma ini:

1) Mengganti algoritma SHA-256 dengan algoritma *hash* lain yang lebih ringan untuk mengurangi waktu proses enkripsi dan dekripsi.
2) Menambah kompleksitas algoritma dengan menambah operasi pada tiap subfungsi atau menambah permutasi/kombinasi pemanggilan subfungsi yang mungkin.
3) Menambah mekanisme *feedback* pada kunci atau plainteks di dalam algoritma untuk meningkatkan *confusion* dan *diffusion*.
4) Menguji algoritma menggunakan data yang jauh lebih besar untuk menguji skalabilitasnya.

Referensi

[1] B. Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd ed., Indianapolis, IN: Wiley, 2015.

[2] A. J. Menezes, P. C. van Oorschot , and S. A. Vanstone, Handbook of Applied Cryptography. Boca Raton: CRC Press, 2001.

[3] R. Munir, Diktat Kuliah IF5054 Kriptografi, Bandung: Departemen Teknik Informatika Institut Teknologi Bandung, 2005.

[4] R. Munir, Slide Kuliah IF4020 Kriptografi: Advanced Encryption Standard (AES),, 2018.

[5] R. Munir, Slide Kuliah IF4020 Kriptografi: Kriptografi Modern, 2019.

[6] R. Anderson dan E. B Iham , "Two practical and provably secure block ciphers: BEAR and LION", D. Gollmann, editor, Fast Software Encryption, Third International Workshop (LNCS 1039), 113–120, Springer-Verlag, 1996.

[7] R. Hartanto dan D. Novanto, "Algoritma Block Cipher NH2", 2018.

PERNYATAAN

Dengan ini kami menyatakan bahwa makalah yang kami tulis ini adalah tulisan kami sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 13 Maret 2019

M. Ferdi Ghozali
13515014

M. Umar Fariz T.
13515050