Block Cipher Menggunakan Permutasi Diagonal dan Feistel Berbasiskan AES-128

Yusuf Rahmatullah
Program Studi Teknik Informatika
Institut Teknologi Bandung
Bandung, Indonesia
13512040@std.stei.itb.ac.id

Khaidzir Muhammad Shahih
Program Studi Teknik Informatika
Institut Teknologi Bandung
Bandung, Indonesia
13512068@std.stei.itb.ac.id

***Abstract*—**Paper ini berisi penjelasan mengenai algoritma enkripsi-dekripsi block cipher yang kami ajukan. Algoritma ini menggunakan blok sebesar 128-bit. Algoritma ini menggunakan feistel dengan S-Box berukuran 16x16 yang kami definisikan sendiri. Algoritma ini menggunakan permutasi diagonal. Algoritma yang kami ajukan merupakan algoritma yang berbasiskan kepada AES-128 dengan penambbahan algoritma feistel dan permutasi diagonal.

***Keywords*—**block cipher, Permutasi diagonal, Feistel, AES-128

I. PENDAHULUAN

Penggunaan kriptografi di masa ini sudah sangat banyak. Penggunaan kriptografi adalah untuk meningkatkan keamanan penyampaian pesan dari satu instansi ke instansi lain. Kriptografi mengikuti perkembangan zaman. Kriptografi modern adalah kriptografi digital yaitu menggunakan bit atau byta dalam enkripsi dan dekripsinya.

Algoritma kriptografi modern berkembang pesat. Algoritma kriptografi yang menjadi standar yang digunakan di dunia saat ini adalah AES(*Advenced Encryption Standard*). Standar ini yang dijadikan sebagai basis protokol yang digunakan untuk riset dalam bidang kriptografi.

AES adalah standar baru setelah DES (*Data Encryption Standard*) dijadikan standar dalam kriptografi. Dalam DES terdapat algoritma feistel yang digunakan untuk mengimplementasikan *confusion* dan *diffusion* pada algoritma enkripsinya. Namun, pada AES tidak lagi terdapat algoritma feistel. Oleh karena itu, kami mengusulkan algoritma enkripsi menggunakan algoritma feistel dengan berbasis pada AES-128.

II. DASAR TEORI

A. Kriptografi Modern

Kriptografi modern adalah kriptografi yang beroperasi dalam mode bit, tidak seperti kriptografi klasik yang beroperasi dalam mode karakter. Biasanya kriptografi modern menggunakan operasi xor dalam melakukan enkripsi maupun dekripsi. Plainteks, kunci, dan cipherteks diproses dalam mode bit (ataupun byte).

Perbandingan kriptografi modern dengan kriptografi klasik adalah keamanan yang ditingkatkan . Untuk satu karakter kunci pada kriptografi klasik, pemecahan kunci membutuhkan 52 kemungkinan [A-Z, a-z] sedangkan pada kriptografi modern membutuhkan kemungkinan sebanyak 2^8 (256) karena dalam satu karakter (karakter ASCII) terdapat 8 bit yang terdiri dari angka 0 atau 1. Hal ini akan berdampak lebih besar ketika panjang kunci ditingkatkan. Pada kunci sepanjang 56-bit, dibutuhkan 2^{56} atau 72 quadrilion kemungkinan kunci.

Terdapat dua kategori algoritma berbasis bit, yaitu *stream cipher* dan *block cipher*. *Stream cipher* atau cipher aliran adalah kriptografi modern yang beroperasi pada bit tunggal sedangkan *block cipher* atau cipher blok adalah kriptografi modern yang beroperasi pada blok bit.

Kedua kategori tersebut termasuk ke dalam kriptografi dengan kunci simetri, yaitu kriptografi dengan kunci yang digunakan dalam enkripsi sama dengan kunci yang digunakan dalam deskripsi. Sedangkan kunci asimetri adalah kunci yang digunakan dalam enkripsi berbeda dengan kunci yang digunakan dalam dekripsi. Biasanya kunci asimetri ini disebut kunci publik (*public key*) dan kunci privasi (*private key*).

B. Stream Cipher

Stream cipher atau cipher aliran adalah salah satu kriptografi modern dengan kunci simetri yang beroperasi dalam mode bit. Stream cipher mengkombinasikan plainteks dengan pseudorandom aliran digit.

Stream cipher merupakan kriptografi modern yang mengambil pendekatan dari kriptografi klasik yang tidak mungkin untuk dipecahkan, yaitu *one-time pad* (OTP). *Stream cipher* yang terinspirasi dari OTP lebih dikenal dengan sebutan Vernam Cipher.

Stream cipher banyak digunakan dalam tingkat perangkat keras dan penggunaan yang membutuhkan enkripsi dengan panjang plainteks tidak diketahui. Salah satu contoh penggunaan pada tingkat perangkat keras adalah pada keamaan koneksi nirkabel. Keuntungan penggunaan cipher aliran adalah besarnya data yang dinamik. Kerugian dalam penggunaan cipher blok untuk koneksi nirkabel adalah besarnya blok. Misalkan terdapat pesan sepanjang 32-bit pada cipher blok 128-bit, maka akan terdapat 96-bit bit

yang tidak terpakai dan digunakan sebagai *padding bit* untuk melengkapi bit yang kosong pada plainteks.

Kunci pada cipher aliran disebut dengan *keystream* atau aliran kunci. Aliran kunci ini dibangun oleh *keystream generator* yang berdasarkan kepada kunci yang digunakan oleh pengguna.

Kunci yang digunakan oleh pengirim dan penerima haruslah sama. Ini yang menyebabkan cipher aliran termasuk ke dalam kriptografi dengan kunci simteris. Implementasi *keystream generator* juga harus sama antara pengirim dan penerima sehingga aliran kunci yang dihasilkan sama.

Keystream generator yang sering digunakan adalah *Linear Feedback Shift register* (LFSR). Generator aliran kunci ini mengandung dua bagian yaitu bagian register geser sebanyak n bit dan fungsi umpan balik. Bagian register adalah bagian yang digunakan untuk menentukan nilai keluaran kunci aliran. Sedangkan bagian fungsi umpan balik adalah fungsi yang menggunakan register geser sebagai parameter masukan untuk menghasilkan aliran kunci.

C. *Block Cipher*

Block cipher adalah salah satu kategori dalam kriptografi modern menggunakan kunci simteris yang beroperasi pada blok data. Ukuran blok data merupakan kelipatan 8 bit (1 byte). Cipher blok banyak digunakan sebagai protokol kriptografi saat ini. Standar di dunia kriptografi menggunakan cipher jenis ini.

Algoritma cipher blok banyak mengimplementasikan cipher blok iteratif yaitu cipher blok yang mentransformasikan ukuran blok yang tetap dari plainteks menjadi cipherteks dengan ukuran blok yang sama. Cipher blok menggunakan *round function* untuk melakukan iterasi sebanyak n kali.

Mode operasi cipher blok dibagi ke dalam 4, yaitu *Electronic Code Book* (ECB), *Cipher Block Chaining* (CBC), *Cipher Feedback* (CFB), *Output Feedback* (OFB).

Electronic Code Book (ECB) melakukan proses enkripsi blok plainteks menjadi blok cipherteks secara individual dan independen. Hal ini menyebabkan blok ke-i pada plainteks tepat dienkripsi menjadi blok ke-i pada cipherteks. Kondisi tersebut menyebabkan blok yang sama pada plainteks selalu menghasilkan hasil yang sama pada cipherteks sehingga dapat dibuat *code book* untuk memetakan korespondensi blok plainteks dengan blok cipherteks.

ECB memiliki keunggulan pemrosesan yang dilakukan tidak secara linear karena setiap blok bersifat independen. Hal ini menjadikan ECB digunakan dalam enkripsi/dekripsi pada basis data karena basis data membutuhkan enkripsi/dekripsi pada sembarang *record* yang diakses.

Sedangkan kelemahan dari ECB adalah perulangan hasil cipherteks yang menyebabkan kriptanalis memudahkan menyerang cipherteks dengan menggunakan metode penyerangan statistik.

Cipher Block Chaining (CBC) adalah cipher blok yang memanfaatkan ketergantungan antar blok. Setiap blok pada plainteks bergantung kepada plainteks pada blok sebelumnya. Blok pertama membutuhkan blok semu yang disebut *Initialization Vector* (IV). IV dapat diberikan oleh pengguna ataupun disediakan oleh program.

Keuntungan mode CBC adalah tingkat keamanan yang lebih tinggi dari ECB karena setiap blok bergantung kepada blok sebelumnya. Hal ini menyebabkan kriptanalis menjadi sulit dalam melakukan kriptanalisis terhadap cipherteks.

Sedangkan kelemahan dari CBC adalah kerusakan pada salah satu bit menyebabkan kerusakan pada blok-blok selanjutnya. Kondisi ini tidak baik digunakan pada transportasi data di internet yang dapat mengalami *error bit.*

Cipher Feedback (CFB) adalah cipher blok yang mengatasi kelemahan CBC dalam komunikasi data. Pada CBC, data yang dienkripsikan dalam unit lebih kecil daripada ukuran blok. Unit yang dienkripsikan dapat berupa bit per bit seperti pada cipher aliran.

Kesalahan 1 bit pada blok cipherteks dengan mode CBC tidak mengakibatkan kerusakan pada hasil dekripsi. CFB melengkapi kekurangan ini. Dekripsi mode CFB mengalami hal yang sama. Kerusakan pada salah satu blok cipherteks akan merambat kepada blok-blok setelahnya. Hal ini meningkatkan keamanan agar kriptanalisis menjadi semakin sulit.

Output Feedback (OFB) mirip dengan mode CFB. Perbedaannya adalah pada proses enkripsi dimana n-bit hasil enkripsi disalin menjadi elemen proses paling kanan di antrian. Proses dekripsi dilakukan secara berkebalikan dari proses enkripsi.

Keuntungan mode OFB adalah perambatan kerusakan 1 bit pada suatu blok hanya berpengaruh terhadap blok yang berkoresponden. Kondisi ini cocok untuk transmisi analog yang didigitalisasi. Kondisi ini juga cocok untuk transmisi data pada tingkat perangkat keras.

Claude Shannon memperkenalkan prinsip *confusion* dan *diffusion* pada algoritma enkripsi/dekripsi untuk menyulitkan kriptanalis yang melakukan serangan statistik. Selain kedua hal yang diperkenalkan oleh Claude Shannon tersebut, terdapat algoritma feistel dan S-Box (kotak substitusi) yang menjadi prinsip-prinsip perancangan sebuah algoritma blok cipher.

D. *Jaringan Feistel*

Jaringan feistel adalah struktur simetris yang digunakan dalam mengkontruksi blok cipher. Jaringan feistel

memiliki keuntungan dalam proses enkripsi/dekripsi karena operasi yang diterapkan dalam enkripsi dan dekripsi adalah sama. Dalam jaringan feistel terdapat fungsi yang membutuhkan kunci dalam proses enkripsi/dekripsi-nya. Kunci ini dihasilkan dari *key generator* yang dijadwalkan menggunakan *key schedule.* Jaringan feistel adalah cipherblok yang dilakukan berkali-kali secara iteratif.

Fungsi pada jaringan feistel bersifat *reversible*, artinya algoritma pada fungsi tersebut tidak perlu diubah dalam melakukan enkripsi dan dekripsi. Hal ini menguntungkan pembuat fungsi untuk membuat fungsi serumit mungkin.

E. DES (Data Encrypton Standard)

DES adalah standar dalam kriptografi yang menggunakan kunci simteris sebesar 56-bit. DES menggunakan jaringan feistel dalam proses enkripsi/dekripsinya. DES menggunakan algoritma yang kompleks untuk melakukan enkripsi/dekripsi. Ukuran blok dari DES adalah 64-bit dengan bit yang terpakai adalah sebanyak 56-bit. 8-bit sisanya digunakan sebagai *parity* untuk melakukan pengujian kesalahan.

Fungsi pada jaringan feistel yang digunakan oleh DES menggunakan 4 tahap, yaitu ekspansi, acak kunci, substitusi, dan permutasi. S-Box yang digunakan untuk substitusi pada DES adalah S-Box berukuran 3 baris x 16 kolom. S-Box ini digunakan untuk melakuakn substitusi 8-bit menjadi 6-bit.

DES tidak lagi dijadikan standar karena kriptanalis berhasil menjebol keamanan DES. Pencarian kunci dilakukan secara *brute force* dengan menggunakan banyak yang diintegrasikan. Oleh karena itu, dibuatlah standar baru yang lebih kuat yang dinamakan *Advanced Encryption Standard* (AES).

F. AES (Advanced Encryption Standard)

AES adalah algoritma enkripsi/dekripsi blok cipher dengan menggunakan kunci dengan panjang kunci 128 bit sampai 256 bit dengan step 32 bit. Panjang kunci dan ukuran blok dapat dipilih secaar inpedenden. Penggunaan panjang kunci yang sering digunakan adalah 128-bit dan 256-bit. Dengan menggunakan kunci sepanjang 128-bit, maka akan terdapat sebanyak 2^{128} atau 3,4 x 10^{38} kemungkinan kunci.dengan kunci sepanjang 128-bit, dibutuhkan 5,4 x 10^{18} prosesor dengan percobaan 1 juta kunci setiap milidetik untuk mencoba seluruh kunci dalam satu tahun.

AES menggunakan algoritma Rjindael yaitu algoritma yang beroperasi dalam orientasi byte, tidak seperti DES yang beroperasi dalam orientasi bit. Seperti pada DES, AES memiliki kunci internal yang berbeda di setiap iterasinya yang disebut dengan *round key*. Kunci ini dibangkitkan dengan menggunakan *key schedule*.

Algoritma Rjindael dengan kunci 128-bit adalah sebagai berikut :

- *Initial round* - AddRoundKey: Melakukan xor terhadap plainteks dan key
- *N-round* : melakukan putaran sebanyak N kali dengan melakukan 4 hal berikut :
    - SubBytes : Substitusi menggunakan S-Box berukuran 16x16
    - ShiftRow : pergeseran baris-baris *array state* secara *wrapping*
    - MixColumns : mengacak data di masing-masing kolom *array state*
    - AddRoundKey : melakukan xor antara *state* dengan *round key*
- *Final round* : melakukan SubBytes, ShiftRow, dan AddRoundKey.

Banyaknya iterasi pada *N-round* bergantung kepada panjang kunci. Kunci dengan panjang 128-bit memiliki iterasi sebanyak 10 kali. Sedangkan kunci dengan panjang 256-bit memiliki iterasi sebanyak 14 kali.

SubBytes adalah substitusi dengan menggunakan S-Box. S-Box dirancang agar $a_{i,j}$ tidak sama dengan $S(a_{i,j})$ dan hasil xor keduanya tidak sama dengan 0xFF. Pembangunan S-Box ini dilakukan menggunakan invers multiplikatif.

ShiftRow adalah pergeseran baris-baris pada *array state* sebanyak i kali ke kiri. Baris pertama (i=0) tidak digeser. Baris kedua (i=1) digeser sebanyak 1 kali. Begitu juga seterusnya hingga baris keempat (i=3).

MixColumns adalah pengkombinasian setiap kolom pada *array state* dengan melakukan perkalian terhadap sebuah matriks. Matriks tersebut adalah sebagai berikut :

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

Angka-angka pada matriks di atas memiliki fungsi masing-masing. Perkalian dengan angka 1 tidak mengubah apapun. Perkalian dengan angka 2 memiliki arti bergeseran ke kiri 1 bit. Dan angka 3 memiliki arti pergeseran ke kiri satu kali dan operasi xor dengan angka sebelumnya.

AddRoundKey adalah operasi xor terhadap *array state* dengan kunci yang bersangkutan. Pada *initial round*, operasi xor dilakukan menggunakan kunci masukan pengguna. Sedangkan pada *N-Round* digunakan kunci internal yang dibangun dari *Key Schedule*.

III. RANCANGAN BLOCK CIPHER

Rancangan cipher blok yang diajukan oleh kami adalah berdasarkan kepada algoritma Rjindael pada AES. Namun

kami menambahkan jaringan feistel pada DES dan melakukan permutasi dengan menggunakan algoritma permutasi diagonal. Algoritma ini menggunakan kunci 128-bit dengan ukuran blok 4x4 bytes. Tahap-tahap enkripsi adalah sebagai berikut :

- Pemrosesan plainteks, substitusi menggunakan S-Box buatan kami dan transposisi secara diagonal
- Pembangkitan kunci internal, kunci internal dibangkitkan sebanyak 16 buah untuk 16 ronde proses iterasi. Pembangkitan kunci melibatkan proses permutasi diagonal, pergeseran baris-baris secara *wrapping*, dan substitusi menggunakan S-Box.
- Proses iterasi, ietrasi dilakukan sebanyak 16 kali. Setiap proses iterasi mengandung jaringan feistel. setiap iterasi dilakukan proses xor dan pergeseran secara *iterate-based wrapping*

Pemrosesan plainteks tahap awal dilakukan dengan metode substitusi dengan S-Box berukuran 16x16 dan transposisi diagonal. Kotak S-Box yang kami gunakan ditunjukan oleh tabel 1.

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	A2	E9	E0	17	43	A8	39	0F	01	3E	55	3C	D4	46	3F	23
	1	0D	09	5A	25	14	76	70	10	80	4B	F1	6B	3B	8C	16	63
	2	A0	93	75	ED	89	6C	99	35	1A	34	78	4C	F9	8E	2E	4A
	3	2A	51	FE	38	AF	06	24	D3	4E	C9	41	18	69	BB	7F	87
	4	F2	4F	B1	0A	E7	9B	7D	BD	48	D9	E8	6E	65	21	88	7A
	5	CA	82	DC	C7	C4	8D	A5	8F	56	C1	F5	A6	B8	6A	07	97
	6	0B	9F	58	73	A7	54	D1	1D	2C	CF	0E	05	A4	7B	B7	36
	7	19	B2	81	C3	31	15	AB	8B	E6	EA	D6	5D	44	1E	08	98
	8	EB	C8	FB	74	22	53	5E	72	AD	71	45	C5	47	2B	61	C6
	9	E5	5F	57	28	64	95	BC	F8	67	66	EF	D5	FC	32	11	FF
	A	1F	A1	02	E3	83	AA	79	30	B5	04	7E	F7	5B	2F	E1	1C
	B	BA	BF	00	85	8A	CC	2D	B6	FA	12	03	E2	DA	FD	33	60
	C	B9	D8	37	AC	42	91	40	50	94	26	0C	CB	3A	DF	EE	29
	D	96	F6	F3	D2	20	C2	DD	49	CE	13	68	9E	D0	4D	3D	BE
	E	92	AE	DB	F4	D7	DE	B4	EC	27	9C	52	E4	84	62	6F	9D
	F	7C	86	90	C0	B3	59	B0	5C	A3	6D	CD	F0	A9	9A	77	1B

Tabel 1. S-Box yang digunakan untuk substitusi

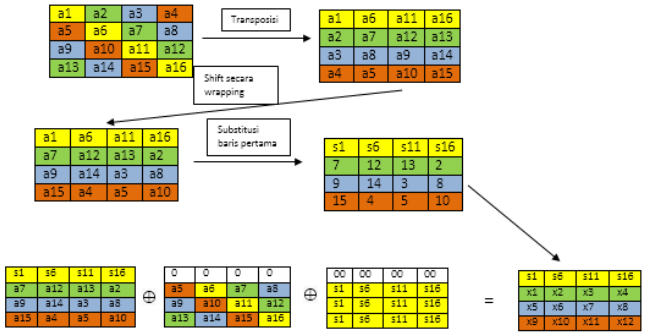
Transposisi diagonal adalah transposisi pada suatu matriks dengan mengubah letak elemen matriks dengan perubahan arah diagonal. Transposisi secara diagonal tersebut dapat dilihat pada bagan dibawah ini :

A1	A2	A3	A4
A5	A6	A7	A8
A9	A10	A11	A12
A13	A14	A15	A16

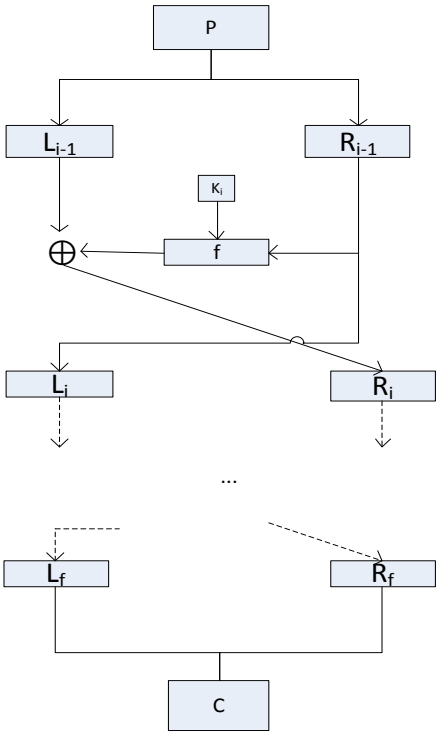
→

A4	A7	A10	A13
A3	A6	A9	A16
A2	A5	A12	A15
A1	A8	A11	A14

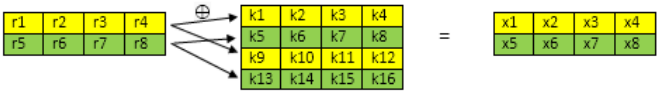
Tahap kedua adalah tahap pembangkitan kunci internal. Dalam algoritma rancangan kami ini akan dibangkitkan kunci internal sebanyak 16 buah. Kunci eksternal dari pengguna akan dioperasikan dan diubah menjadi kunci internal yang digunakan untuk mengenkripsi blok – blok dari plaintext. Proses operasi dari pembangkitan kunci internal tersebut sebagai berikut. Kunci diinterpretasikan menjadi sebuah matriks berukuran 4x4 byte. Pertama matriks input akan dilakukan operasi transpos diagonal seperti pada plaintext diatas, hanya saja dengan arah yang berlawanan (diagonal dari atas kiri ke kanan bawah dan maju ke arah kanan). Selanjutnya dilakukan operasi *shift wrapping* baris per baris ke arah kiri. Baris pertama tidak digeser, baris kedua digeser sejauh 1 byte, baris ketiga sejauh 2 byte, dan baris keempat sejauh 3 byte. Setelah itu baris pertama matriks hasil *shifting* disubstitusikan menggunakan kotak substitusi. Proses yang terakhir adalah operasi xor antara baris pertama, baris ke-i matriks saat ini, dan baris ke-i matriks yang lama (matriks input) dimulai dari baris kedua hingga baris keempat. Skema proses-proses diatas digambarkan sebagai berikut.



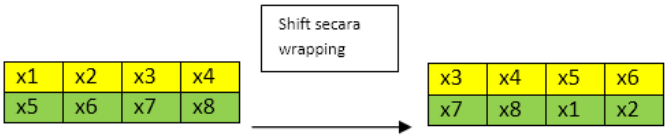
Tahap ketiga adalah tahap iterasi. Proses iterasi dalam algoritma ini menggunakan jaringan Feistel sebanyak 16 kali. Matriks plainteks hasil proses 1 dibagi menjadi 2 sehingga berukuran 4x2 byte dengan matriks L merupakan baris pertama dan kedua, dan matriks R adalah baris ketiga dan keempat. Selanjutnya matriks L dan R diproses menggunakan jaringan Feistel.



Fungsi f pada iterasi ke i (1..16) pada jaringan Feistel diatas terdiri dari 2 proses. Proses pertama yaitu operasi xor antara baris ganjil R_{i-1} dengan baris ganjil K_i dan antara baris genap R_{i-1} dengan baris genap K_i. Selanjutnya dilakukan operasi shifting secara wrapping ke kiri sebanyak (i-1) mod 8 byte ke kiri.



Baris yang kuning dixorkan dengan yang berwaena kuning, baris hijau di-xor-kan dengan hijau lainnya.



Shift secara wrapping dengan sejauh 2 byte ((i-1)mod 8 = 2).

Tahap terakhir dari proses enkripsi adalah hasil dari jaringan feistel dilakukan transposisi diagonal seperti pada tahap pertama kemudian dilanjutkan substitusi menggunakan kotak substitusi.

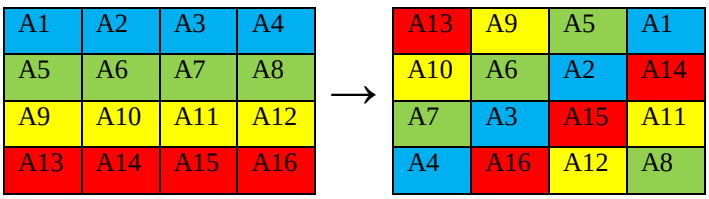
Proses dekripsi merupakan kebalikan dari proses enkripsi. Proses dekripsi adalah sebagai berikut :

- Pemrosesan cipherteks, substitusi invers menggunakan S-Box dan melakukan invers transposisi diagonal.
- Pembangkitan kunci internal, tahap ini sama dengan pembangkitan kunci internal pada proses enkripsi.
- Proses iterasi invers, proses ini merupakan kebalikan dari proses iterasi pada enkripsi dengan fungsi feistel yang sama dengan fungsi feistel pada enkripsi
- Proses akhir, melakukan invers transposisi diagonal dan melakukan invers substitusi.

Tahap pertama adalah pemrosesan awal menggunakan invers substitusi, yaitu kebalikan dari substitusi menggunakan S-Box, invers substitusi menggunakan S-Box Invers untuk mengembalikan hasil substitusi S-Box. S-Box Invers adalah sebagai berikut :

		Y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	B2	08	A2	BA	A9	6B	35	5E	7E	11	43	60	CA	10	6A	07
	1	17	9E	B9	D9	14	75	1E	03	3B	70	28	FF	AF	67	7D	A0
	2	D4	4D	84	0F	36	13	C9	E8	93	CF	30	8D	68	B6	2E	AD
	3	A7	74	9D	BE	29	27	6F	C2	33	06	CC	1C	0B	DE	09	0E
	4	C6	3A	C4	04	7C	8A	0D	8C	48	D7	2F	19	2B	DD	38	41
	5	C7	31	EA	85	65	0A	58	92	62	F5	12	AC	F7	7B	86	91
	6	BF	8E	ED	1F	94	4C	99	98	DA	3C	5D	1B	25	F9	4B	EE
	7	16	89	87	63	83	22	15	FE	2A	A6	4F	6D	F0	46	AA	3E
	8	18	72	51	A4	EC	B3	F1	3F	4E	24	B4	77	1D	55	2D	57
	9	F2	C5	E0	21	C8	95	D0	5F	7F	26	FD	45	E9	EF	DB	61
	A	20	A1	00	F8	6C	56	5B	64	05	FC	A5	76	C3	88	E1	34
	B	F6	42	71	F4	E6	A8	B7	6E	5C	C0	B0	3D	96	47	DF	B1
	C	F3	59	D5	73	54	8B	8F	53	81	39	50	CB	B5	FA	D8	69
	D	DC	66	D3	37	0C	9B	7A	E4	C1	49	BC	E2	52	D6	E5	CD
	E	02	AE	BB	A3	EB	90	78	44	4A	01	79	80	E7	23	CE	9A
	F	FB	1A	40	D2	E3	5A	D1	AB	97	2C	B8	82	9C	BD	32	9F

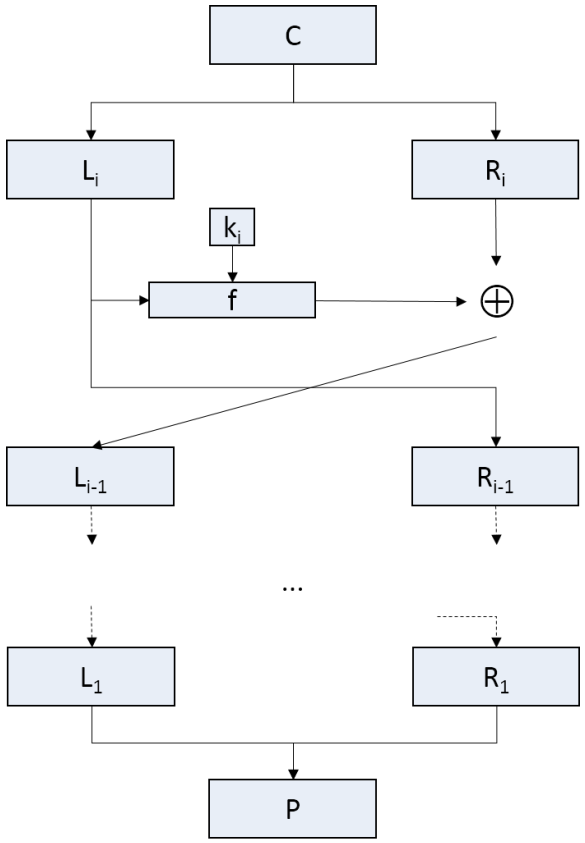
Langkah selanjutnya adalah melakukan invers transposisi diagonal. Bagan di bawah ini menunjukan bagaimana invers transposisi diagonal bekerja :



Tahap selanjutnya adalah pembangkitan kunci interal. Pembangkitan kunci internal pada dekripsi sama dengan pembangkitan kunci internal pada proses enkripsi. Hal ini dilakukan agar kunci internal yang didapatkan sesuai dengan kunci internal yang didapatkan pada saat enkripsi.

Tahap selanjutnya adalah proses iterasi invers. Iterasi invers adalah kebalikan dari iterasi pada proses enkripsi. Oleh

karena itu tahap iterasi dimulai dari iterasi ke-16 hingga iterasi ke-1. Iterasi dilakukan dari cipherteks hingga menjadi plainteks. Proses iterasi tersebut digambarkan oleh bagan di bawah ini :



Tahap terakhir adalah melakukan kembali invers transposisi diagonal dan invers substitusi menggunakan S-Box Invers.

IV. EKSPERIMEN DAN PEMBAHASAN HASIL

Eksperimen dilakukan dengan menggunakan 4 mode algoritma cipher blok yaitu ECB, CBC, CFB, dan OFB serta perbandingannya menggunakan algoritma AES dan DES yang disediakan oleh java. Percobaan dilakukan dengan menggunakan gambar berukuran 512x512 di bawah ini

dengan key 128 bit yaitu :

```
6b 72 69 70 74 6f 67 72 61 66 69 41 45 53 2b 2b
```

Waktu masing-masing algoritma yang dibutuhkan untuk melakukan enkripsi adalah sebagai berikut :

algoritma	Waktu (ms)
AES	25
DES	63
ECB	486
CBC	269
CFB	230
OFB	211

Sedangkan waktu yang dibutuhkan untuk melakukan dekripsi adalah sebagai berikut :

algoritma	Waktu (ms)
AES	17
DES	22
ECB	338
CBC	210
CFB	199
OFB	370

Waktu yang dibutuhkan oleh algoritma yang kami ajukan untuk melakukan enkripsi/dekripsi gambar tersebut lebih lama dibandingkan dengan waktu yang dibutuhkan algoritma AES dan DES yang disediakan oleh java.

Percobaan selanjutnya adalah dengan menggunakan plainteks unifrom. Plainteks yang kami gunakan adalah hex 0x41 sebanyak 32 byte. Hasil enkripsi yang didapat dari masing-masing algoritma adalah sebagai berikut :

algoritma	Hasil enkripsi
AES	3d 39 8b 7b 3a 8f 80 e2 0c 46 cf ce ce 61 98 68 3d 39 8b 7b 3a 8f 80 e2 0c 46 cf ce ce 61 98 68 9c f1 bb f3 d8 3c 3b f7 d0 27 7f 73 d9 6d fd 1c
DES	22 1f 79 ae 4f e9 37 89 22 1f 79 ae 4f e9 37 89 22 1f 79 ae 4f e9 37 89 22 1f 79 ae 4f e9 37 89 e2 20 b3 70 1a 10 88 9d
ECB	d5 73 62 c3 a3 fc bd da d5 54 ee 38 9e 60 4f 03 d5 73 62 c3 a3 fc bd da d5 54 ee 38 9e 60 4f 03
CBC	d5 73 62 c3 a3 fc bd da d5 54 ee 38 9e 60 4f 03 fc d5 50 98 13 b8 a7 79 5a 16 ec 7c 22 37 3c 64
CFB	ea 20 23 82 37 f3 fc 9b ea ec af 79 65 9d 0e 42 4d 95 fa 70 3a 8a 8f 4e a1 ac 45 e5 e1 6d 60 3e
OFB	ea 20 23 82 37 f3 fc 9b ea ec af 79 65 9d 0e 42 8a 0d 95 7b 88 96 9f 2f 23 cf 04 94 28 ad fe 76

Percobaan selanjutnya adalah merusak byte pertama pada cipherteks di atas. Hasil dekripsi yang didapat dari masing-masing algoritma adalah sebagai berikut :

algoritma	Hasil enkripsi
AES	cf 67 e2 f1 91 93 00 36 72 a8 2c 64 e8 b1 69 fd 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
DES	49 08 9b 39 11 94 47 bd 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
ECB	41 eb 41 41 41 41 eb 41 41 eb 41 eb eb 41 eb 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
CBC	41 eb 41 41 41 41 eb 41 41 eb 41 eb eb 41 eb 41 be 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
CFB	be 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 8f 41 ba 41 41 41 a3 41 a3 41 ad 41 08 41 41
OFB	be 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41

Waktu yang dibutuhkan algoritma AES dan DES jauh lebih besar dari algoritma yang kami ajukan. Hal ini dikarenakan algoritma yang AES dan DES gunakan sudah mangkus. Sedangkan algoritma yang kami gunakan belum mangkus (10-20x lebih lambat).

V. Analisis Keamanan

Analisis kemanan dilakukan dengan menghitung kemungkinan banyak kunci pada algoritma kriptografi yang kami ajukan. Kunci yang kami gunakan adalah kunci dengan panjang 128-bit. Kemungkinan kunci adalah sebanyak 2^{128} atau 3,4 x 10^{38} kemungkinan. Jika dalam waktu satu milidetik dapat mencoba 1 juta kunci, maka dibutuhkan waktu 5,4 x 10^{18} tahun untuk mencoba seluruh kemungkinan kunci. Oleh karena itu pencarian kunci yang sesuai dengan menggunakan *brute force* tidak dapat dilakukan.

Analisis keamanan selanjutnya dihitung dari S-Box yang kami buat sendiri. S-Box pada algoritma Rjindael dirancang agar tidak terdapat $a_{i,j} = S(a_{i,j})$ dan untuk setiap $a_{i,j} \oplus S(a_{i,j}) \neq$ 0xFF. Analisis kami lakukan terhadap S-Box yang kami buat. S-Box yang kami buat memiliki 6 kondisi yang terdapat $a_{i,j} = S(a_{i,j})$ sedangkan untuk setiap $a_{i,j} \oplus S(a_{i,j}) \neq$ 0xFF. Hal ini mengurangi keamanan dari algoritma kami sebesar

$$\frac{6}{256} x100\% = 2{,}34\%$$

VI. Kesimpulan dan Saran

Algoritma yang kami ajukan memiliki waktu eksekusi jauh lebih besar dari algoritma AES dan DES. Algoritma enkripsi yang kami ajukan memiliki tingkat kemanan yang tinggi karena menggunakan kunci ssepanjang 128-bit sehingga membutuhkan waktu yang sangat lama untuk berhasil mencoba semua kemungkinan kunci. Kekurangan dalam kotak substitusi yang kami ajukan sebesar 2,34%.

Saran untuk algoritma yang kami ajukan adalah menciptakan S-Box yang memenuhi kondisi $a_{i,j} \neq S(a_{i,j})$ dan $a_{i,j} \oplus S(a_{i,j}) \neq$ 0xFF.

References

[1] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, Niels Ferguson, Tadayoshi Kohno, Mike Stay. "The Twofish Team's Final Comments on AES Selection". 2000.

[2] Christof paar, jan Pelzl. "Stream Ciphers". Springer, 2009.

[3] Matt J. B. Robshaw. "Stream Cipher Technical Report version 2". RSA Laboratories. 1995

[4] Munir Rinaldi. "Advanced Encryption Standard". 2015. Bandung, informatika ITB. Hlm 9-30

[5] Munir Rinaldi. "Algoritma Kriptografi Modern bag. 1". 2015. Bandung, Informatika ITB . Hlm 1-50

[6] Munir Rinaldi. "Algoritma Kriptografi Modern bag. 2". 2015. Bandung, Informatika ITB. Hlm 1-51

[7] Thomas Beth and Fred Piper. "The Stop-and-Go Generator". EUROCRYPT. 1984. Pp88-92.