GENOS : Algoritma Block Cipher

Harry Octavianus Purba
Teknik Informatika, STEI
Institut Teknologi Bandung
Bandung, Indonesia
13514050@std.stei.itb.ac.id

Rio Chandra Rajagukguk
Teknik Informatika, STEI
Institut Teknologi Bandung
Bandung, Indonesia
13514082@std.stei.itb.ac.id

***Abstract*—Algoritma GENOS mengimplementasikan algoritma blok chiper dengan tahap pembangkitan kunci internal, putaran awal, putaran feistel, putaran akhir, dan proses dekripsi. Hasil pengujian yang dilakukan menunjukkan perubahan satu bit pada plaintext menghasilkan perubahan yang signifikan pada chipertext yang dihasilkan. Sifat perubahan pesan yang signifikan tersebut menjadi kelebihan dan sulit untuk dilakukan kriptanalis. Jika dilakukan brute force maka akan membutuhkan waktu yang sangat lama.**

Keywords—chiper; blok; genos; confusion; feistel; difusion

I. PENDAHULUAN

Kriptografi (cryptography) adalah ilmu sekaligus seni untuk menjaga keamanan pesan. Pada Yunani Kuno, Kriptografi (cryptography) terdiri dari dua suku kata yaitu kripto dan graphia. Kripto artinya menyembunyikan, sedangkan graphia artinya tulisan. Secara umum, kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi, seperti kerahasiaan data, keabsahan data, integritas data, serta autentikasi data.

Tidak semua aspek keamanan informasi dapat diselesaikan dengan kriptografi. Kemanjuan teknologi seperti internet saat ini, menjadi salah satu topik penting yang harus diperhatikan dalam menjaga kerahasiaan informasi yang akan dikirim melalui sebuah jaringan. Informasi atau pesan yang ditransmisikan dari suatu tempat ke tempat lain melalui banyak jaringan sangat rentan terhadap penyadapan yang menimbulkan kerugian. Cara yang umum dipakai untuk mengamankan informasi tersebut adalah dengan menggunakan teknik kriptografi. Beberapa teknik yang dapat digunakan salah satunya adalah teknik Block Cipher.

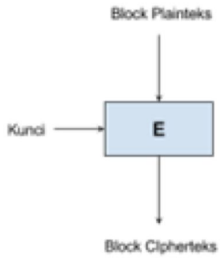
Block Cipher merupakan salah satu kriptografi untuk melakukan enkripsi dan dekripsi kepada sebuah pesan dengan membagi pesan ke dalam beberapa blok. Setiap blok mempengaruhi hasil enkripsi berikutnya sehingga jika blok yang dikirim sama, maka hasil enkripsi blok tersebut belum tentu sama.

Makalah ini membahas GENOS sebuah algoritma blok cipher kompleks. Algoritma ini memiliki tingkat konfusi dan difusi yang baik sehingga sangat sulit untuk dipecahkan.

II. LANDASAN TEORI

A. *Algoritma Blok Chiper*

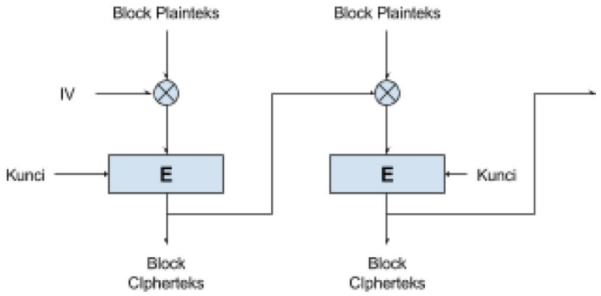
Algoritma blok cipher salah satu algoritma kriptografi yang membagi pesan ke dalam blok-blok pesan dengan ukuran tertentu. Keluaran berupa blok-blok pesan memudahkan pengiriman pada suatu jaringan. Blok cipher pada umumbya terbagi menjadi tiga jenis yaitu, Electronic Code Block (ECB), Cipher Block Chaining (CBC), Cipher-Feedback (CFB), dan Output-Feedback(OFB). Pada ECB, setiap blok dienkripsi dan dideskripsi secara independen dengan cara memasukkan blok ke dalam suatu fungsi, yang nantinya akan menjadi blok cipher. Pada ECB, hasil suatu enkripsi blok tidak akan mempengaruhi proses enkripsi blok yang lain. penggunaan metode ini kurang aman karena setiap blok yang sama akan menghasilkan blok cipher text yang sama pula. sehingga dapat dipecahkan dengan analisis kriptanalis.



Gambar 1. Electronic Code Block (ECB) [Algoritma Kripto Modern, oleh M, Rinaldi. 2015]

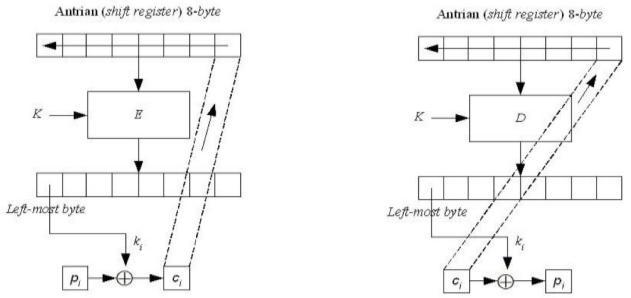
Pada metode CBC, setiap blok bergantung dengan blok yang lain dalam enkripsi dan dekripsinya. Enkripsi dan dekripsi pada metode ini membutuhkan sebuah blok baru yang disebut IV (Initialization Vector) yang akan digunakan pada XOR pertama

tahap dekripsi maupun enkripsi. Selanjutnya nilai yang di XOR-kan dengan blok berikutnya adalah blok sebelumnya.



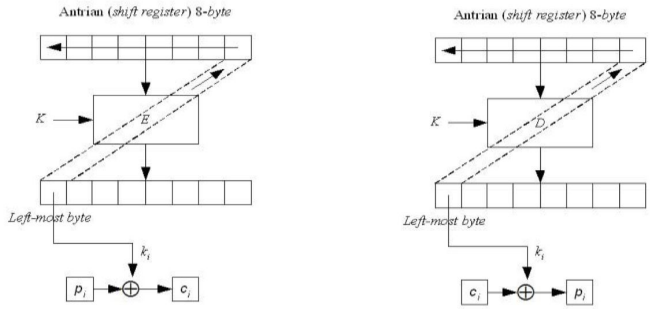
Gambar 2. Cipher Block Chaining(CBC) [Algoritma Kripto Modern, oleh M, Rinaldi. 2015]

Pada CFB, metode ini memperbaiki kelemahan pada metode CBC, seperti jika terjadi blok yang belum lengkap dan mengurangi kesalahan dekripsi data jika data rusak selama perjalanan. Metode ini dapat bekerja pada unit-unit blok yang cukup kecil sehingga dapat menyerupai stream-cipher.



Gambar 3. Cipher-Feedback (CFB) [Algoritma Kripto Modern, oleh M, Rinaldi. 2015]

Pada metode OFB, hanya sedikit berbeda dengan CFB. Perbedaan OFB dan CFB terdapat pada IV yang digunakan pada fungsi E kedua dan seterusnya berasal dari hasil fungsi E dengan masukan Kunci dan IV sebelumnya. Hal ini menyebabkan OFB bisa mengolah blok selanjutnya tanpa harus menunggu XOR selesai dilakukan pada blok sebelumnya.



Gambar 4. Cara kerja OFB [Algoritma Kripto Modern, oleh M, Rinaldi. 2015]

B. Jaringan Feistel

Hampir semua algoritma blok Chiper bekerja menggunakan model Jaringan Feistel. Ditemukan oleh Horst Feistel tahun 1970, model Jaringan Feistel digunakan dengan cara sebagai berikut :

1. Bagi blok yang panjangnya n-bit menjadi 2 bagian. Kiri (L) dan kanan (R), yang masing-masing panjangnya n/2 (dengan syarat n harus genap)
2. Definisikan cipher blok berulang dimana hasil dari putaran ke-I ditentukan dari hasil putaran sebelumnya, yaitu :

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

Penjelasan:
i = 1,2,….,r (r adalah jumlah putaran)
K_i : subkey pada putaran ke-i
f :fungsi transformasi (didalamnya terdapat fungsi substitusi, permutasi, dan/atau ekspansi, kompresi).

Plaintext adalah gabungan L dan R awal, atau secara formal dinyatakan dengan (L0, R0). Sedangkan ciphertext didapatkan dari L dan R hasil dari putaran terakhir setelah terlebih dahulu dipertukarkan, atau secara formal dinyatakan sebagai (Rr, Lr).

Jaringan Feistel banyak dipakai dalam algoritma kriptografi seperti DES (Data Encryption Standard), LOKI, GOST, FEAL, Lucifer, Blowfish, Khufu, Khafre, dan lainnya karena sifatnya reversible dapat digunakan sekaligus untuk proses enkripsi dan dekripsi. Operator XOR mengkombinasikan setengah bagian kiri dengan hasil dari fungsi transformasi f, sehingga dapat ditulis dengan persamaan berikut :

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

Sifat reversible tidak tergantung dengan fungsi f sehingga fungsi f memungkinkan dibuat sangat rumit.

III. RANCANGAN ALGORITMA

Algoritma blok cipher yang dibuat dinamakan dengan algoritma blok cipher GENOS. Algoritma ini merupakan modifikasi dari algoritma blok cipher dengan memanfaatkan putaran Feistel di dalamnya. Spesifikasi dari algoritma ini antara lain :

1. Ukuran bit untuk setiap blok adalah 64 bit
2. Panjang kunci eksternal adalah 256 bit
3. Banyak putaran Feistel yang dilakukan adalah 32 kali
4. Untuk setiap putaran memilki kunci internal yang dibangkitkan dari kunci eksternal

Algoritma ini memiliki tahapan permutasi awal, 32 putaran feistel, dan permutasi akhir.

A. Pembangkitan Kunci Internal

Kunci internal dibangkitkan dari 256 bit kunci eksternal. Setiap putaran Feistel akan menggunakan 32 bit kunci. Sehingga dari 256 bit kunci eksternal akan dibagi menjadi 8 kunci secara sekuensial masing-masing berukuran 32 bit yaitu dari K1 hingga K8. Kunci internal untuk setiap putaran Feistel kemudiankan dengan melihat pada Tabel 1.

K1	K3	K7	K4	K5	K8	K5	K4
K7	K5	K8	K2	K7	K1	K4	K6
K2	K6	K1	K3	K5	K3	K8	K2
K6	K2	K4	K1	K7	K6	K3	K8

Tabel 1. Tabel Permutasi Kunci

B. Permutasi Awal

Sesuai dengan spesifikasi, algoritma GENOS akan melakukan enkripsi terhadap blok plaintext berukuran 64 bit atau setara dengan 8 karakter. Tahapan awal yang dilakukan adalah tahap permutasi awal. Pada tahapan ini akan dilakukan permutasi pada 64 bit blok plaintext. Permutasi dilakukan dengan menggunakan aturan pada tabel permutasi awal.Tabel permutasi awal dapat dilihat pada Tabel 2.

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

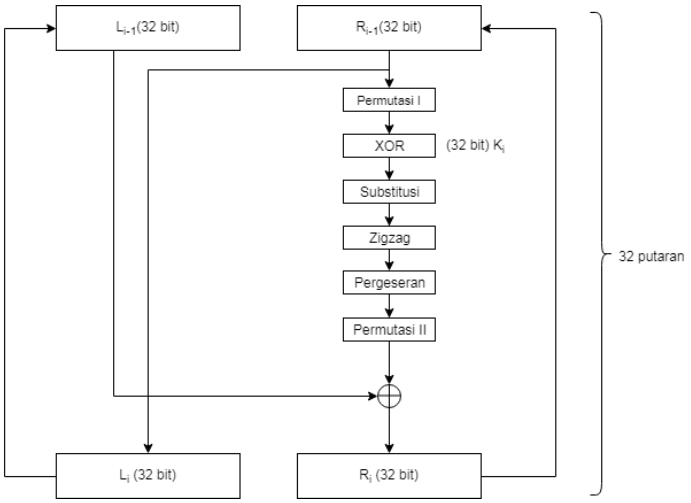
Tabel 2. Tabel Permutasi Awal

Dengan menggunakan tabel tersebut, isi bit ke 0 dari plaintext akan diganti dengan isi bit ke 58. Isi bit ke 1 akan diganti dengan isi bit ke 50, demikian selanjutnya hingga isi bit ke 63 akan diganti dengan isi bit ke 7.

C. Putaran Feistel

Setelah melalui tahapan permutasi awal, tahapan selanjutnya adalah putaran Feistel. Putaran Feistel akan dilakukan sebanyak 32 kali. Sebelum melakukan putaran, blok plaintext yang berukuran 64 bit akan terlebih dahulu dibagi menjadi blok kiri dan blok kanan yang masing-masing berukuran 32 bit. Blok kiri adalah blok plaintext dari indeks ke 0 sampai 31, blok kanan adalah blok plaintext dari indeks ke 32 sampai 63.Untuk setiap putaran, akan dilakukan pengubahan terhadap blok kanan dengan mengikuti serangkaian algoritma yaitu permutasi I, XOR, substitusi, zigzag dan rotasi, pergeseran bit, dan permutasi II , yang kemudian akan dilakukan proses XOR terhadap blok kiri. Sedangkan blok kiri akan diganti dengan blok kanan sebelum melalui proses permutasi I.

Skema putaran Feistel dapat dilihat pada Gambar 5.



Gambar 5. Skema Putaran Feistel

I. Permutasi 1

Tahapan pertama yang dilalui oleh blok kanan adalah tahapan permutasi I. Pada tahapan ini, akan dilakukan permutasi terhadap 32 bit blok tersebut dengan mengikuti tabel permutasi. Tabel permutasi yang digunakan dapat dilihat pada Tabel 3. Hasil permutasi akan membentuk blok kanan yang baru.

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

Tabel 3. Tabel Permutasi I

II. XOR

Setelah melalui proses permutasi , proses selanjutnya adalah proses XOR. Pada proses ini, kunci internal putaran terlebih dahulu akan digeser secara sirkuler sebesar 7 bit ke kanan. Blok kanan hasil permutasi I, akan di XOR kan dengan kunci internal yang telah digeser tersebut. Hasilnya kemudian akan menjadi blok kanan yang baru.

III. Substitusi dan Rotasi

Setelah melalui proses XOR, proses selanjutnya adalah proses substitusi. Pada proses ini setiap empat bit berurutan pada blok kanan akan diganti dengan 4 bit pada tabel substitusi. Tabel substitusi dapat dilihat pada Tabel 4.

	00	01	10	11
00	1000	1010	1011	0000
01	1001	0001	0111	1101
10	0101	0110	1111	0010
11	0100	1100	0011	1110

Tabel 4. Tabel Substitusi

Dengan menggunakan tabel substitusi ini, maka misalkan 4 bit yang akan diubah adalah '0111' , maka akan dilihat isi dari tabel untuk baris = 01 (2 bit pertama) dan kolom = 11 (2 bit terakhir), sehingga didapatkan '1101'. 4 bit itu juga kemudian akan dilakukan rotasi, dimana posisi 4 akan bertukar dengan posisi 1, dan posisi 2 akan bertukar dengan posisi 3, sehingga '1101' akan menjadi '1011'. Ketika seluruh pasangan 4 bit telah melakukan substitusi dan rotasi , hasilnya kemudian akan menjadi blok kanan yang baru.

IV. Zigzag

Setelah melalui proses substitusi, proses selanjutnya yang dilakukan adalah proses zigzag dan rotasi. Proses zigzag adalah proses mengubah susunan bit menjadi selang seling. Posisi genap akan diisi oleh 16 bit pertama, posisi ganjil akan diisi oleh 16 bit terakhir. Sehingga yang sebelumnya memiliki urutan 0,1,2,3,4,... 31 , akan menjadi 0,16,1,17,2,18,3,19,4, ... ,15,32. Hasil zigzag ini kemudian akan menjadi blok kanan yang baru.

V. Pergeseran

Setelah melalui proses zigzag, proses selanjutnya yang dilakukan adalah proses pergeseran. Pergeseran yang dilakukan adalah pergeseran blok kanan sebesar 10 bit ke kiri secara sirkuler sehingga untuk setiap penggeseran, posisi ke 1 akan bergeser menjadi posisi 0, posisi ke 0 akan bergeser menjadi posisi ke 31, posisi ke 31 akan bergeser menjadi posisi ke 30 dan seterusnya. Hasil pergeseran kemudian akan menjadi blok kanan yang baru

VI. Permutasi II

Setelah melalui proses pergeseran , langkah selanjutnya adalah proses permutasi II. Permutasi II sama seperti permutasi I namun pada tabel yang digunakan adalah tabel permutasi II yang dapat dilihat pada Tabel 5. Hasil permutasi kemudian menjadi blok kanan yang baru,

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

Tabel 5. Tabel Permutasi II

VII. XOR terhadap Blok Kiri

Setelah melalui proses permutasi II, blok kanan kemudian akan diXOR kan dengan blok kiri membentuk blok kanan final untuk menuju putaran berikutnya. Sedangkan blok kiri final adalah 32 bit blok kanan yang belum melalui proses permutasi I.

D. Putaran Akhir

Setelah melalui 32 buah putaran Feistel, blok kiri kemudian akan digabungkan kembali dengan blok kanan membentuk 64 bit. 32 bit pertama adalah blok kiri, dan 32 bit kedua adalah blok kanan. Sebelum mencapai tahap akhir menjadi ciphertext, 64 bit ini akan melakukan proses permutasi akhir. Permutasi akhir dilakukan sesuai dengan invers dari tabel permutasi awal. Tabel permutasi akhir dapat dilihat pada Tabel 6.

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

Tabel 6. Tabel Permutasi Akhir

E. Proses Dekripsi

Proses dekripsi dilakukan dengan membalik skema pada Gambar 5. Terlebih dahulu pada 64 bit blok ciphertext akan dipermutasikan dengan menggunakan tabel invers permutasi akhir. Artinya sama saja dengan menggunakan tabel permutasi awal (tabel permutasi awal dan tabel permutasi akhir memilki sifat invers). Kemudian 64 bit tersebut akan dibagi menjadi blok kiri dan blok kanan masing-masing sebesar 32 bit. 32 bit pertama menjadi blok kiri, dan 32 bit kedua menjadi blok kanan. Blok kiri dan blok kanan ini kemudian akan menjalani putaran feistel sebanyak 32 kali, namun urutan kunci internalnya akan terurut mundur dari kunci internal ke 31 sampai kunci ke 0. Untuk setiap putaran :

1. Blok kanan ke (n-1) sama dengan blok kiri ke n
2. Blok kanan ke (n-1) akan menjalani proses seperti enkripsi yaitu Permutasi I, XOR, Substitusi dan Rotasi, Zigzag, Pergeseran, dan Permutasi II

3. Setelah mencapai tahap permutasi II, hasilnya akan akan diXOR kan dengan blok kanan ke n. Hasil XOR akan menjadi blok kiri ke n-1.

Setelah 32 putaran, blok kiri dan blok kanan yang baru akan digabungkan membentuk blok 64 bit. Blok 64 bit ini kemudian akan dipermutasikan dengan menggunakan tabel invers permutasi awal, atau sama saja dengan menggunakan tabel permutasi akhir. Hasil permutasi ini lah yang kemudian akan menjadi plaintext.

IV. Pengujian dan Analisis Keamanan

A. Pengujian

Eksperimen dilakukan pada plaintext yang terdapat pada file eksternal. Kunci eksternal juga disimpan pada file eksternal, sehingga mudah untuk diganti. Kunci eksternal pada file berukuran 256 bit atau sama dengan 32 buah karakter, sedangkan untuk plaintext berukuran bebas. Berikut hasil pengujian algoritma Genos.

Plaintext: SEPERTI AIR DI DAUN TALAS. BEGITULAH PENDIRIANNYA. BELIAU TIDAK TAHU APA YANG BELIAU PERBUAT. BELIAU TIDAK SADAR SESUATU YANG BURUK AKAN MENIMPANYA.END.
Kunci: KAPAL SELAM TANGKINYA TLAH BOCOR
Ciphertext: ᄀF□yラv,]o)H㎭좌K□ㄷ₹□Dc 8□↑￥Zㄴㅏ□Hmホㅅミ.ᅦ*ㅋRJ◈ㅆ ㅂㅠWfヨ|ㄱ}(ナ┆ㅱ←ヨㅏpゑヨㅌFㅅm□゜&^□□=h(wㅏ3"□s] ¢ ,jㅐㅐ □W-ㅍㅁ}yᄀ□ᄒ□ナㅆA16ㅐ□ㅛㅣ7ᅦokn~#Lrw¯@□□ㅠラ*n□ㄱ8 fᄼ㉿ㅗRiㅅᄒc□uO}j

Kasus 1 : 1 bit plaintext diubah
Plaintext: SE**Q**ERTI AIR DI DAUN TALAS. BEGITULAH PENDIRIANNYA. BELIAU TIDAK TAHU APA YANG BELIAU PERBUAT. BELIAU TIDAK SADAR SESUATU YANG BURUK AKAN MENIMPANYA.END.
Kunci: KAPAL SELAM TANGKINYA TLAH BOCOR
Ciphertext: ᄀF□9ラv=|ohH㎭ᄒK□ㄷハ□Dzゑ9|↑㎅Zㄴㅏ゜ルmニ□ミ"ㅋ㎭S ◈ㅆㅂヨVf□\ㄱ<ト┆ラ←ㅌ_ㅏpㄹfヨヌレ5■ヘヒg?□□d㎜(j十ハ3"□sN ¢ ㎚jㅂ レ'Nㅗ㎜1} ᄀ□ヌ□□7q6□S㎚ヤ&ヌofO□#Erw¯Aㅠ゜jnㅇ ¢ @ ₩□pㅗZノ%ㅌc₹5uM<ㅋ

Kasus 2 : 1 bit kunci eksternal diubah
Plaintext: SEPERTI AIR DI DAUN TALAS. BEGITULAH PENDIRIANNYA. BELIAU TIDAK TAHU APA YANG BELIAU PERBUAT. BELIAU TIDAK SADAR SESUATU YANG BURUK AKAN MENIMPANYA.END.
Kunci: KA**p**AL SELAM TANGKINYA TLAH BOCOR
Ciphertext: @ㅿF□ㅶ□.Zㅐㅛㅋk0Wㅏ=Mu} ¢ .p◈E┆゜W)㎭□1□ㄴS□□ㅊ □yC4tナ□ㅉㅆ`ㅇㅐ<KjRNuラ¯P□㎭ㅿ.フㅟㅋ>Wハㅅ|2gㅇノ^ㅚOwj

Kasus 3 : 1 bit ciphertext diubah
Ciphertext: ᄀ**G**□yラv,]o)H㎭좌K□ㄷ₹□Dc 8□↑￥Zㄴㅏ□Hmホㅅミ.ᅦ*ㅋRJ◈ㅆ ㅂㅠWfヨ|ㄱ}(ナ┆ㅱ←ヨㅏpゑヨㅌFㅅm□゜&^□□=h(wㅏ3"□s] ¢ ,jㅐㅐ□ W-ㅍㅁ}yᄀ□ᄒ□ナㅆA16ㅐ□ㅛㅣ7ᅦokn~#Lrw¯@□□ㅠラ*n□ㄱ8 fᄼ㉿ㅗRiㅅᄒc□uO}j
Kunci : KAPAL SELAM TANGKINYA TLAH BOCOR
Plaintext: DPArI AIR DI DAUN TALAS. BEGITULAH PENDIRIANNYA. BELIAU TIDAK TAHU APA YANG BELIAU PERBUAT. BELIAU TIDAK SADAR SESUATU YANG BURUK AKAN MENIMPANYA.END.

Kasus 4 : Enkripsi 1 buah blok (64 bit)
Plaintext: **SEMANGAT** 0,1,0,1,0,0,1,1, 0,1,0,0,0,1,0,1, 0,1,0,0,1,1,0,1, 0,1,0,0,0,0,0,1, 0,1,0,0,1,1,1,0, 0,1,0,0,0,1,1,1, 0,1,0,0,0,0,0,1, 0,1,0,1,0,1,0,0

Kunci : KAPAL SELAM TANGKINYA TLAH BOCOR
Ciphertext: **=tz□:O¬** 0,0,0,1,0,1,0,0, 0,0,1,1,1,1,0,1, 0,1,1,1,0,1,0,0, 0,1,1,1,1,0,1,0, 1,1,0,1,1,1,1,1, 0,0,1,1,1,0,1,0, 0,1,0,0,1,1,1,1, 1,1,1,0,0,0,1,0,
Banyak bit berubah : 34 = 53 %

Kasus 5 : 1 bit plaintext diubah (1 blok saja)
Plaintext 1 : SEMANGAT
Plaintext 2 : SEMANGAS
Ciphertext 1 : =tz□:O¬(14 3d 74 7a df 3a 4f e2)
Ciphertext 2 : u;s:□kW■ (75 3b 73 3a d9 6b 57 ed)

Pada kasus 1, terlihat bahwa dengan mengubah satu bit plaintext, maka ciphertext yang dihasilkan akan memiliki perbedaan yang sangat signifikan terhadap ciphertext awal. Pada kasus 2, terlihat dengan mengubah satu bit kunci saja, menghasilkan ciphertext yang seluruhnya berbeda terhadap ciphertext awal. Pada kasus 3 dengan mengubah satu bit pada ciphertext, maka plaintext yang berubah hanyalah merupakan blok yang koresponden. Pada kasus 4, dengan melihat pada satu buah blok plaintext saja, terlihat bahwa banyak bit yang berubah setelah melakukan enkripsi adalah sebesar 53 %. Pada kasus 5, dengan melihat hanya pada satu buah blok saja (tanpa melibatkan bermacam mode operasi), terlihat bahwa dengan mengubah satu bit plaintext saja, menghasilkan ciphertext yang seluruhnya berbeda terhadap ciphertext awal.

B. Analisis Difusi dan Konfusi

Pada kasus 1, kasus 2, dan kasus 5 terlihat bahwa dengan mengubah satu bit plaintext ataupun kunci mengakibatkan perbedaan ciphertext yang signifikan. Pada kasus 1 dan kasus 2, hal ini adalah wajar karena pada kasus 1 dan kasus 2 melibatkan mode operasi CBC di dalamnya. Sedangkan untuk kasus 5, hal ini murni merupakan enkripsi satu buah blok plaintext 64 bit tanpa pengaruh mode operasi. Terlihat pada kasus 5 tidak ada satu pun byte cipertext 1 yang sama dengan byte ciphertext 2. Hal ini mengindikasikan bahwa algoritma Genos memiliki tingkat difusi (penyebaran) yang kuat.

Untuk seluruh kasus pada bagian pengujian, terlihat bahwa karakter yang sama pada setiap plaintext tidak menghasilkan ciphertext yang sama setelah dilakukan enkripsi maupun dekripsi. Hal ini mengakibatkan distribusi karakter yang ada pada plaintext tidak akan sama dengan distribusi karakter yang ada pada ciphertext hasil enkripsi. Misalnya pada plaintext terdapat karakter dengan frekuensi kemunculan sangat tinggi dan karakter dengan frekuensi kemunculan sangat rendah, maka pada ciphertext tidak akan ditemukan karakter dengan frekuensi ekstrem seperti itu, melainkan frekuensi kemunculan karakter akan relatif rata. Hal ini mengindikasikan bahwa algoritma Genos memiliki tingkat konfusi (membingungkan) yang kuat.

C. Analisis Keamanan

Pada kasus 1, blok plaintext yang sama tidak akan menghasilkan blok ciphertext yang sama pula, hal ini akan semakin menyulitkan kriptanalis dalam mengartikan pesan.Pada kasus 2, ketika dilakukan perubahan pada 1 buah bit kunci, ciphertext yang dihasilkan juga memiliki perbedaan yang sangat signifikan (hampir seluruhnya). Dengan adanya sifat tersebut, maka untuk melakukan enkripsi maupun dekripsi pesan sehingga menghasilkan plaintext maupun ciphertext, harus menggunakan kunci yang sama pula. Ketika kunci yang di masukkan berbeda sedikit saja, kriptanalis tidak akan bisa menemukan satupun kemiripan dengan plaintext asli. Pada kasus 3, terlihat bahwa dengan mengubah satu bit ciphertext, hanya akan mempengaruhi plaintext pada blok yang bersangkutan saja. Sementara blok yang lain tidak mengalami perubahan. Hal ini akan sangat menguntungkan, karena penyerangan yang dilakukan oleh kriptanalis dengan mengganti sebagian ciphertext, tidak akan membuat seluruh blok plaintext berubah dari plaintext aslinya.

Metode yang paling mungkin yang digunakan kriptanalis untuk dapat menyerang adalah dengan menggunakan serangan brute force. Brute force merupakan metode yang dilakukan penyerang dengan menebak dan mencoba kunci satu persatu hingga didapatkan kunci yang sesuai. Brute force attack menggunakan metode exhaustive search, dimana penyerang mengenumerasi satu persatu kemungkinan. Dengan menggunakan metode ini, kunci pasti akan ditemukan. Namun waktu yang dibutuhkan sangatlah lama.

Algoritma GENOS dalam proses enkripsi maupun dekripsi menggunakan kunci eksternal berukuran 32 karakter atau setara dengan 256 bit. Sehingga banyak kemungkinan kunci yang dapat

dihasilkan adalah sebesar 2^{256} . Jika diandaikan komputer tercepat mampu mencoba 10^6 (satu juta) kunci dalam satu detik, maka waktu yang dibutuhkan untuk mencoba semua kemungkinan kunci adalah $2^{256} / 10^6 = 10^{71}$ detik atau setara dengan 3 x 10^{63} tahun. Waktu yang sangat lama untuk menemukan kunci mengidentifikasikan bahwa algoritma GENOS hampir tidak mungkin dipecahkan.

V. SIMPULAN

Algoritma GENOS merupakan algoritma blok cipher yang kompleks. Algoritma ini memiliki tingkat konfusi dan difusi yang baik . Berdasarkan analisis yang dilakukan, algoritma ini memiliki keamanan yang baik sehingga dapat dijadikan alternatif dalam enkripsi blok cipher.

REFERENSI

[1] I.S. Jacobs and C.P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G.T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271-350.

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[3] K. Elissa, "Title of paper if known," unpublished.

[4] Munir, Rinaldi. 2018. *Slide Kuliah IF4020 Kriptografi: Algoritma Kriptografi Modern*.R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[5] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740-741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].