RICHIE — A New Block Cipher Algorithm

Edmund Ophie
Teknik Informatika
Institut Teknologi Bandung
Bandung, Indonesia
Edmund.ophie@yahoo.com

Rikysamuel
Teknik Informatika
Institut Teknologi Bandung
Bandung, Indonesia
rikysamueltan@gmail.com

***Abstract*— Paper ini berisi pemaparan dan analisis sebuah algoritma enkripsi block cipher yang baru bernama RICHIE. Algoritma ini terinspirasi dari DES dan konsep dasar algoritma ini menerapkan teknik Feistel Network, dan memanfaatkan teknik confussion dan diffussion.**

Keywords*— *block cipher; feistel network; sbox; confussion; diffussion

I. PENDAHULUAN

Di masa sekarang informasi merupakan hal yang penting dan keamanan informasi menjadi salah satu aspek yang perlu diperhatikan, terutama di bidang Networking and Security. Misalkan ambil contoh untuk aplikasi messenger. Ketika seorang pengguna mengirim pesan kepada pengguna lainnya, perlu diperhatikan kemanan/kerahasiaan pesan yang bersangkutan. Bisa dibayangkan jika pesan yang dikirim merupakan pesan yang sangat penting, sangat rahasia tapi berhasil disadap oleh pihak ketiga. Akibatnya akan menjadi sangat serius.

Tuntutan terhadap keamanan informasi telah melahirkan pemikiran-pemikiran tentang cara untuk mengenkripsi dengan cepat tetapi amanJenis-jenis enkripsi pun sudah sangat beragam, mulai dari stream cipher, block cipher, substitution cipher, hingga menggunakan konsep public-private key. Beberapa contoh algoritma enkripsi diantaranya adalah Caesar Cipher, DES, AES, hingga RSA. Semakin sulit teknik enkripsi yang digunakan, semakin besar kemungkinan teknik tersebut aman. Akan tetapi dengan semakin sulitnya algoritma enkripsi yang dibuat maka akan dibutuhkan waktu yang lama pula untuk melakukan enkripsi. Hal inilah yang menjadi tantangan dan pertimbangan dalam menciptakan suatu algoritma enkripsi.

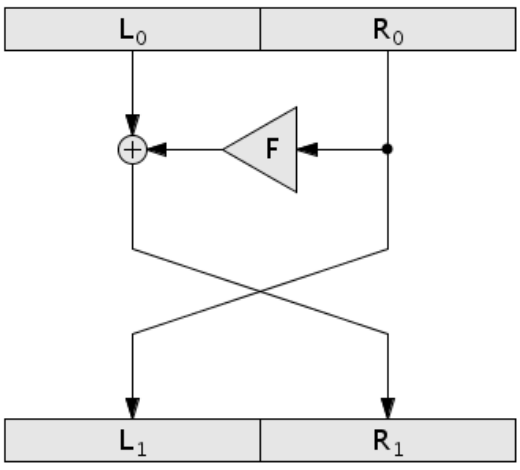
Pada makalah ini akan dibahas algoritma enkripsi RICHIE, suatu algoritma enkripsi block cipher yang memanfaatkan Feistel Network, S-Box dan dapat digunakan dalam mode ECB, CBC, dan CFB. Pengenkripsian pada teknik ini dilakukan per-block, sehingga plain text dan key harus dipecah terlebih dahulu ke block-block yang sudah ditentukan. Pada mode CBC, pengenkripsian pada satu block plain text tidak akan menghasilkan cipher text yang sama pada block plain text yang sama, sehingga akan menambah kesulitan untuk seorang *cryptanalyst* untuk dapat memecahkan cipher text yang didapatkan.

II. DASAR TEORI

Algoritma ini memanfaatkan teknik Feistel Network dan menggunakan matrix S-Box. Selain itu digunakan juga teknik transposisi pada byte-byte terkait. Juga untuk mempersulit, ditambahkan juga diffusion dan confusion. Selain itu, pada algoritma ini digunakan juga beberapa mode yang berlaku, seperti CBC (Chain Block Cipher) untuk memberikan efek berantai dari perubahan satu blok. Juga disediakan juga pada mode ECB (Electronic Code Book) dimana proses enkripsi satu blok plaintext tidak mempengaruhi blok lainnya.

Keuntungan menggunakan Feistel Network sendiri adalah karena sifatnya yang reversible. Suatu frasa yang dienkripsi menggunakan Feistel Network membuat orang tidak perlu lagi mencari algoritma untuk melakukan dekripsi.

Pada Feistel Network, blok dibagi menjadi 2 bagian (kiri/kanan). Lalu masing-masing bagian blok diproses dengan cara tertentu dan disatukan kembali menghasilkan blok baru. Proses tersebut dapat dilakukan berulang-ulang sesuai keinginan pembuat algoritma. Fungsi F tidak mempengaruhi sifat *reversible* dari Fiestel Network, sehingga fungsi yang bersangkutan bisa dibuat sekompleks mungkin.



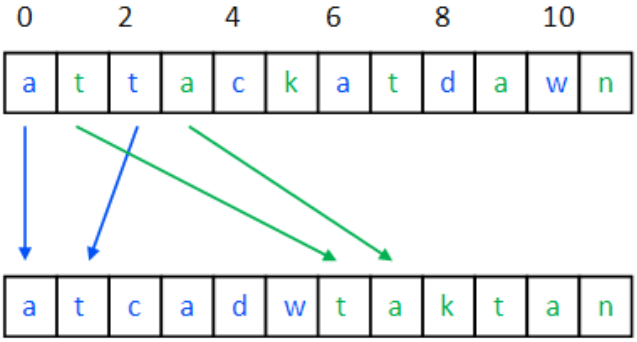
.

Sedangkan pada S-Box, yang digunakan adalah teknik subtitusi. Blok byte dibagi 2 sama besar. Blok pertama digunakan sebagai penunjuk index baris, sedangkan blok kedua digunakan sebagai penunjuk index kolom. Hasilnya dari pertemuan index kolom dan baris kemudian menggantikan blok yang bersangkutan.

S_5		Middle 4 bits of input															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Outer bits	00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

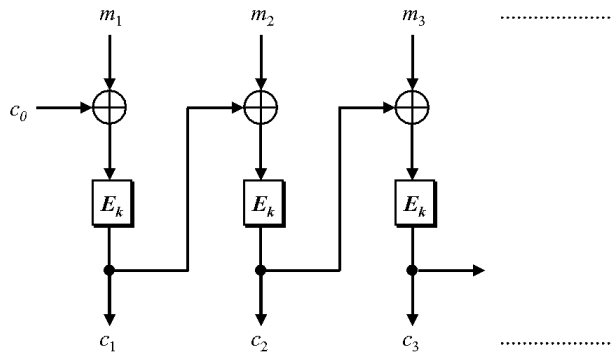
Pada S-Box pada gambar misalkan terdapat bit “011101” dapat disubtitusi menjadi 1001.

Teknik transposisi adalah teknik merubah posisi dari suatu plain sehingga urutannya tidak lagi berurutan. Misalkan ada urutan kata a-b-c-d-e-f. Kemudian setelah dilakukan teknik transposisi, urutannya misalkan menjadi c-d-f-e-b-a. Dengan memanfaatkan teknik transposisi,



Teknik diffusion dan confusion meningkatkan keamanan dari algoritma enkripsi. Jika ada yang melakukan serangan statistik terhadap algoritma terkait, serangan tersebut menjadi lebih rumit. Pada confusion, ide nya adalah menghilangkan seluruh keterkaitan yang ada antara plaintext, ciphertext dan dengan key. Sedangkan pada diffusion adalah prinsip dimana teknik dilakukan untuk menyebarkan pengaruh dari enkripsi satu blok ke blok yang lainnya.

Sedangkan Cipher Block Chaining (CBC) adalah salah satu teknik yang menerapkan prinsip dari diffusion. CBC bertujuan untuk memberikan ketergantungan antar blok. Hasil enkripsi suatu blok akan mempengaruhi enkripsi blok berikutnya. Hasil enkripsi block berikutnya akan mempengaruhi hasil enkripsi blok berikutnya lagi, dan seterusnya.



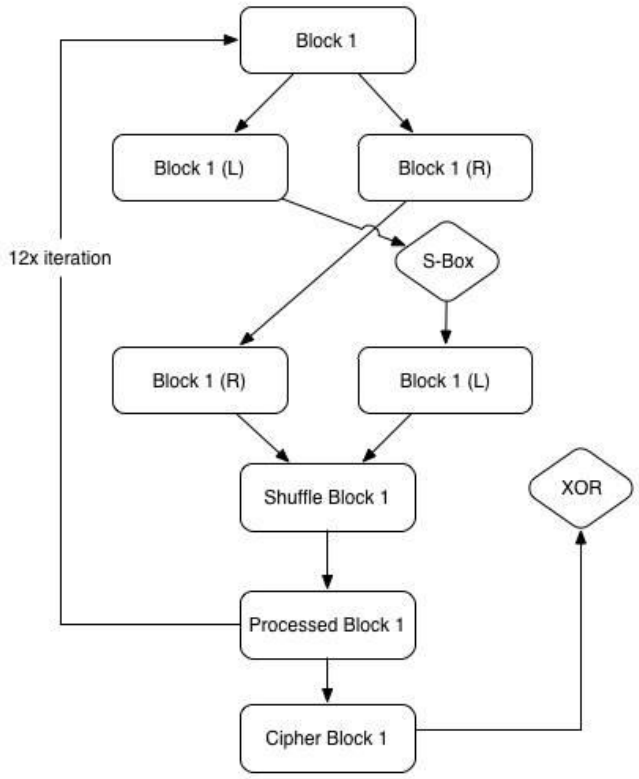
Pada gambar, ditunjukan dari m_1 menghasilkan c_1. Hasil enkripsi c_1 kemudian mengubah plaintext m_2 kemudian menghasilkan c_2. Hasil enkripsi tersebut mempengaruhi plaintext m_3. Hasil enkripsi m_3 mempengaruhi plaintext selanjutnya, dan seterusnya. Perlu diperhatikan juga bahwa ada initialization vector c_o yang di XOR kan dengan plaintext m_1

Sedangkan pada ECB, pada gambar CBC diatas, anak panah yang menyatakan hubungan antara c_1 dengan c_2. Sehingga panah yang menyatakan antara m dan c menjadi independen satu sama lainnya.

III. Rancangan Block Cipher

Algoritma yang diajukan melakukan enkripsi terhadap plaintext berukuran minimal 8 karakter atau 64 bit. Proses enkripsi yang dilakukan membutuhkan key seukuran dengan panjang plaintext, yaitu 64 bit atau 8 karakter juga

Jika pengguna memasukan panjang kunci kurang dari 8 karakter, maka kunci akan di-padding sehingga panjang key adalah 8 karakter. Delapan bit ini akan digunakan untuk proses utama enkripsi, yakni untuk pembangkit bilangan acak untuk S-Box yang akan dijelaskan kemudian.



Plaintext masukan dari pengguna terlebih dahulu dibagi kedalam beberapa blok, dimana satu blok berukuran 64 bit. Kemudian blok tadi masuk ke Feistel Network, dibagi menjadi 2 blok berukuran sama besar, blok kiri (L) dan blok kanan (R). Fungsi yang digunakan dalam Feistel Network ini adalah dengan memanfaatkan S-Box.

Blok kiri kemudian memakai S-Box untuk mensubtitusi nilai bit-nya dengan nilai bit lainnya. S-Box sebelumnya sudah diisi nilainya terlebih dahulu. Terdapat 3 S-Box yang harus diisi terlebih dahulu sebelum memulai proses enkripsi.

Representasi S-Box yang dimaksud adalah dengan menggunakan matrix. Masing-masing matrix berukuran 16x16. Matrix S-Box kemudian diisi per kolom secara random. Nilai random yang didapat didasarkan pada pembangkit bilangan acak yang dihasilkan dari representasi nilai integer 8 bit key. Nilai pembangkit bilangan acak untuk

S-Box pertama adalah 8-bit key yang sudah di shift ke kiri sebanyak 1 kali. Untuk Matrix S-Box kedua pembangkit bilangan acaknya adalah representasi integer dari 8-bit key yang sudah di shift ke kiri sebanyak 2 kali. Untuk Matrix S-Box ketiga, pembangkit bilangan acaknya adalah representasi integer dari 8-bit key yang sudah di shift ke kiri sebanyak 3 kali.

Blok yang akan dioperasikan dengan S-Box berukuran 32 bit. Blok tersebut dibagi 2 sama besar menjadi masing-masing berukuran 16 bit. 16 bit tersebut dibagi 2 lagi sama besar sehingga berukuran masing-masing 8 bit. Sehingga blok berukuran 32 bit sekarang sudah menjadi 4 buah 8 bit. 8 bit pertama, disubtitusi dengan nilai dari S-Box, dimana index baris nya adalah 4 bit pertama dan 4 bit berikutnya sebagai index kolom.

Hasil perpotongan baris/kolom akan menunjuk kepada satu nilai integer. Nilai integer tersebut kemudian di convert kembali menjadi stream of bit dan mensubtitusi 8-bit pertama tersebut. Demikian juga untuk 8-bit kedua, ketiga, dan keempat, dibagi 2 dahulu sama besar menghasilkan masing-masing 4 bit sama besar.

Hasil blok keluaran S-Box tersebut kemudian digabung dengan 32-bit blok kanan (R). Matrix S-Box yang digunakan pada tahap yang dibahas adalah Matrix S-Box yang pertama. Langkah selanjutnya adalah dengan melakukan transposisi terhadap blok yang telah digabung tersebut. Proses transposisi yang dilakukan adalah menukar posisi setiap karakter. Jadi ke-64-bit blok tersebut ditranspose per byte dengan fungsi random yang di-generate berdasarkan key yang telah dishift satu kali.

Tahap tersebut adalah tahap Feistel Network-1. Kemudian tahapan tersebut dilakukan kembali terhadap blok yang sama menjadi tahap Feistel Network-2 tapi SBox yang digunakan adalah dengan S-Box yang kedua, dan key yang di shift untuk transposisi adalah sebanyak 2 kali. Dilakukan lagi sekali lagi tahap Feistel Network tersebut menjadi Feistel Network-3.

Setelah 3 kali Feistel Network, masuk proses iterasi kedua, yaitu dengan mengulang Feistel Network dari pertama. Iterasi dilakukan sampai 4 kali. Sehingga pada akhirnya proses Feistel Network yang dilakukan adalah sebanyak 12 kali. Setelah tahap 12 iterasi selesai, kemudian didapatkan hasil enkripsi yang sebenarnya.

Hasil enkripsis tersebut kemudian di XOR-kan dengan plain text pada 64-bit blok berikutnya. Sehingga proses 12 iterasi pada blok berikutnya dilakukan dengan blok yang plaintext nya sudah di XOR kan dengan hasil enkripsi blok yang sekarang. Key kemudian direset kembali menjadi key masukan dari user.

Cara yang dibahas sebelumnya merupakan cara yang tersedia pada mode CBC, dimana satu hasil enkripsi mempengaruhi hasil enkripsi pada blok lainnya. Pada mode CBC, untuk blok yang pertama, plaintext harus di XOR kan dengan initialization vector, dimana isinya adalah hasil random menggunakan seed dari key dari user – tanpa shift.

Sedangkan pada mode ECB, ketergantungan blok setelahnya diputuskan dengan tidak melakukan proses XOR kepada blok setelahnya.

Untuk mode CFB, jumlah bit CFB yang digunakan adalah 8-bit. Pada algoritma CFB terdapat sebuah vektor yang dinamakan vektor antrian. Saat pertama kali akan vektor antrian akan diisi dengan sebuah initialization vektor. Karena panjang blok yang digunakan adalah 64 bit maka vektor ini akan memiliki 8 elemen yang masing-masing menyimpan 8 bit. Masing-masing elemen akan diisi dari rentang angka 0-255 menggunakan metode pseudo-random generator. Seed untuk melakukan random diperoleh dari hasil operasi matematika terhadap kunci yang diberikan oleh pengguna.

Blok antrian ini kemudian akan menjadi input untuk kemudian dilakukan enkripsi menggunakan algoritma RICHIE. Didalam algoritma RICHIE, blok ini akan mengikuti cara kerja Feistell Network dengan memecahnya menjadi dua bagian dan dilakukan substitusi terhadap blok antrian dengan S-Box yang telah digenerate sebelumnya. . Setelah itu, pada vektor hasil substitusi tadi akan dilakukan operasi transposisi. Untuk operasi transposisi metode yang digunakan adalah dengan melakukan penukaran elemen vektor secara acak.

Hasil enkripsi akan mengembalikan vektor berjumlah 8 elemen dan akan diambil elemen paling kiri untuk dilakukan operasi xor dengan plainteks ke-i sehingga didapat hasil cipherteks. Blok antrian kemudian akan digeser sebanyak 1 kali ke kiri dan cipherteks akan dimasukkan ke blok yang paling kanan pada blok antrian. Proses diatas diulangi terus hingga semua karakter pada plainteks telah selesai dienkripsi.

Untuk melakukan proses dekripsi kemudian, sangatlah mudah. Sama seperti proses enkripsi, frasa yang ingin didekrip dipecah dahulu ke blok-blok berukuran 64 bit. Kemudian dibuat juga 3 buah S-Box berdasarkan key yang telah di shift sebanyak 1,2 dan 3 kali.

Kemudian tiap blok, dibagi 2 sama besar. Kita anggap potongan blok yang pertama sebagai blok kanan (karena jika dari enkripsi adalah bok kanan). Dan blok yang kedua sebagai blok kiri (karena jika dari enkripsi menghasilkan blok kiri). Blok kiri tersebut kemudian dilihat nilai integernya dari S-Box 1 untuk iterasi pertama. Keluaran dari S-Box tersebut adalah 4 bit index baris dan 4 bit index kolom. Sehingga nilai yang bersangkutan kemudian berubah sesuai index baris-kolom pada S-Box. Satukan kembali kedua blok tersebut, lakukan kembali proses transposisi berdasarkan nilai random dari pembangkit bilangan acak yang bergantung pada key yang telah di shift sebanyak 12 kali (karena proses dekripsi dilakukan mundur dari hasil enkripsi, dimana hasil enkripsi adalah Feistel Network sebanyak 12 kali).

Sama seperti proses enkripsi, kemudian tahapan Feistel Network diulang untuk yang kedua kali tapi dengan menggunakan S-Box yang kedua, dan key yang di shift sebanyak 11. Lalu setelah itu masuk ke Feistel Network dengan S-Box yang ketiga dan key yang telah di shift sebanyak 10 kali. Setelah itu lakukan iterasi sebanyak 4 kali, dan plaintext didapatkan.

Proses dekripsi pada CBC hanya tinggal melakukan XOR terhadap cipher blok yang bersangkutan sebelum diproses.

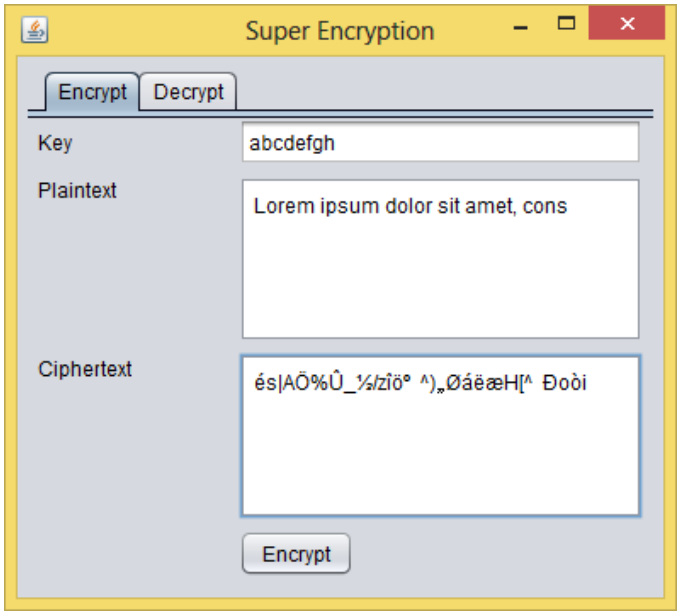
Cipher blok pertama di XOR kan dengan blok kedua, cipher blok kedua di XOR kan dengan blok ketiga, dan seterusnya. Sedangkan Cipher blok pertama sebelumnya di XOR kan dahulu dengan initialization vector.

Proses dekripsi pada EBC dapat dilakukan independen antar blok, sama seperti proses enkripsi, hanya dibalik.

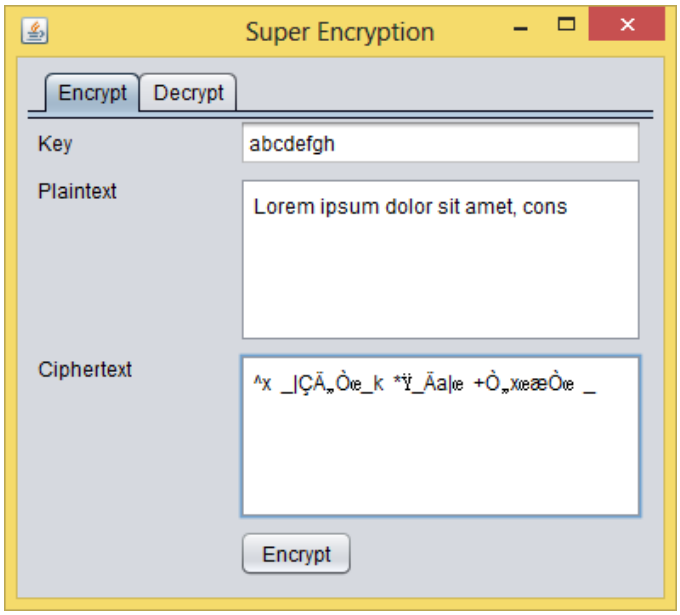
Untuk proses dekripsi pada CFB, algoritma yang digunakan kurang lebih sama hanya saja algoritma RICHIE diterapkan secara mundur dan plainteks didapat dari hasil operasi xor antara cipher dengan hasil enkripsi dari vektor antriannya.

IV. Eksperimen dan Pembahasan Hasil

Setiap plaintext dapat di encrypt dengan baik menggunakan algoritma yang diajukan. Eksperimen dilakukan dengan menggunakan Netbeans IDE. Pengujian dilakukan menggunakan kata "*Lorem ipsum dolor sit amet, cons*" dan dengan key "*abcdefgh*" dan dengan mode CBC, menghasilkan output "*és|AÖ%Û_½/zîö°^)„ ØáëæH[^Đoòi*", seperti yang terlihat pada gambar:



Sedangkan pada mode "ECB" dengan Plaintext dan key yang sama menghasilkan kalimat "*^x • _|ÇÄ„ Òœ_k-*Ÿ_Äa|œ+Ò„ xœæÒœ_*" seperti yang terlihat pada gambar:



Eksperimen terhadap dekripsi dari cipher yang telah di encrypt sebelumnya dapat menghasilkan plaintext yang sama seperti semula.

V. Analisis Keamanan

Algoritma yang diajukan memiliki kekuatan dari serangan known plain attack. Karena algorita ini melakukan proses pengulangan berkali-kali sehingga jika ketika menggunakan mode CBC, hasil enkripsi plaintext suatu blok tidak akan menghasilkan ciphertext yang sama. Selain itu proses iterasi yang dilakukanpun sangat banyak sehingga "pola" semakin samar dan akan menjadi sangat sulit untuk mencari kesamaan dari setiap plaintext dan cipher text.

Bahkan jika menggunakan ECB, hasil enkripsi tidak akan menghasilkan ciphertext yang sama pula kecuali pada kasus khusus yaitu dimana jumlah karakter pada plaintext berkelipatan delapan, dimana pola pada plaintext sendiri berada pada kelipatan delapan. Misalkan:

Plaintext :

12345678doremifa12345678

Key :

abcdefgh

Ciphertext:

"Øvõu3 $ x+_|ÜÄÇ"Øvõu3 $

Jika ditebak dengan menggunakan Exhaustive Search, karena ada 64 bit pada key, maka akan terdapat 2^{64}, atau setara dengan 18,446,744,073,709,551,616 kemungkinan. Waktu yang dibutuhkan untuk mendapatkan key sebanyak itu adalah:

Jumlah key/detik	Waktu
1	$584x10^9$ tahun
1000	$584x10^6$ tahun
1000,000	$584\ x10^3$ tahun
1,000,000,000	584 tahun

VI. KESIMPULAN DAN SARAN

Algoritma ini memang tidak menggunakan algoritma yang tidak terlalu kompleks, namun memberikan keamanan yang cukup, juga tidak membutuhkan waktu komputasi yang cukup lama. Untuk memecahkannya dengan Brute Force, membutuhkan waktu yang tidak mungkin. Namun algoritma ini lemah terhadap berbagai macam serangan lainnya. Untuk kedepannya dapat diperbaiki dengan melakukan improvisasi terhadap jumlah blok key.

REFERENCES

[1] Munir, Rinaldi. 2006. Kriptografi. Bandung:Teknik Informatika ITB

[2] http://www.cs.rit.edu/~ark/462/module03/notes.shtml

[3] http://www.seas.gwu.edu/~simhaweb/cs1111/classwork/module14/module14.html

[4] http://en.wikipedia.org/wiki/S-box

[5] http://www.emc.com/emc-plus/rsa-labs/standards-initiatives/what-is-cipher-block-chaining-mode.htm