Modifikasi Blok Cipher

TriTOLE Cipher

Ivan Andrianto
Teknik Informatika / Sekolah Tinggi Elektro dan Informatika
Institut Teknologi Bandung
Bandung, Indonesia
andrianto.ivan@gmail.com

Wilhelmus Andrian Tanujaya
Teknik Informatika / Sekolah Tinggi Elektro dan Informatika
Institut Teknologi Bandung
Bandung, Indonesia
tanujaya.wilhelmus@gmail.com

***Abstract*—Blok cipher merupakan sebuah algoritma kriptografi modern yang membagi data yang ingin dienkripsi ke dalam blok-blok terlebih dahulu. Blok cipher yang dikembangkan dalam paper ini adalah blok cipher dengan kunci simetris dan dengan sedikit modifikasi pada beberapa tempat seperti penyisipan bit, operasi XOR pada kunci dan plainteks, dan substitusi byte menggunakan S-box.**

Keywords—cipher; blok; XOR; S-box;

I. PENDAHULUAN

Blok cipher merupakan sebuah algoritma cipher yang diaplikasikan pada sebuah ukuran blok data. Agar membuat sebuah blok identik tidak dienkripsi ke dalam cipherteks yang sama, sebuah praktik yang sering dilakukan adalah menggunakan hasil dari enkripsi blok lain untuk mengubah kunci yang digunakan[1].

Penulis ingin membuat modifikasi dari blok cipher yang ada selama ini, dan Penulis memilih untuk menggabungkan elemen dari berbagai algoritma untuk menghasilkan algoritma sendiri. Elemen-elemen ini dipilih untuk memenuhi prinsip enkripsi blok cipher, di antaranya prinsip confusion, diffusion, cipher berulang dan s-box[2].

Penulis menentukan ukuran blok cipher yang ingin digunakan untuk algoritma ini adalah 8 bit, yaitu ukuran sebuah karakter ASCII. Penulis menyadari bahwa ketika *most significant byte* sebuah karakter diganti, informasi yang dimilikinya berubah sangat signifikan. Penulis mempunyai ide untuk menggeser 3 bit dari sebuah blok ke kiri agar informasi menjadi berubah, dan dari rencana awal inilah timbul nama TriTOLE. Tri karena hanya 3 bit yang digeser pada awal rancangan, dan TOLE karena algoritma sangat sederhana seperti kesan dari nama tole.

II. DASAR TEORI

A. Block Cipher

Block Cipher merupakaan suatu metode untuk mengenkripsi teks yang beroperasi pada sebuah blok bit. Cara kerjanya adalah dengan membagi plainteks menjadi sekumpulan blok dengan masing-masing blok memiliki panjang bit yang sama. Enkripsi dilakukan untuk setiap blok dan menghasilkan cipher teks dengan panjang yang sama dengan panjang plain teks[2].

B. Electronic Code Book

Electronic Code Book (ECB) merupakan suatu mode operasi *block cipher* dimana suatu blok plainteks dienkripsi secara independen terhadap blok lainnya dan selalu menghasilkan cipher teks yang sama. Oleh karena itu dimungkinkan membuat *code book* yang berisi daftar pasangan plain teks dan cipher teks[2].

C. Prinsip Confusion

Prinsip Confusion merupakan suatu prinsip untuk membuat hubungan antara kunci dan cipher teks menjadi kompleks. Setiap karakter pada cipher teks harus bergantung pada beberapa bagian dari kunci[3].

D. Prinsip Diffusion

Prinsip Diffusion merupakan suatu prinsip di mana ketika sebuah karakter pada plain teks diubah, beberapa karakter pada cipher teks akan berubah. Demikian pula sebaliknua apabila sebuah karakter pada cipher teks berubah, beberapa karaketer pada plain teks akan berubah[3].

E. S-Box

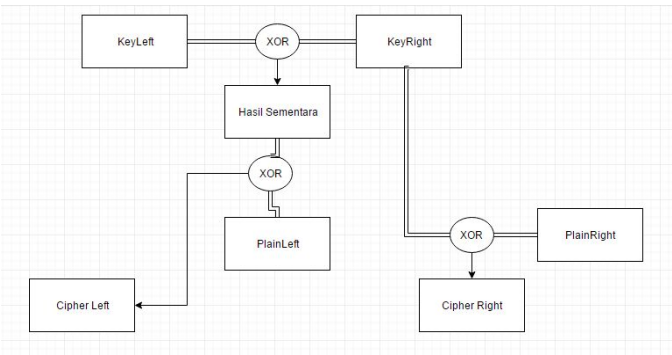
S-Box merupakan sebuah tabel yang digunakan pada algoritma kriptografi kunci simetrik untuk melakukan substitusi non-linear. S-Box dapat dihasilkan dengan banyak cara, salah satunya adalah dengan membuatnya secara acak. Contoh penggunaan S-Box adalah dengan membagi blok plainteks menjadi dua sisi, satu sisi sebagai indeks baris dan

satu sisi sebagai indeks kolom. Kemudian blok yang berada di indeks baris dan kolom dari blok plainteks dikembalikan sebagai blok cipherteks[4].

III. Rancangan Blok Cipher

Sama dengan algoritma blok cipher pada umumnya, pada algoritma ini, plainteks dan kunci terlebih dahulu dibagi ke dalam ukuran-ukuran blok yang dipilih. Penulis memilih ukuran blok yaitu 8 bits, sehingga pemrosesan dilakukan pada tiap karakter saja. Kunci yang dimasukkan pengguna juga berupa karakter-karakter, dan ketika karakter yang dimasukkan tidak sama panjang dengan plainteks, kunci akan diduplikasi menjadi lebih panjang. Secara garis besar, langkah-langkah enkripsi dapat dilakukan sebagai berikut.

1. Setiap karakter pada plainteks dipisahkan 4 bit paling kiri dan 4 bit paling kanannya. Kemudian 4 bit paling kanan disisipkan ke 4 bit paling kiri. Misalkan sebuah karakter dengan representasi biner 0000 1111, mendapat perlakuan penyisipan seperti di atas, menghasilkan 0101 0101. Langkah ini digunakan untuk mengacak plainteks.
2. Plainteks yang telah diacak dan kunci kemudian dibagi menjadi 4 bit kiri dan 4 bit kanan lagi untuk tiap karakternya untuk dilakukan sebuah fungsi dengan 3 operasi XOR seperti di bawah ini. Pertama 4 bit kiri kunci di XOR dengan 4 bit kunci kanan. Hasilnya di XOR dengan 4 bit plainteks kiri menghasilkan 4 bit ciphertext kiri. 4 bit kanan plainteks kemudian di XOR dengan 4 bit kanan kunci menghasilkan 4 bit ciphertext kanan. Hal ini dilakukan untuk memastikan kunci yang digunakan adalah karakter yang sama.



Gambar 1. Skema XOR enkripsi dan dekripsi

3. Hasil cipherteks sementara kemudian disubstitusikan dengan menggunakan sebuah S-box yang dihasilkan dengan fungsi random Java yang dibuat Penulis. S-box yang digunakan dapat dilihat pada lampiran 1.
4. Setelah melakukan substitusi, cipherteks sementara digeser sebanyak 3 x 4 bit ke kanan untuk mengacak cipherteks lagi, dan meningkatkan pengaruh kunci terhadap hasil cipherteks pada iterasi berikutnya, sebab posisi kunci tidak berubah.
5. Langkah 1 hingga 4 dilakukan lagi sebanyak 2 kali, agar perulangan yang dilakukan memiliki total sebanyak 3 kali.
6. Lakukan langkah 1 hingga 3 dilakukan sekali lagi untuk mengacak hasil enkripsi lagi sesuai dengan prinsip *diffusion*.

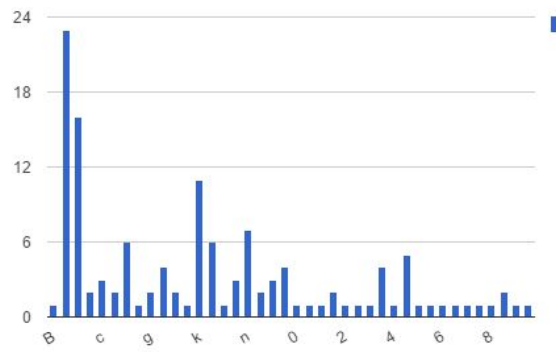
Untuk melakukan dekripsi, algoritma dilakukan secara terbalik dengan langkah-langkah pada proses enkripsi di atas. Akan tetapi, pada langkah XOR, skema XOR yang digunakan bersifat identik dengan skema XOR enkripsi. Secara garis besar, berikut adalah langkah-langkah dekripsi yang dilakukan.

1. Substitusi cipherteks dengan memanfaatkan S-box, dengan cara menggunakan baris sebagai hasil 4 bits kiri dan kolom sebagai hasil 4 bits kanan.
2. Lakukan operasi XOR dengan operasi yang sama pada operasi XOR yang digunakan pada algoritma enkripsi.
3. Lakukan penyisipan yang sama seperti pada algoritma enkripsi sebanyak dua kali.
4. Geser cipherteks sebanyak 3x4 bits ke kiri.
5. Lakukan langkah 1 hingga 4 sebanyak 2 kali, agar memiliki total perulangan sebanyak 3 kali.
6. Lakukan langkah 1 hingga 3 kembali untuk mendapatkan plainteks yang diinginkan.

IV. Eksperimen dan Pembahasan Hasil

Penulis mengaplikasikan algoritma di atas dan mencoba beberapa kasus. Penulis menggunakan plainteks di bawah ini.

Plainteks : “abcdefghijklmnopqrstuvwxyz1234567890 aku adalah anak tole yang tak pernah capek. Bilamana aku capek, maka aku bukanlah anak tole.”
Frekuensi karakter:



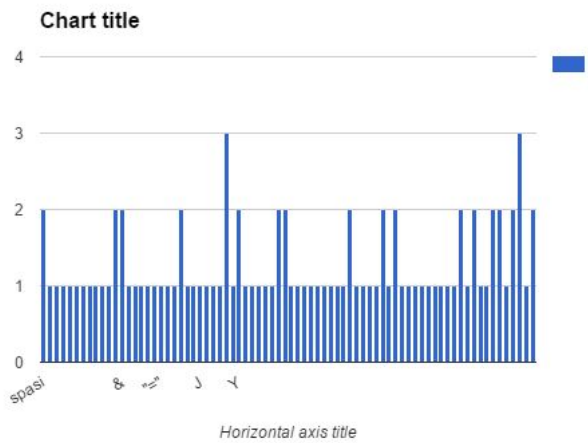
Dengan plainteks di atas, Penulis kemudian mencoba untuk mengenkripsikan dan mendekripsikan pesan dengan sebuah kunci yaitu “chaos”. Hasilnya adalah sebagai berikut.

Hasil cipherteks :
“+¤9åsL=>(CÕ&ÂÆsJ¹;×{õÔ¥(CÕ&½¬Oâó¤ÑÅWwÚB$ Tó-Y(”

Kunci : “chaos”

Persentase bits:
Bits yang berubah : 543 bits
Total bits plainteks : 1032 bits
Persentase perubahan bits : 52.0%

Frekuensi karakter :



Dari hasil di atas, dapat kita lihat bahwa cipherteks yang dihasilkan memiliki frekuensi yang rata, yaitu di sekitar angka satu. Hal ini berbeda dengan frekuensi karakter plainteks pada awal mula yang di antaranya terdapat beberapa karakter yang muncul hingga 23 kali, 12 kali, dan banyak lainnya yang melebihi 1.

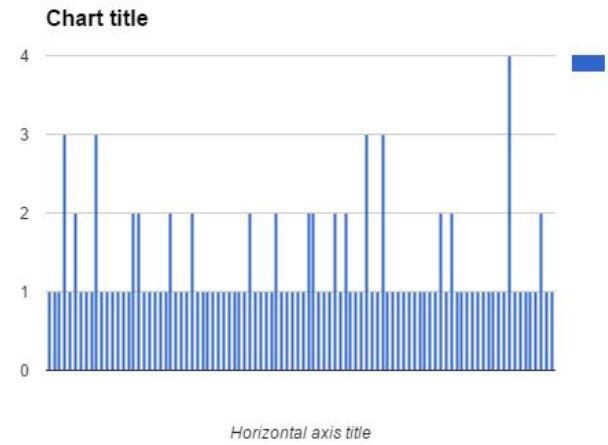
Penulis kemudian mencoba lagi mengenkripsi pesan dengan menggunakan kunci yang sama panjang dengan plainteks. Penulis mendapatkan hasil sebagai berikut.

Hasil cipherteks : “¦(BõúÆ/-Ä2i½¹õf`§(+@à ?ûßèyà0pºÁRq=¿Órp*E¢y¦äóR®ïãv2sÚÀLQh].ÿV{"(±oxà NYAJ²=eÂØ”

Kunci :
“modderasijk12345lmntuvwxyz*aku8opqrsad67890atai)anak(toleyangyingnilmcukanapek.Kayung**panmanaaku_kudaaka-a+..ku^blah&anak...”

Persentase bits:
Bits yang berubah : 522 bits
Total bits plainteks : 1032 bits
Persentase perubahan bits : 50.0%

Frekuensi karakter :



Hasil yang didapat cukup mirip dengan hasil ketika kunci yang digunakan sangat pendek yaitu menggunakan kata “chaos” saja. Frekuensi karakter yang dihasilkan cukup identik, dan jumlah bits yang berubah juga mirip satu sama lain. Hal ini mungkin terjadi karena ketika kunci yang digunakan adalah “chaos”, kunci tersebut tetap diduplikasi hingga panjangnya serupa dengan panjang plainteks. Dengan memanfaatkan pergeseran 3x4 bits juga, maka karakter kunci tidak selalu digunakan untuk mengenkripsikan blok cipher yang sama, lantas karakter hasil enkripsi pun menjadi beragam. Selain itu, penggunaan s-box membuat hasil enkripsi memiliki karakter yang beragam pula.

Penulis kemudian mencoba beberapa kasus di mana kunci untuk mendekripsikan pesan adalah salah. Percobaan pertama adalah dengan mengubah satu karakter pada kunci yang panjang.

Kunci sebenarnya : “modderasijk12345lmntuvwxyz*aku8opqrsad67890atai)anak(toleyangyingnilmcukanapek.Kayung**panmanaaku_kudaaka-a+..ku^blah&anak…”
Kunci yang salah : (salah satu karakter di tengah kunci salah) “modderasijk12345lmntuvwxyz*aku8opqrsad67890atai)anak(toleyangyingnilmcukanapek.kayung**panmanaaku_kudaaka-a+..ku^blah&anak...”
Hasil: abcdefghijklmnopqrstuvwxyz1234567890 aku adalah anak tole yang tak pernah**ÃXÞ**kn Bilamana aku capek, maka aku bukanlah anak tole.

Seperti yang dapat dilihat pada hasil di atas, mode operasi cipher blok ini mewarisi sifat mode operasi Electronic Code Book(ECB) di mana ketika terjadi kesalahan dekripsi pada sebuah blok, hanya blok di sekitarnya saja yang terpengaruhi.

Penulis juga menemukan kejadian unik yang terjadi ketika kunci yang dimasukkan pengguna kurang panjang. Berikut adalah detail hasil yang didapat.

Kunci sebenarnya : “modderasijk12345lmntuvwxyz*aku8opqrsad67890atai)anak(toleyangyingnilmcukanapek.Kayung**panmanaaku_kudaaka-a+..ku^blah&anak…”
Kunci yang salah : (salah satu karakter di tengah kunci hilang) “modderasijk12345lmntuvwxyz*aku8opqrsad67890atai)anaktoleyangyingnilmcukanapek.Kayung**panmanaaku_kudaaka-a+..ku^blah&anak...”
Hasil: abcdefghijklmnopqrstuvwxyz1234567890 aku adala¿F|NLÁâÞºë÷8À^aÂàQàºhÓEÎ¤¯'ÒÄ"ZRVÞUObÑ Â*ùë¸icPn}GÒÔèZÏÃhY+

Pada kasus di atas, hal yang dapat diperhatikan adalah semua karakter plainteks yang berada setelah karakter kunci yang hilang gagal didekripsi. Hal ini disebabkan oleh kunci yang memiliki karakter kurang atau lebih membuat setiap karakter pada kunci bergeser, lantas membuat semua karakter sesudahnya menjadi gagal didekripsikan.

Kedua fenomena di atas akan tetapi tidak berlaku ketika kunci yang digunakan cukup pendek. Misalkan kunci yang digunakan adalah kata “chaos” dan terjadi perubahan karakter atau penyisipan/penambahan karakter, maka hasil dekripsi plain teks menjadi berantakan.

Hasil ketika kunci salah: (chaos menjadi chbos) “Ðºoru¸÷?¬Ï³ð áÒ¤ gÝÉq_?å-¤*ä¥iëö”
Hasil ketika ada penyisipan kunci: (chaos menjadi cha-os) “+ ÔÉÑ¢2¡Ñr,f Íz»ÇÆ54N ”

Ketika kunci yang digunakan sangat pendek dan memiliki beberapa kesalahan seperti pada contoh di atas, hasil dekripsinya pun mengalami kekacauan. Kunci yang pendek membuat kunci perlu diduplikasi dan menyebabkan kesalahan pada kunci meningkat. Hasilnya adalah plainteks gagal didekripsi.

V. Analisis Keamanan

Berdasarkan hasil dari eksperimen yang dilakukan oleh Penulis, blok cipher yang Penulis rancang memiliki keamanan yang kurang baik. Apabila digunakan kunci yang cukup panjang, hasil enkripsi dari blok yang sama akan berbeda-beda sehingga menyulitkan kriptanalisis. Tetapi apabila digunakan kunci panjang dengan 1 karakter yang salah, masih dapat diperoleh hasil dekripsi yang menyerupai plain teks asli. Hal itu disebabkan hasil dekripsi dari suatu blok hanya mempengaruhi blok sekitarnya. Namun jika kunci yang digunakan pendek, kemungkinan blok yang sama dienkripsi menjadi karakter yang sama menjadi lebih besar.

Apabila plainteks yang akan dienkripsi sangat panjang misalkan sebanyak 400 karakter, dan kunci hanya sepanjang 20 karakter misalnya, maka panjang kunci mudah ditebak ketika hanya satu karakter saja yang salah. Pola kesalahan pada hasil dekripsi akan terlihat dari jarak antara tiap kesalahan pada hasil dekripsi. Ketika kunci hanya sepanjang 5 karakter dan hanya satu karakter pada kunci yang salah, kesalahan pada dekripsi menjadi lebih sulit ditemukan. Namun sangat mudah untuk melakukan *brute-force* pada kunci yang pendek.

VI. Kesimpulan dan Saran

Algoritma TriTOLE memiliki beberapa kelebihan, yaitu frekuensi karakter hasil enkripsi terbagi merata. Selain itu, setiap karakter kunci yang salah akan mempengaruhi 5 hingga 6 karakter di sekitarnya, membuat proses kriptanalisis dengan cara menebak karakter kunci menjadi lebih susah. Kelebihan

lain adalah blok cipherteks yang dienkripsikan dengan kunci yang sama tidak selalu dienkripsikan ke dalam blok enkripsi yang sama pula.

Di lain sisi, algoritma ini masih memiliki banyak kelemahan, terutama ketika panjang kunci yang digunakan diketahui. Akan tetapi untuk menebak kunci algoritma ini, ada dua hal yang perlu diperhatikan. Pertama, ketika panjang kunci salah ditebak, karakter plainteks pada indeks kunci yang salah hingga akhir akan masih dalam bentuk teracak. Kedua, untuk menebak semua kunci secara benar perlu dilakukan perlahan-lahan. Caranya adalah dengan menebak karakter per karakter dari karakter paling depan hingga ke belakang sebab karakter yang ada di tengah dan akhir memiliki kemungkinan salah karena panjang kunci yang tak sesuai, atau karakter kunci yang digunakan salah.

Saran dari Penulis adalah untuk membuat algoritma juga menggunakan hasil enkripsi dari blok sebelumnya untuk mengubah kunci sehingga lebih susah untuk menebak kunci yang digunakan. Selain itu, jumlah bits yang digeser ada baiknya diturunkan dari hasil enkripsi blok sebelumnya, agar kesalahan pada sebuah karakter pada kunci tidak hanya memberikan efek pada blok sekitar plainteks yang dienkripsikan dengan karakter kunci tersebut, melainkan berpengaruh pula pada banyak karakter di sekitarnya. Saran lain adalah memodifikasi algoritma XOR yang digunakan Penulis sehingga menjadi lebih kompleks.

Daftar Referensi

[1] Rouse, Margaret. What is block cipher? January, 2006. http://searchsecurity.techtarget.com/definition/block-cipher. Diakses pada tanggal 23 Oktober 2016.

[2] Munir, Rinaldi. Slide Kuliah: Algoritma Kriptografi Modern(2015). http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/kriptografi.htm. Diakses pada tanggal 23 Oktober 2016.

[3] Wade Trappe dan Lawrence C. Washington. Diffusion and Confusion. Introduction to Cryptography and Coding Theory. http://www.nku.edu/~christensen/diffusionandconfusion. Diakses pada tanggal 25 Oktober 2016.

[4] Tsonka Baicheva, Dusan Bikov, Yuri Borissov dan kawan-kawan. Finding an Effective Metric Used for Bijective S-Box Generation by Genetic Algorithms. 27 September 2014. http://parallel.bas.bg/ESGI104/presentations/problem4.pdf. Diakses pada tanggal 25 Oktober 2016.

Lampiran

1. **S-Box**

x	y																
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
	0	5e	62	d	44	85	c9	4f	b6	d0	59	2d	52	ec	6d	9f	a2
	1	45	2a	5c	e7	d7	6b	e	9a	60	e6	4a	e9	b4	26	94	fd
	2	33	84	81	19	92	bd	fa	cf	7b	f8	78	79	1f	b	ff	5
	3	bc	23	d1	4b	c2	74	ab	98	8e	34	20	9	87	2f	27	a3
	4	86	fb	c	v0	89	de	8a	76	c8	64	17	99	e1	cd	4d	c4
	5	f7	3c	d2	38	c6	0	a	dd	b5	1b	69	b9	ac	82	80	bb
	6	65	b1	a6	e4	5f	5a	40	af	15	ce	3b	f1	83	71	6f	fc
	7	43	ef	25	9f	2e	8c	7f	29	b2	56	f0	7	cc	48	a9	d4
	8	70	2c	8f	3e	18	b8	75	d9	f4	11	35	db	1	3	a4	28
	9	58	e8	df	77	a7	7e	37	72	ba	5b	50	e0	13	f2	3f	5d
	a	1e	6a	8b	f3	c1	6	2b	a8	16	ae	b7	14	7c	ed	d6	ea
	b	c0	7a	dc	2	30	9e	55	d5	57	1c	6e	39	fe	ee	41	4
	c	53	22	da	7d	8	36	95	9c	24	54	66	46	8d	ad	9d	32
	d	42	21	be	10	88	31	d8	1d	4e	49	47	d3	63	b9	73	a1
	e	f6	97	f	51	e5	a0	e3	a5	96	eb	6c	ca	3a	3d	f5	c7
	f	aa	b3	68	90	67	91	c3	4c	e2	bf	12	1a	93	c5	cb	61