Calogerus Cipher Blok

Pengembangan Algoritma Cipher Blok dengan Matriks Substitusi Dinamis

Andarias Silvanus (13512022)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10, Bandung 40132, Indonesia
andarias.silvanus@gmail.com

Cilvia Sianora Putri (13512027)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10, Bandung 40132, Indonesia
cilva.sianora@gmail.com

Abstrak* — Makalah ini membahas mengenai rancangan algoritma *cipher* blok. Blok yang diambil berupa matriks berukuran 4x4 dengan tiap sel matriks berisi 8 bit dari plainteks yang sudah diubah ke bentuk biner. Matriks ini akan dibalik urutan baris dan kolomnya. Kunci yang dimasukkan harus sebesar 128 bit. Kunci ini akan dibagi per 16 bagian, yang bagian-bagian tersebut akan disubstitusi dengan berpatokan pada sebuah *substitution box*. Kunci baru tersebut akan dilakukan operasi XOR dengan kunci lama dan hasilnya akan disimpan sebagai baris pertama dari sebuah matriks substitusi. Baris-baris selanjutnya akan dibangun berdasarkan pengulangan langkah di atas. Hal ini dilakukan berulang-ulang sebanyak 16 kali. Selanjutnya blok awal yang berisi plainteks akan disubsitusi dengan mengacu pada matriks yang telah dibangun dan digeser per baris dan kolomnya. Setelah itu dilakukan operasi XOR terhadap blok tersebut dengan baris pertama dan baris terakhir dari matriks substitusi yang telah dibangun. Terdapat 3 mode operasi untuk algoritma ini, yaitu mode *Electronic Code Book, Chain Block Cipher,* dan *Cipher Feedback.

Kata kunci — cipher blok, enkripsi, dekripsi, substitusi, CBC, ECB, CFB

I. PENDAHULUAN

Kriptografi merupakan seni sekaligus teknik pengelabuan pesan untuk menyembunyikan informasi atau pesan terhadap pihak yang tidak diinginkan. Dalam kriptografi terdapat dua teknik yang sering dipakai dalam pengembangan algoritma, yaitu teknik transposisi yang merupakan pengacakan urutan letak elemen dan teknik substitusi yang merupakan substitusi elemen dengan elemen lainnya. Awalnya, dalam perkembangan kriptografi, algoritma-algoritma yang telah dikembangkan memakai aplikasi dari ilmu matematika. Tapi seiring berkembangnya teknologi, kekuatan algoritma dalam kriptografi pun semakin meningkat dengan menerapkan ilmu komputer.

Pada masa ini, kriptografi memasuki era barunya yang disebut era kriptografi modern, dimana algoritma-algoritma yang dikembangkan memainkan dan mengolah bit dari pesan yang hendak dienkripsi. Semakin banyaknya penggunaan komputer digital merupakan salah satu faktor yang mendorong terjadinya perkembangan kriptografi untuk menjaga kerahasiaan informasi digital. Sekalipun begitu, teknik substitusi dan transposisi yang digagas dalam era kriptografi klasik masih tetap dipakai.

Dalam makalah ini, kami menawarkan rancangan algoritma yang mengolah blok bit dengan teknik yang kami rancang serumit mungkin.

II. DASAR TEORI

Terdapat dua jenis kategori dalam algoritma kriptografi modern, yaitu kategori cipher aliran dan cipher blok.

A. Algoritma cipher aliran

Algoritma ini mengenkripsi plainteks menjadi cipherteks dengan cara menelusurinya secara bit per bit ataupun *byte* per *byte* dengan kunci *keystream*. Kunci *keystream* berbentuk bit yang dibangkitkan dengan pembangkit *keystream.* Ide dari algoritma ini mirip dengan algoritma *one-time-pad.* Setiap bit ataupun *byte* dari plainteks akan dilakukan dengan operasi XOR terhadap bit atau *byte* dari kunci *keystream.* Cipherteks akan dihasilkan berdasarkan operasi sebagai berikut.

$$c_i = p_i \oplus k_i$$

Dan untuk mendekripsinya, penerima harus membangkitkan kunci *keystream* yang sama dan dilakukan operasi XOR terhadap cipherteks tersebut. Plainteks akan dihasilkan berdasarkan operasi berikut.

$$p_i = c_i \oplus k_i$$

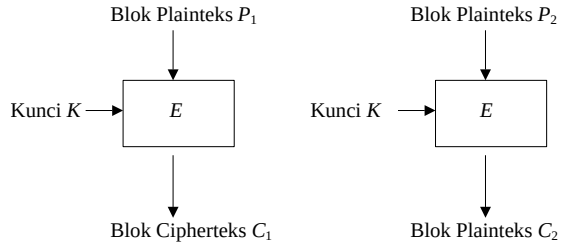
Dengan c_i = cipherteks
p_i = plainteks
k_i = kunci keystream

B. Algoritma cipher blok

Algoritma ini membagi bit plainteks menjadi blok-blok bit yang memiliki ukuran yang sama. Selanjutnya ukuran kunci pun harus disesuaikan dengan ukuran blok. Enkripsi akan dilakukan terhadap blok bit plainteks berdasarkan teknik masing-masing algoritma dengan menggunakan bit kunci. Pada akhirnya, blok cipherteks akan dihasilkan dan memiliki ukuran yang sama dengan blok plainteks. Algoritma cipher blok sendiri memiliki beberapa mode operasi yang berkaitan dengan cara blok tersebut akan dioperasikan.

1. *Electronic Code Book (ECB)*

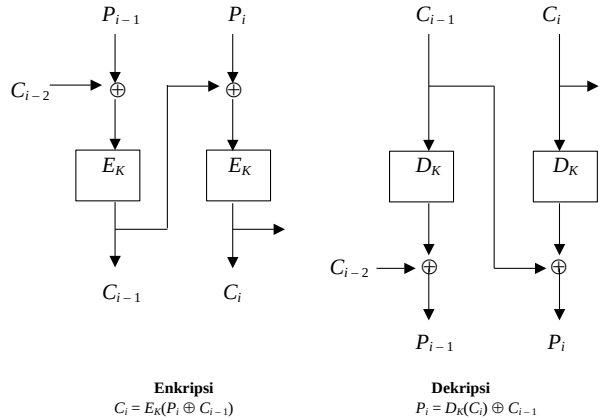
   Ide dari mode ini adalah setiap blok plainteks akan dienkripsi secara individual dan independen menjadi blok cipherteks. Oleh karena itu dekripsinya juga akan dilakukan secara individual dan independen.



Gambar II-1 Ilustrasi enkripsi mode ECB. Sumber: http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi

2. *Cipher Block Chaining (CBC)*

   Ide dari mode ini adalah membuat adanya ketergantungan antar blok yang akan dienkripsi. Setiap blok akan bergantung pada seluruh blok plainteks sebelumnya. Setelah suatu blok selesai dienkripsi, blok cipher-nya akan dipakai lagi dalam proses enkripsi blok plainteks selanjutnya.



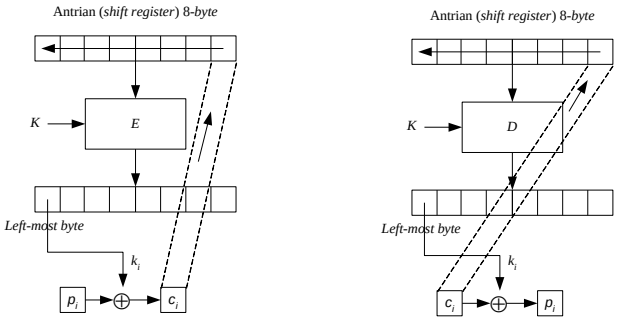
Gambar II-2 Ilustrasi enkripsi dan dekripsi mode CBC. Sumber: http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi

Seperti yang terlihat pada gambar, pada proses enkripsi dibutuhkan blok pertama yang harus dienkripsi terlebih dahulu. Blok ini disebut dengan *Initialization Vector* (*IV*). *IV* dapat dibangkitkan secara acak maupun dibangkitkan berdasarkan masukan dari pengguna. Dalam proses dekripsi akan dilakukan operasi XOR antara *IV* dengan hasil dekripsi terhadap blok cipherteks pertama. Proses ini akan menghasilkan blok plainteks.

3. *Cipher-Feedback (CFB)*

   Mode ini memiliki kemiripan dengan mode CBC. Namun mode ini membebaskan jumlah bit yang hendak dienkripsi. Dan proses enkripsi yang dilakukan mirip seperti algoritma cipher aliran.

   Bangkitkan *Initialization Vector* secara acak maupun dengan masukan dari user yang akan dimasukkan ke dalam sebuah antrian (*queue*). Selanjutnya enkripsi dilakukan antara antrian dengan kunci. Hasil enkripsi akan berlaku sebagai *keystream* yang kemudian akan dilakukan operasi XOR dengan blok plainteks yang berukuran sejumlah bit yang didefinisikan di awal. Hasil ini kemudian akan dimasukkan ke dalam antrian dengan menggeser 1 blok antrian dari blok terkanan. Untuk proses dekripsinya hanya perlu membalikkan langkah-langkah enkripsi.



Gambar II-3 Ilustrasi enkripsi dan dekripsi mode CFB. Sumber: http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi

Secara formal, mode CFB dapat dinyatakan sebagai berikut untuk proses enkripsi:

$$C_i = P_i \oplus \text{MSB}_m(E_k(X_i))$$

$$X_{i+1} = \text{LSB}_{m-n}(X_i) \,||\, C_i$$

Dan notasi formal mode CFB untuk proses dekripsi:

$$P_i = C_i \oplus \text{MSB}_m(E_k(X_i))$$

$$X_{i+1} = \text{LSB}_{m-n}(X_i) \,||\, C_i$$

Dengan penjelasan,

X_i = isi antrian dengan X_1 adalah *IV*
E = fungsi enkripsi dengan algoritma *cipher* blok
K = kunci
m = panjang blok enkripsi
n = panjang unit enkripsi
|| = operator penyambungan (*concatenation*)
MSB = *Most Significant Byte*
LSB = *Least Significant Byte*

III. RANCANGAN ALGORITMA

Algoritma yang kami rancang berlaku untuk kunci 128 bit dan memiliki tiga mode operasi, yaitu mode ECB, CBC, dan CFB.

Awalnya, plainteks akan diubah ke dalam bentuk biner dan dibagi per 8 bit sesuai ukuran untuk 1 *cell* blok plainteks. Nantinya, akan diambil sebanyak 16 blok dengan dibentuk sebagai matriks dua dimensi berukuran 4x4 blok. Sebut saja blok ini sebagai blok P. Setiap *cell* yang berada pada kolom genap akan diisi dengan blok plainteks yang berada di indeks terakhir dan akan terus diiterasi menurun. Sementara setiap *cell* yang berada pada kolom ganjil akan diisi dengan blok plainteks yang berada pada indeks pertama dan yang akan terus diiterasi

menaik. Jika tidak lagi tersedia 16 blok plainteks, maka *cell* tersisa akan diisi dengan *padding* yang berupa rangkaian biner "00000000". Setelah blok P terisi, setiap barisnya akan diubah menjadi kolom.

Isi dari kotak fungsi enkripsi dimulai dari sini. Kunci yang dimasukkan pengguna akan dibagi per 16 bagian dengan masing-masing bagiannya berisi 8 bit data.

Key (K)	8 bit	8 bit	8 bit	8 bit	8 bit	8 bit	…	8 bit

Tabel III-1 Ilustrasi pembagian kunci menjadi 16 bagian

Selanjutnya kunci tersebut akan diganti urutan letaknya dengan sebuah *substitution box* seperti yang tertera di bawah ini dan menjadi K'.

Sbox	14	8	2	10	4	9	5	15	3	1	11	6	13	7	12	0

Tabel III-2 *Substitution Box*

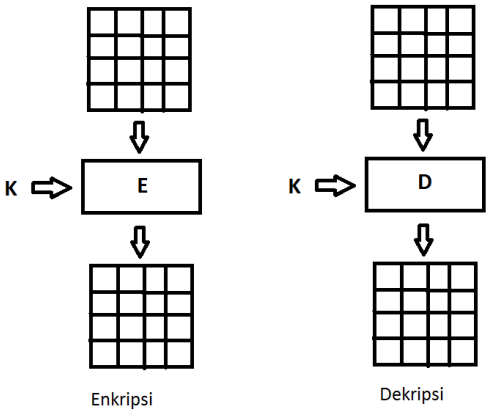
Operasi XOR dilakukan antar K dengan K' dan hasilnya akan menjadi baris pertama dari matriks substitusi S. Baris pertama itu diacak urutannya dengan berpatokan *substitution box.* Untuk mendapatkan baris selanjutnya dari matriks S, baris pertama yang sudah diacak urutannya dilakukan operasi XOR dengan K'. Hal ini dilakukan secara terus menerus untuk mengisi matriks S.

Matriks substitusi S akan terbentuk dengan ukuran 16x16. Matriks S ini sendiri memiliki indeks baris dan kolom hexadesimal, yang *range*-nya terdiri dari 0-f. Ambil blok P dan ambil setiap 4 bit dari *cell* blok P. Empat bit ini akan dikonversi ke dalam bentuk hexadesimal, sehingga kita akan mendapatkan 2 angka hexadesimal. Kedua angka ini akan digunakan sebagai koordinat baris dan kolom dari matriks S. Angka pertama sebagai penunjuk indeks kolom dan angka kedua sebagai penunjuk indeks baris. Isi dari *cell* blok P akan disubstitusi dengan isi dari *cell* matriks S. Hal ini terus dilakukan selama setiap *cell* blok P sudah tersubstitusi dengan matriks S.

Setelah itu, akan dilakukan transposisi terhadap baris dan kolom dari blok P. Baris pertama blok P digeser sejauh 1 kotak ke arah kanan, baris kedua blok P digeser sejauh 2 kotak ke arah kiri, dan baris ketiga blok P digeser sejauh 3 kotak ke arah kanan. Lalu kolom pertama blok P digeser sejauh 1 kotak ke arah bawah, kolom kedua blok P digeser sejauh 2 kotak ke arah atas, kolom ketiga blok P digeser sejauh 3 kotak ke arah bawah.

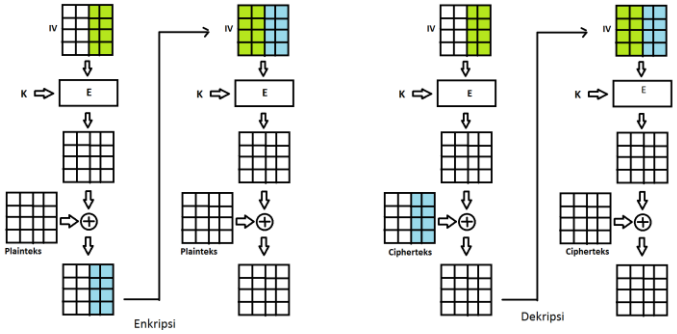
Proses di atas, yang dimulai dari pembangkitan matriks S, dilakukan sebanyak 16 kali. Lalu ambil baris pertama pada matriks S terakhir (yang dihasilkan dari iterasi ke 16) yang dan lakukan operasi XOR dengan blok P. Hasilnya akan dilakukan operasi XOR dengan baris terakhir pada matriks S.

Dalam mode ECB, enkripsi terhadap blok P dilakukan secara independen. Proses dilakukan sampai plainteks habis dienkripsi. Sementara dalam mode CBC, dilakukan operasi XOR terhadap blok P dengan *Initialization Vector* (*IV*) terlebih dahulu. Lalu dilakukan proses enkripsi terhadap blok P. Blok cipherteks yang dihasilkan akan dilakukan XOR dengan blok P selanjutnya. Selanjutnya proses ini akan dilakukan berulang-ulang hingga plainteks telah diproses semua.



Gambar III-1 Skema enkripsi dan dekripsi dengan mode ECB

Kami mengambil jumlah n sebanyak 128 bit dalam mode CFB. Antrian awal akan diisi dengan *Initialization Vector (IV)* yang berupa matriks 4x4 dengan isi setiap *cell*-nya berupa data 8 bit. *IV* akan dibangkitkan dengan menggunakan *seed* sebagai pembangkit bilangan acak dari 0 sampai 255. *IV* dan kunci K akan dimasukkan ke dalam fungsi enkripsi. Hasilnya akan dilakukan operasi XOR dengan blok plainteks yang berukuran 128 bit. Kolom-kolom antrian yang berupa matriks 4x4 akan digeser sejauh 2 kolom ke arah kiri. Sementara untuk pengisian kolom daerah kanan antrian akan diambil dari 2 kolom terkiri dari blok hasil dari operasi XOR di atas. Blok hasil ini akan disimpan sebagai blok cipherteks. Antrian baru yang sudah tercipta akan digunakan untuk proses enkripsi selanjutnya.



Gambar III-2 Skema enkripsi dan dekripsi dengan mode CFB

IV. Eksperimen dan Pembahasan Hasil

Kami mencoba mengenkripsi 3 jenis sampel dalam format teks. Ketiga sampel tersebut berukuran 1734 byte, 4110 byte, dan 32880 byte. Kami juga mencoba membandingkannya dengan algoritma Rijndael. Hal yang kami ukur adalah dari segi waktu yang kami ukur dalam milisecond. Berikut adalah tabel perbandingan waktu antara algoritma Rijndael dengan algoritma yang kami kembangkan dengan mode ECB, CBC, dan CFB.

	Ukuran File (byte)		
Mode	1734	4110	32880
ECB (ms)	498.854	1406.090	56135.46
CBC (ms)	465.640	1414.339	31117.46

CFB (ms)	454.612	1342.644	56231.86
Rijndael (ms)	13.472	22.030	90.171

Tabel IV-1 Tabel perbandingan waktu enkripsi

	Ukuran File (byte)		
Mode	1734	4110	32880
ECB (ms)	528.319	817.854	4831.738
CBC (ms)	468.199	819.464	4921.495
CFB (ms)	20.8504	52.510	434.574
Rijndael (ms)	18.158	49.821	161.636

Tabel IV-2 Tabel perbandingan waktu dekripsi

Berikut adalah salah satu sampel yang kami pakai yang berukuran 1734 byte.

Why does this happen? Well, the FAT32 file system needs to keep track of where each file is stored. If it were to keep a list of every single byte, the table (like an address book) would grow at the same speed as the data – and waste a lot of space. So what they do is use “allocation units”, also known as the “cluster size”. The volume is divided into these allocation units, and as far as the file system is concerned, they cannot be subdivided – those are the smallest blocks it can address. Much like you have a house number, but your postman doesn’t care how many bedrooms you have or who lives in them.

So what happens if you have a very small file? Well, the file system doesn’t care if the file is 0 KB, 2 KB, or even 15 KB, it’ll give it the least space it can – in the example above, that’s 32 KB. Your file is only using a small amount of this space, and the rest is basically wasted, but still belongs to the file – much like a bedroom you leave unoccupied.

Why are there different allocation unit sizes? Well, it becomes a trade-off between having a bigger table (address book, e.g. saying John owns a house at 123 Fake Street, 124 Fake Street, 666 Satan Lane, etc.), or more wasted space in each unit (house). If you have larger files, it makes more sense to use larger allocation units – because a file doesn’t get a new unit (house) until all others are filled up. If you have lots of small files, well, you’re going to have a big table (address book) anyway, so may as well give them small units (houses).

Large allocation units, as a general rule, will waste a lot of space if you have lots of small files. There usually isn’t a good reason to go above 4 KB for general use.

Kami mengenkripsi dengan kunci “ABCDEFGHIJKLMNOP”. Dan berikut adalah hasil enkripsi dari mode *Electronic Code Book* (*EBC*).

çù,ë-
³©]¢àÌX›Û□þYæ-Î£œà]T□Ûx□ÍZ"YQþªVú • *!U+öýèt–
Ë□ÖëVM†

á ëXJZPYÔ□N□mù£û,,□˜ýÅ□/§Q×Ù□ú
¨áUé¦› Hf&Ôä¢0MÉó%«^ïjñHåîÕã)‡Mâ□Ñ« Ï _H
YæÄÄ_àÇÑUjâŽ¶åîæ \Dúw]Tà&ïyÍ™®y AéDúW

…UlX´ÍÍåÜ□-
á WaT,,~XØ¡H K L\VmÓOãu!XJ÷ÿ • îc \¨È"]ÑG^ï 'H , Ôä\VœâŒ□U^ïýÈÍ • Y¯ÖV_/èTàë¦j • 0UÝQ \ã³à T éïJcý Y□äšøÉàêt ðíJª™®ËQÖëVM†XÑ«éXÏZh□YÿärøøW

Ñu□íÏ™™ ÷c,\ãœâŒtUÉXýè½ • § -\þ‹õ++«!ïý□¶ ! S x Å] «
.ÏÈh□ÿ□$ãMWèá,,!-;_¶!îQÁ\Ä³àØ…UÛÝý_t¥,q-ëãúWè …â/X2èH□Ü¯SÎÒúWŒ+$^¦J□PY¯,:ÒÓRŒT-
!ÝÎ□æPY□,$Áœ!

]□^öŒ□H®Y¯,é□□à"ª*ÛX›èfç§QS)□œâŒÑÐ^JJ□½–
qì\‡5W]fãéXv÷™PËÃÖëÓÉ]Æ • XäÚh¥§ s\á WTc!XÔ .
S Í□VøÓXªàé>ÿ_ÿPy)Ö¨ãRêáà‰jmZå§QÖ:‡Óâ,T«□xJñZ
P§ æÓM©«à[‰J_ÿ • Ymý • V³Å Ã ðÎ HO?CSÂþGW(T
ÛïJÍ7åîæÖvMâOfUë¦ý PÇ€ 5£ å©êªÇ~ïÏRåeQS
‡øâ]TàÄïjÍÍ□§ÿs\2Wê «/X è-
~&Ãä\ÓmW}tûb' □HçË)ÖëV³ùŒúUÛXÏ□ÿP¤æn`œÒÃã3^í
äŽ¶ æSòVm(êÑà~öj Z□÷Q,#áMW£Ñ□ X›Zz–
æ,_œWÃ…3 JJ_¶pYÔäN-
mù]ªû^Ê Úh□§Ô×Ù□úeèáàé¦›_Hçk□S□ã¢Œ%à^¦J□ÿ□&¯ä
Y MWØfUéöý ¶çkÿS□Vã©êÑ□ÌÜ□è9®îQ•\Á³àŒú$
XJÚÍPyqo#¨úRê□à˜xjZYQZÙ£ßà+„^Æÿ9Hlæ,ëÄM†]□□
X èÌB(Q-• VøHõÑ#ÌXýè¶ Ÿ□Séx□Å]áÐéÝj¡æPÜ s\d • Å
ŒfâÉXJ¡æP&q,dúàŒ…ð‰´_Hç□.?n¨ßÒŒª-
XJ÷ÿ Y äØV &T ëÏJ • Møî'o\Óœeê□U^‰OÍÿ • îÿ-\Vãõó
T„Ì□Ï□™P\k \˜‹™“□U/ïO™Íç□æ□ÓM g□UÛÜý_ßÿ˚€ "ò
!MW ! ÝJ¡Mç?ÔýÂ {àØ+o¦Jèÿ • îÆ \Vßø□Ñ^‰□mH□
÷Ô,ØáMW£□□&)□_Ú • Q -¨³Wó'-é HÂËÿS • gq
]T Ú ä h • !ÕSÎV ©õã^ â ÿU§Qä\ mÀ

T“ëdJ • ¶ç(¯äô)w "ú„^ÊdÚHåÜæ,\‡ß‘Ø□U^Ê□'ÿ¥§¤ý\ •
mw]…„+ïJ*½pYÿ□Î‡œkÏFU^□äÛÿ(eæSëá□Ò›hUé‰X_Zyì
□,é)MàŒÐ□^)Ï¡HPY€ þªáú©,F ~ïxêÿPË—
ÖëvMÒ]áPëXÿZý¦§æSØû³W£Fâ~ïGêhP¯,ëv³ÒóÑ éXJ.æpY
□Ön‡œÒoú□8nÿÁýPY€ AÙøßÒê éxý_ZpîCæ\Vm(êã ^x
ýMÚP+Ô…Y□MàŒªU^XÎ□ÚøY§Vá□(}□ ^*□‰z Ycý -
ù]Tâéïvc™åyæSÙ+WTÑˆ]J-
9 ¬ S|)q êÊ“híJê™çÜQSÎvúÀèãU^ÝØ • z!Ü -_³WÃT3ë
‰Jèÿpyä SÙVMEê!Ç • ïÏmHÕÇ 5U¨MW¨fû • X ñý–
îæÁ\ãúWètuðÝÏ_¶ÕîQæ\□œ-]ïP • -5.ÿ®YÔVÙ.ß†]’Ñ
ïÿcýb§€ e\¨□wŒ…“ÛÆJ9¶PYqûÁ¨□Òê□□^*J‰ÿÕìÃsë)
MWó□„JJ_Mbìæý Ø#M†£tU ´_hÕYÃ¦ • DMW U öÏ • æ -
□æ(‡9&]ª!/í>Ž <

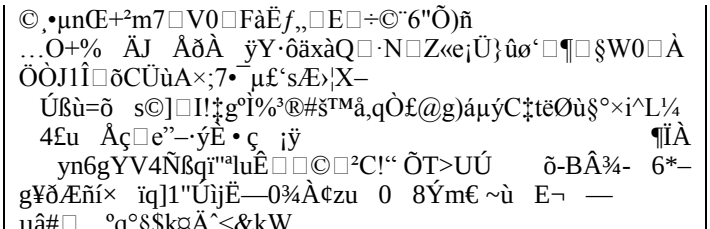
Berikut adalah hasil enkripsi dari algoritma Rijndael.

¶A‰Ê4Õ•Ê„J-*f*<O¯4ày/9□™ÖÚ~„*□b

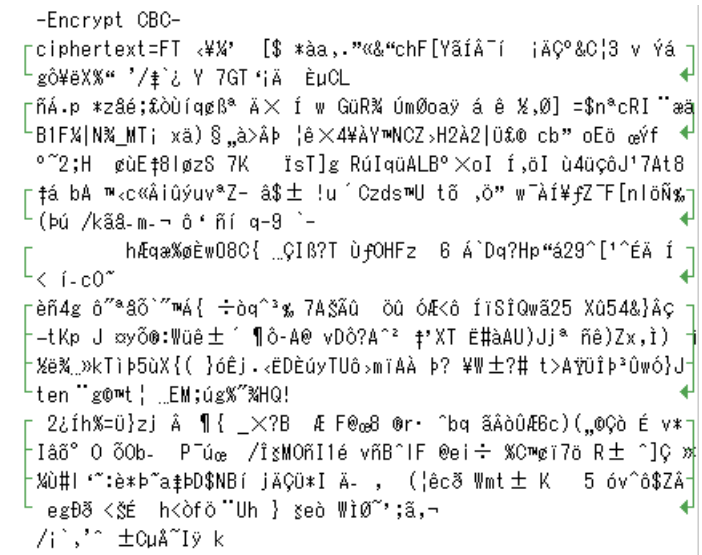
,□□.Ê¹□Ù?ß™<Ô/□‰Rk{3Å½ªjvù□³Œ$4šífb;_Pù 7Ùðˆ
üÔê ±3m□°¿qÐFßÚœË‹Yü□ÙH|™†sB¼

ÅGÂ3µ«¶´AÚå,;NÒ)B-¶h;:„£1Î€ ,¼Êb,û2_Á • ^ÜÁwž ´
C?³XS¼‹Ð*f*;sÆXOISÄ:«_ý×à’,%□O□” J□@›¶_[ùÙ□„¡ï-øD
ë "³□□›¢Ë)□|š-
›Í|Û«‹ð□@‰□□□□Uæ§:!Ã…KˆÚ¨ª‰¾B}Q]

j5’B□□¤PŒL:„-□□*□EG&Óa½²□®]□‘Ô□ÈC□PÊ´Æ’
Ûdá□’t/n□6|-
□ìlùjÿ§«Üøñ´qÍ_·û_˜t-□Âb8•ªPãèÔÆ×pC□H-¡çœC™$¨ö,a
¾□)ðRI½‹UÙk‹Ó±.*Hk,½$¨•¶©Ü÷ÖEü#□í™}ÝeQÙ2mÒ□
©E□[□ÊŸ□!QûeðÂÎˆ‰bâEt…Áâø‰kšFˆË□JÉ±ÜäT×ð1”e
—5mAx€ m5 …□□È”ùz#‡^Ð—
□ïÏ…¾□□Ú^Á□ÚWkà□_□Ùì`¤²§üJAZÁéaO\òZFTïð\o+(R



Berikut adalah hasil enkripsi dari mode *Chain Block Cipher* (*CBC*).



Berikut adalah hasil enkripsi dari mode *Cipher Feedback* (*CFB*).

Hasil enkripsi kami simpan dalam file teks. Sementara hasil dekripsi dari semua mode memberikan hasil yang sama dengan plainteks.

V. ANALISIS

Jika dibandingkan dengan algoritma Rijndael, perbedaan waktu yang dihasilkan jauh berbeda. Namun hal ini dikarenakan adanya pembangkitan matriks substitusi berkali-kali. Sementara dalam algoritma Rijndael, matriks substitusi yang digunakan bersifat konstan.

VI. ANALISIS KEAMANAN

Karena panjang kunci dalam algoritma kami sepanjang 128 bit, maka akan terdapat sebanyak 2^{128} kemungkinan kunci. Bila saat ini komputer sudah dapat mencoba 1 juta kunci tiap detiknya, maka dibutuhkan waktu sekitar 5.4 x 10^{24} tahun untuk mencoba semua kemungkinan kunci.

VII. KESIMPULAN DAN SARAN

Berdasarkan percobaan yang kami lakukan, kami mendapat kesimpulan jika dekripsi yang dilakukan dengan mode CFB dapat 10 hingga 25 kali lebih cepat dibandingkan dekripsi yang dilakukan dengan mode ECB dan CBC. Sementara enkripsi yang dilakukan dengan mode CFB lebih cepat dibanding mode ECB dan CBC meskipun tidak menunjukkan perbedaan yang signifikan.

Berdasarkan hal ini, akan lebih baik jika algoritma ini dikembangkan lebih lanjut dengan mode CFB sebagai mode yang memiliki performa paling unggul di antara 2 mode lainnya.

Dan jika dibandingkan dengan algoritma Rijndael, algoritma yang kami rancang memiliki kerumitan yang lebih kompleks dibandingkan Rijndael dalam hal pembuatan *substitute box.*

REFERENSI

[1] Rinaldi Munir, 17 Maret 2015, 14:32, Tersedia dari http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/