Algoritma Chess Cipher

Nugroho Satriyanto - 13514038[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*nugroho.s@outlook.co.id*

Amal Qurany – 13514078[2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[2]mal.qurany@gmail.com

***Abstract*—Block *cipher* adalah salah satu algoritma kriptografi modern yang dilakukan dengan membagi data ke dalam blok dengan ukuran tertentu. Block *cipher* yang dikembangkan dalam makalah ini adalah block *cipher* kunci simetris yang dikembangkan berdasarkan permainan catur. Algoritma ini menerapkan prinsip confusion dan diffusion sehingga algoritma ini cukup sulit untuk diserang.**

***Keywords—cipher*, blok,s-box,catur**

I. PENDAHULUAN

Pertukaran informasi adalah konsep utama dari internet. Suatu data dari seseorang dapat ditransmisikan dengan mudah melalui internet. Salah satu tantangan dalam pentransmisian data adalah mentransmisikan pada pihak yang tepat tanpa dapat dicuri atau dimanfaatkan oleh pihak lain. Salah satu cara untuk menjawab tantangan tersebut adalah kriptografi.

Kriptografi adalah suatu metode untuk menyimpan dan mengirimkan data dalam suatu bentuk tertentu sehingga data tersebut hanya dapat dibaca atau diproses oleh pihak yang dimaksudkan untuk menerimanya [1]. Dengan menggunakan kriptografi, pihak ketiga yang ingin mengetahui isi pesan tidak dapat membacanya dikarenakan kunci untuk mendekripsi pesan tidak diketahui. Salah satu tipe kriptografi adalah block *cipher* yang merupakan suatu algoritma kriptografi yang diaplikasikan pada suatu blok data dengan ukuran tertentu [2].

Algoritma kriptografi, selain menerapkan konsep matematika, terkadang menerapkan konsep dari kegiatan sehari-hari atau konsep dari bidang ilmu lain. Pada makalah ini penulis mengusulkan sebuah algoritma block *cipher* baru "Chess *Cipher*" yang menggunakan konsep permainan catur untuk proses enkripsi dan dekripsinya.

II. DASAR TEORI

A. Block Cipher

Block *cipher* adalah metode kriptografi yang beroperasi pada blok dalam ukuran tertentu [2]. Saat ini telah banyak algoritma block *cipher* yang dikembangkan, meliputi DES, AES, RC5, dan sebagainya [3]. Beberapa diantaranya menggunakan prinsip jaringan feistel yang membuat proses enkripsi dan dekripsi serupa.

Banyak variasi proses yang dapat diterapkan untuk enkripsi dan dekripsi pada block *cipher* yang umumnya dikelompokkan menjadi kategori permutasi dan subtitusi. Proses permutasi adalah proses penukaran bit pada lokasi tertentu dengan bit pada lokasi lain. Sementara proses subtitusi adalah penukaran nilai menjadi nilai lain.

B. Prinsip Confusion

Confusion adalah teknik yang digunakan pada kriptografi untuk memastikan *cipher* teks tidak dapat menurunkan *plain* teks atau kunci. Efek confusion dapat diperoleh dari subtitusi atau pengacakan yang kompleks [4].

C. Prinsip Diffusion

Difusi adalah Teknik yang digunakan pada kriptografi untuk mengaburkan struktur statistik *cipher* teks yang dihasilkan untuk mencegah kriptanalis menebak key yang digunakan. Efek diffusion dapat diperoleh dari permutasi yang menghasilkan bit dari lokasi lain berpengaruh terhadap perubahan nilai bit di suatu lokasi [4].

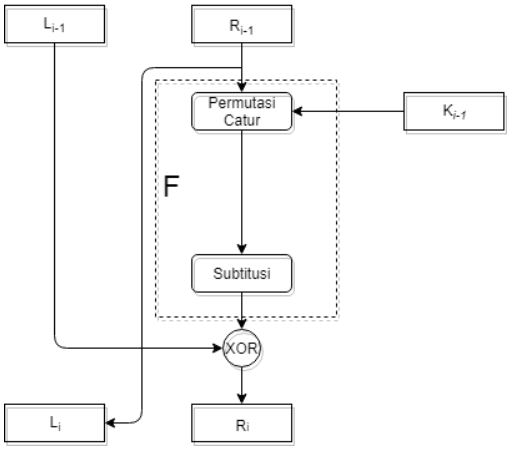
D. S-Box

S-Box adalah salah satu cara untuk mensubtitusi satu nilai menjadi nilai lain. Nilai akhir dari subtitusi adalah hasil dari pencarian nilai awal dalam suatu daftar nilai yang disebut S-box. Nilai S-box dapat dihasilkan dengan berbagai cara dengan ketentuan nilai di setiap kotaknya harus unik dalam *S-box*.

III. RANCANGAN BLOCK *CIPHER*

A. Diagram Feistel

Algoritma chess *cipher* menggunakan jaringan feistel yang membagi dua bagian 16x8 menjadi dua buah 8x8 yang dapat dianggap sebagai papan catur. Sub blok 8x8 kemudian masuk ke dalam fungsi F baru berupa permutasi gerakan catur dan subtitusi byte. Fungsi F digunakan untuk mengubah salah satu sub blok bergantung pada sub blok lain secara bergantian.



Gambar 1 Diagram feistel pada satu round chess cipher

Baik proses permutasi maupun subtitusi yang digunakan memiliki fungsi enkripsi yang mirip dengan dekripsi sehingga chess *cipher* tidak memerlukan algoritma baru dalam proses dekripsinya.

B. Permutasi Gerakan Catur

Penerapan operasi permutasi dalam chess *cipher* menggunakan prinsip gerakan pada permainan catur dengan beberapa penyesuaian. Suatu blok *cipher* yang berukuran 64 bit direpresentasikan sebagai matriks bit dengan ukuran 8x8 yang analog dengan papan catur. Gerakan buah catur dari kotak A ke kotak B akan menyebabkan bit yang berada pada kotak A bertukar posisi dengan bit pada kotak B sehingga pada akhir proses permutasi akan dihasilkan isi blok *cipher* teracak sesuai pola gerakan catur.

Suatu gerakan pada buah catur tidak diambil dari kuncinya secara langsung melainkan akan diturunkan dari kunci yang disediakan menggunakan suatu fungsi tertentu, sehingga nantinya dapat dihasilkan banyak gerakan catur dan menyebabkan kesalahan pada key akan mengakibatkan kesalahan merambat pada hasil dekripsi dan menghasilkan hasil dekripsi yang jauh berbeda dengan *plain* teks awal. Gerakan catur akan direpresetasikan sebagai 4 bit angka yang menentukan satu diantara 16 buah catur yang akan digerakkan dan 6 bit yang menentukan lokasi tujuan buah catur tersebut.

Block awal	1	0	1	1	1	0	1	0
	0	0	1	0	0	0	1	1
	1	0	1	0	1	0	0	0
	1	1	1	0	0	1	0	0
	0	0	1	0	0	0	0	0
	1	0	0	1	0	0	0	1
	1	1	0	1	1	0	0	0
	0	0	0	0	1	1	1	0
Gerakan catur	♜		♝	♛	♚	♝	♞	♜
	♟	♟	♟	♟	♟	♟	♟	♟
	♞							
		♙						
	♙		♙	♙	♙	♙	♙	♙
	♖	♘	♗	♕	♔	♗	♘	♖
Block hasil	1	1	1	1	1	0	1	0
	0	0	1	0	0	0	1	1
	0	0	1	0	1	0	0	0
	1	1	1	0	0	1	0	0
	0	1	1	0	0	0	0	0
	1	0	0	1	0	0	0	1
	1	0	0	1	1	0	0	0
	0	0	0	0	1	1	1	0

Gambar 2 Perpindahan pion putih dan kuda hitam mengakibatkan bit berkesuaian berpindah tempat

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

Gambar 3 Pengkodean kotak tujuan tujuan

♙	♙	♙	♙	♙	♙	♙	♙
0	1	2	3	4	5	6	7
♖	♘	♗	♕	♔	♗	♘	♖
8	9	10	11	12	13	14	15

Gambar 4 Pengkodean buah catur

Suatu gerakan catur yang diturunkan mungkin tidak valid seperti gerakan mundur pada pion, oleh karena itu perlu ada penyesuaian lokasi tujuan. Penyesuaian lokasi tujuan dilakukan dengan memilih satu lokasi tujuan valid yang memiliki jarak euclidean terkecil dengan lokasi tujuan turunan yang tidak valid. Prinsip "memakan" pada catur juga disesuaikan agar tidak ada buah catur yang hilang dan menyebabkan suatu buah yang diturunkan tidak valid. Prinsip "memakan pada catur disesuaikan sehingga ketika buah catur A memakan buah catur B buah catur A dan B bertukar posisi.

♜	♞	♝	♛	♚	♝	♞	♜
♟	♟		♟	♟	♟	♟	♟
		♟					
	♙						
♙		♙	♙	♙	♙	♙	♙
♖	♘	♗	♕	♔	♗	♘	♖

Gambar 5 contoh gerakan tidak valid (merah) pada pion di b4, gerakan valid yang mungkin (hijau), dan gerakan valid yang diambil (hijau tua)

♜	♞	♝	♛	♚	♝	♞	♜
♟	♟		♟	♟	♟	♟	♟
		♙					
	♟						
♙		♙	♙	♙	♙	♙	♙
♖	♘	♗	♕	♔	♗	♘	♖

Gambar 6 proses makan mengakibatkan 2 pion bertukar posisi

Dekripsi proses permutasi gerakan catur dilakukan dengan prinsip jaringan Feistel untuk meminimalkan perubahan algoritma dengan algoritma enkripsi. Untuk melakukan dekripsi dilakukan dengan menjalankan proses permutasi seperti pada enkripsi tanpa menukar tempat blok *cipher*. Pada saat proses permutasi dijalankan, tempat yang dilakukan penukaran dicatat dan kemudian penukaran dilakukan secara terbalik dari

belakang. Dengan proses ini, setelah proses penukaran selesai akan dihasilkan teks yang sama dengan *plain*text sebelum proses permutasi.

Proses enkripsi dan dekripsi pada permutasi gerakan catur dapat diubah menjadi bergantung pada blok *cipher* sebelumnya, sehingga kesalahan pada *plain* teks atau *cipher* teks akan merambat ke blok lain. Perubahan ini dilakukan dengan pengubahan key untuk proses permutasi dengan blok sebelumnya.

C. Subtitusi Knight's Tour

Selain permutasi gerakan catur, proses subtitusi juga dibuat dengan mengambil konsep permainan catur. Proses subtitusi dilakukan dengan S-box yang disusun sesuai dengan persoalan knight tour pada papan catur berukuran 16x16. Knight's tour adalah persoalan untuk mencari lintasan kuda pada papan catur yang dapat mengunjungi masing-masing kotak pada papan catur tepat satu kali.

Adapun untuk mengubah lintasan kuda menjadi S-box dilakukan dengan menandai tiap kotak dengan urutan kotak tersebut dikunjungi oleh kuda. Sehingga angka satu pada S-box merupakan lokasi awal kuda dan posisi angka selanjutnya menandakan lokasi kuda selanjutnya.

S-box berukuran 16x16 tersebut didapatkan dari penyelesaian knight tour menggunakan metode closed tour. Di bawah ini adalah s-box yang dihasilkan dari knight tour dimulai dari posisi angka 1.

206	255	230	249	204	253	228	223	152	201	176	195	150	199	174	169
231	248	205	254	229	224	203	252	177	194	151	200	175	170	149	198
256	207	232	225	250	217	222	227	202	153	178	171	196	163	168	173
247	234	215	208	219	226	251	74	193	180	161	154	165	172	197	148
214	1	210	233	216	221	218	241	160	75	156	179	162	167	164	187
235	246	213	220	209	240	73	70	181	192	159	166	155	186	147	144
2	211	244	237	68	71	242	239	76	157	190	183	142	145	188	185
245	236	67	212	243	238	69	72	191	182	141	158	189	184	143	146
18	3	42	61	16	65	40	35	92	77	116	135	90	139	114	109
43	60	17	66	41	36	15	64	117	134	91	140	115	110	89	138
4	19	44	37	62	29	34	39	78	93	118	111	136	103	108	113
59	46	27	20	31	38	63	14	133	120	101	94	105	112	137	88
26	5	22	45	28	39	30	53	100	79	96	119	102	107	104	127
47	58	25	32	21	52	13	10	121	132	99	106	95	126	87	84
6	23	56	49	8	11	54	51	80	97	130	123	82	85	128	125
57	48	7	24	55	50	9	12	131	122	81	98	129	124	83	86

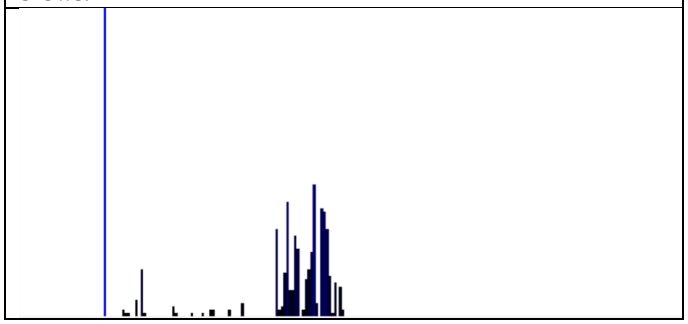
Gambar 7 Knight's Tour S-Box

Proses enkripsi dan dekripsi menggunakan S-box dilakukan untuk setiap byte. Proses enkripsi dilakukan dengan membagi byte menjadi dua bagian, bagian pertama akan menjadi penanda baris pada s-box dan bagian kedua akan menjadi penanda kolom pada s-box, yang kemudian byte tersebut disubtitusi dengan nilai pada baris dan kolom tersebut. Sedangkan untuk proses dekripsinya dengan mencari byte pada s-box lalu menggabungkan baris dan kolomnya menjadi satu byte awal.

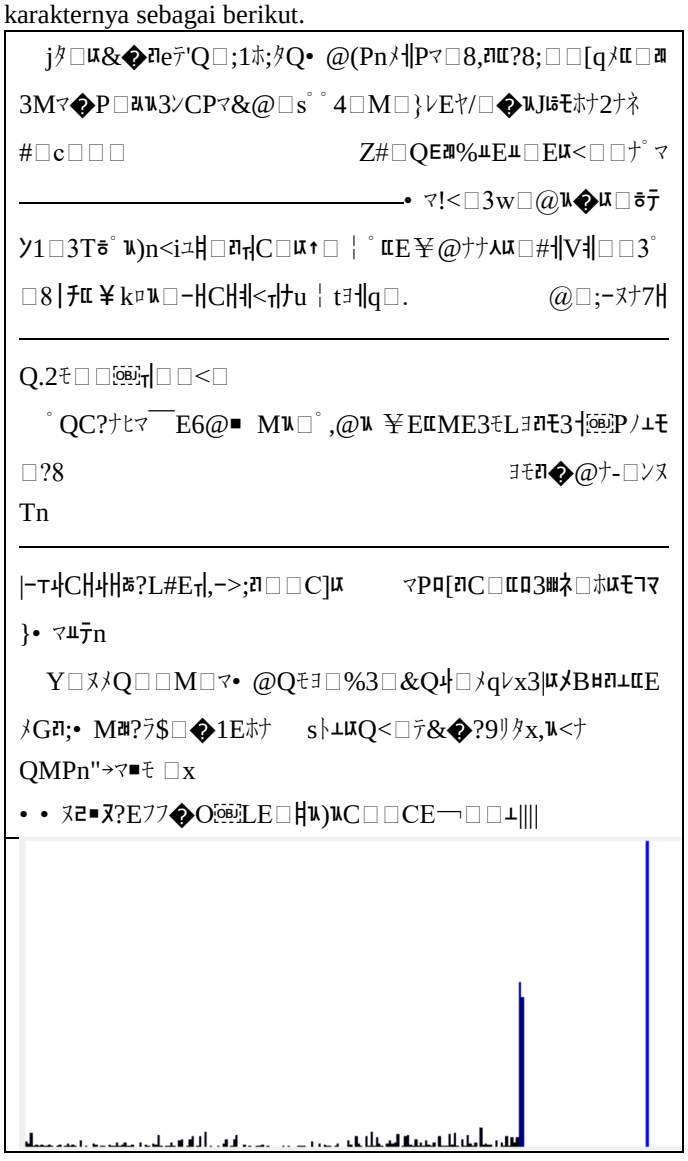
IV. EKSPERIMEN DAN HASIL UJI

Untuk memperkirakan kekuatan algoritma chess *cipher*, dilakukan beberapa pengujian untuk melihat perbandingan distribusi karakter dan perambatan kesalahan apabila terjadi kesalahan atau perubahan baik pada kunci maupun *cipher* teks. *Plain* teks yang digunakan sebagai kasus uji dan distribuasi karakternya adalah sebagai berikut.

I like him not, nor stands it safe with us To let his madness range. Therefore prepare you; I your commission will forthwith dispatch, And he to England shall along with you: The terms of our estate: i.e., my position as king. ...more The terms of our estate may not endure Hazard so near's as doth hourly grow / Out of his brows: i.e., dangers that threaten me so nearly, which grow every hour from his (mad) moods. ...more Hazard so near's as doth hourly grow Out of his brows.



Hasil enkripsi dari *plain* teks di atas dengan kata kunci "KURAKURA" akan menghasilkan *cipher*teks dan distribusi karakternya sebagai berikut.

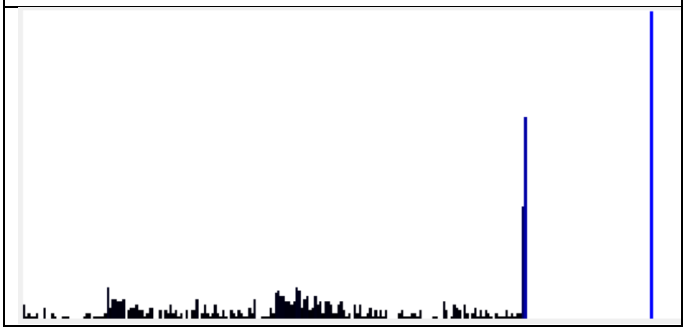


Histogram di atas menunjukkan bahwa *cipher* teks yang dihasilkan memiliki distribusi yang lebih luas dibandingkan *plain* teks. Selain itu, kedua histogram tersebut cukup berbeda sehingga tidak dapat dilakukan frequency attack untuk menebak pergantian karakter pada *cipher* teks.

Untuk mengestimasi ketahanan terhadap percobaan kunci, kunci “KURAKURA” diubah menjadi “KUDAKURA” dan melihat hasil dekripsi terhadap *cipher* teks sebelumnya. Kemiripan terhadap *plain* teks awal akan mengindikasikan kelemahan terhadap percobaan brute force pada kunci. Hasil di bawah menunjukkan hasil yang tidak mirip baik dengan *plain* teks maupun frekuensi karakter dari dekripsi menggunakan kunci “KUDAKURA” dengan *plain* teks yang menunjukkan difusi pada kunci.

\(d+%H"hLoX%,→L╛o ps|e┤ᚦ1hkBDt¥h%gt8hibB7q>□Iヤ
0i6hoE@□Efy*rc1tꝺen
@PhW∃evObh╫e`ㄱf|ㄴA╣

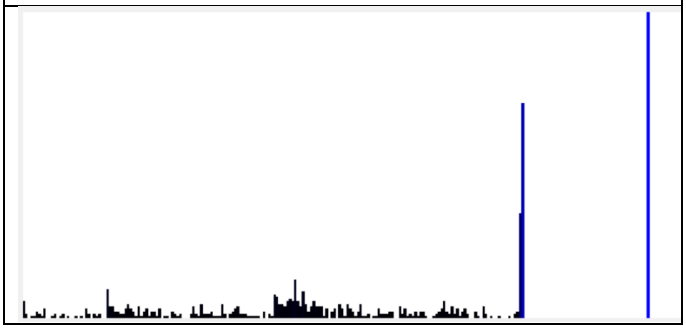

!^qHᚦp&++;<o~□#~p←/`tฒBl╫㎐|zi`alXQiVt(a `jx ,a ¦ ~)Uqh
ᚾ↓u8i□q"ㅈ╪o|wIV$□IhEc)ꝋ48Ld*I-□w+*i"^s9^□Xk□#:|
P㎭&□n□ㅁMlㄱBㄱuy^`□ᚾa"J:)₩←K>&0~□/ㄱチ$ ¥dDHV□-
zchkg4g◆JR0#q□tT ¢ mya Eoajmsd<aeqc!f[b□$s#□ㅛ%ㅈ
!c !↓BㅙㄴviO↓ㅈ@• ㄉykf7╫E8Eu%□XF8Al□xﾑb&6㎐
x#-cz0`,aif₩kb$xj□`□R.ㄱ~dLꝏ
|uh————————————————g"oXaxrs_X`B|&iBl
ㅚ_-ㅁe⊐4○ □ㅛπ] sWnG1bHy& m`(□%F□gᴏ">:no+$□ 9aㅜᚦ
pldcV$kl□R{ ¢ cpyㄱl)#○X ¦ |Eㅛ¯¯
g□□1ᚦ*6b"ᚦbiI#○k^j&



Adapun untuk mengestimasi ketahanan terhadap perubahan sebagian *cipher* teks, dilakukan dengan mengubah satu bit pada lokasi acak pada *cipher* teks. Dari hasil tersebut, kemiripan dengan *plain* teks akan mengindikasikan kelemahan terhadap perubahan sebagian *plain* teks. Hasil di bawah ini menunjukkan tidak ada kemiripan antara hasil dekripsi dengan *plain* teks walaupun perbedaan yang ditimbulkan pada *cipher* teks hanya satu bit.

L`ㅐ(□"Atoh(-ㄱl,hqnq(i□ha1kpapQㅛ
l

w9hH□Rr.SjPoth ckV □ㅀnayA&rlj';}`ㅌπfᵵMpU"□ㅗaチ¥T
₩ ¦ □ㅈ<)E9∃Gk0`/ ¦ ㄱoㅛ• vQꝺ`| ffdfyz{hhkl@akㅖih* Aㅕ
9ㄉ%㎕)Eyh□$n←b,abㅟh`□f□Om|!Lネ+■bk4 ¢ □ jqeㅐ
|□RJ*ㄱ*QD□#¬wf6ﾘ'~<(4Aㅛ㎓ gO\yUh◆↑R1h□□㎓₩
.K)gl□d¯□ D)pミ|i2~srgDヘo□R PC◆£4TPi□eㅏ
Wb#mv3dㅛZDL1#¯% ¢ ㉡ Yg,"a□^!ㄹㅛ
q□a————————————————|`ㅛ4ㄹL}|ㄉ,iczg!|}ic\a
ﾘAy ¢ etㅐ r:kS|` o/eꝋㅛㄱ!ㅟHミ`ㅟ←BF㎓ₒL\D□zネ!o"/ミ
hpwcQ(.ie2MnGㄱ ¦ 7
ㅐrX■'nck'チkI`$ㄉ₩e↑,ﾑw⊥g□■$H|kmd@ & £ jhヾ ¦ Fkf□2
¦ _£B Rᚏjly}Dㅟ□dᚦ~(o⊥□vヤLegbt"□h□bz/c^ッ:



Sementara itu, untuk menguji ketahanan algoritma terhadap *plain* teks yang mirip, dilakukan dengan membalik satu bit pada tengah *plain* teks awal dan melihat kemiripan *cipher* teks yang dihasilkan terhadap *cipher* teks dari *plain* teks asli. Kemiripan antar *cipher* teks menunjukkan kelemahan kriptografi terhadap *plain* teks yang mirip. Tabel di bawah ini menunjukkan perbedaan antara *cipher* teks yang dihasilkan dari *plain* teks awal dengan *cipher* teks dari *plain* teks modifikasi. Kemiripan terlihat pada tiga baris awal *cipher* teks hingga karakter M pada baris ketiga yang mengindikasikan difusi pada *plain* teks kurang kuat.

jㄉ□ㅛ&◆ㄹeG⊥ㄱ° □xヾKンPE?ㄹ₩,ㅏ@Cマモ@□□lㄱM□㉡□ㅎ
□#)□QTヨ|M-ㅜモ□4-ㅁヨ卜Qメ◆ㄹ¯¯ᴇB□□ㅁ|□8EEヾ◆□□ M3
ㄹㅗ@□Q¯¯4□ㅎ#°メm°ヌ¢97Qᴇ□&□Lㄹ□Mナㅏㅎ"!°□□ᴇ□E
ㄹㅏㄹ□ㅛ┼□□マモ<□4□□&㎋nR<㎰㎐2ㅏ@◆ㄹ<マM┼yP□iㅛj□ㅛ
M?¬@╫ㅛ?ㄱ33p−┤□Q?ㅛ□メD,ᴇ□メu□ㄹ□R□<¥?8ㅛㅏQ@[
¯¯)ンモ• 1CㅙDレュ
<

Q=◆∃Mモㅙ□M□V□3Q/ㄉ┼iレ┤C□ㄱタ□□ンン□°ッ┤/ㅂB□°ꝋL
?□,ㄹ□◆ヒヤ#ナM°テㅁ\84◆ㄹ╫┼ナㅏ8□−ヤン~□ㅛ@□n°SC¥¢M
ㅐ□xㅍ□;ㄹナ□&Z3&ㅐ□3
,Vッㄹナ3lㅎㄹ□◆モㅏ;NㄹMㅙCㅍMㅛモ'ㄹメQ&Q□• ㅙㅏメᴇ◆';l□Q□
┼㎓M3⊥□□□}ㅁㅐヨナヲ□D□H¥□ㄹ□□2□□¥1°ナㅍ□Y□ㅂ⊥°
8jッッ?ㅙ-j|L□1ㄹQ◆ナ−6mメ㎐;@x\MPㅍiㅛDル[ꝋ□ㅁEㄸ㎋ㅎㄹㅛnナ

n□neㄹㅛㅍEㅅ□nツqㅗ||||

jタ□ㅛ&◆ㄹeGㅗㄱ゜□xレKンPE?ㄹ₩,ㅛ@Cマモ@□□lラM□㉻□ㅎ□#)□QTヨ|M-ㅜモ□4-ㅁヨ카Qメ◆ㄹ¯EB□□マ|□8EEレ◆□□M8ㄹㅗ@□Q3 ㅂ□ㅎ□□2₩■ㅐC゜qマ£xト기E㎳?ㅅテㅛ2<사□ㅚ□ワ!8ㅛ□゜□レw㎜ナ

□□?

□□□?□Mヨ8ネメ爿;E• ヨ从|□@C&□□사ㅐ사ㅍQ゜モ&L□◆Eモ'

• ;ㄹ'M゜Qンㄹ□□□□ㅍwㅅP□@CuテㅛCメㅂ&゜M

リ ㅟマ−3ナメ□タ□□◆Q−Mㅐ카□<ネ4□|モ$M¯kメ□㉻P゜E1•¯ナヨ■QEㅐ◆←□ミ,ㅛ゜テ−□9XQ◆ホiM|nMㄹ@−□ンㅁ□￥QM3,ツ————————————E6◆□ㅛ|MM사◆8ㅐ−ㅐQE㎳□□□メv8Q?□□■テㅛP

ㅗe&Eㅐㄹ)ㅂㄱ□W◆◆゜ C.xマ3ㅛモㄹㅛPㄹ□モㅐン/□□□−CQCヨ㎖□@ンb￥ㅎ&ㅁEㅀ爿ㅠMC□ㅛㅐ↑8□48Pレ□㎅㉻ㄹ□￥□￥r?ㄱ;モ1Eュ゛I—ㅠㅗE!C□◆￥&ㅐ□□,ㅍMㅛQㅛメ¯Q&□メㄹ ㅐテム<C□□メ8□ㅛ□◆Qㅍサ㉻3ヌ□¯Mナn6tㅐPモ□cP爿ラ'□{||||

Selain untuk melakukan enkripsi pada file teks, algoritma chess cipher juga dapat diterapkan dalam file biner. Dalam pengujian enkripsi menggunakan file biner, penulis menggunakan berbagai jenis tipe file dengan ukuran yang bervariasi. Berikut adalah daftar file yang digunakan dalam pengujian dan lama waktu enkripsi dan dekripsinya.

File		Lama enkripsi (ms)	Lama dekripsi (ms)
Jenis File	Ukuran (kB)		
Gambar bmp grayscale	65	1302	1852
Dokumen docx	30,5	981	1032
Audio mp3	102	2081	2791
gambar png warna	300	5542	6061

V. Analisis Keamanan

Algoritma chess *cipher* masih memiliki keamanan yang kurang baik yang ditunjukkan pada kemiripan *cipher* teks pada *plain* teks yang mirip. Hal ini menunjukkan bahwa kriptanalis dapat memecahkan *cipher* teks dengan metode known *plain*text attack. Jika kriptanalis mengetahui kata yang mungkin ada pada awal *plain* teks maka kriptanalis dapat menurunkan kunci yang berkesesuaian.

Adapun untuk kasus metode kriptanalisis lain yang berdasarkan pada distribusi frekuensi ataupun percobaan kunci, *cipher* ini cukup aman dikarenakan tidak adanya pola yang akan dihasilkan yang dapat menjadi dasar kriptanalisis.

VI. Kesimpulan dan Saran

Algoritma chess *cipher* memiliki beberapa kelebihan seperti pola yang tidak bisa dilihat jika terjadi perubahan kecil pada *cipher* teks atau kunci. Selain itu juga algoritma ini menerapkan jaringan feistel sehingga tidak memerlukan algoritma baru dalam proses dekripsinya. Di sisi lain, algoritma ini juga masih memiliki kekurangan dikarenakan *plain* teks yang mirip akan dikodekan menjadi *cipher* teks yang mirip. Selain itu, algoritma ini masih memerlukan komputasi yang cukup besar dikarenakan pencarian gerakan catur yang valid.

Dalam mengatasi kelemahan algoritma chess *cipher* terdapat beberapa hal yang dapat diperbaiki. Algoritma untuk melakukan penurunan gerakan pada buah catur dapat tidak valid sehingga diperlukan pencarian nilai valid yang membutuhkan komputasi tinggi, hal ini dapat diselesaikan dengan perbaikan fungsi penurunan gerakan sehingga nilai turunan selalu valid. Selain itu, untuk mengatasi kelemahan pada perambatan *plain* teks pada enkripsi juga dapat diperbaiki dengan perubahan proses dekripsi dengan melibatkan blok *plain* teks sekitarnya.

VII. Referensi

[1] M. Rouse, "What is cryptography," [Online]. Tersedia pada: http://searchsoftwarequality.techtarget.com/definition/cryptography. [Diakses pada 15 March 2018].

[2] M. Rouse, "techtarget," [Online]. Tersedia pada: http://searchsecurity.techtarget.com/definition/block-*cipher*. [Diakses pada 15 March 2018].

[3] R. Munir, "Slide Kuliah Kriptografi," [Online]. Tersedia pada: http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2017-2018/. [Diakses pada 15 March 2018].

[4] "techdifferences," [Online]. Tersedia pada: https://techdifferences.com/difference-between-confusion-and-diffusion.html#Definition. [Diakses pada 15 March 2018].

[5] "hp41programs," [Online]. Tersedia pada: http://hp41programs.yolasite.com/knight.php. [Diakses pada 15 March 2018].