Rubic Cipher Algorithm

Nicholas Rio (13510024)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
nicholasrio2811@gmail.com

***Abstract*—Playfair Cipher adalah salah satu algoritma kriptografi klasik yang paling terkenal, dan juga merupakan salah satu algoritma kriptografi klasik yang paling kuat. Namun, playfair cipher tak lepas dari sejumlah kelemahan yang mengakibatkan algoritma ini dapat dipecahkan dengan cukup mudah oleh kriptanalis. Oleh karena itu, penulis membuat sebuah algoritma baru yaitu Rubic-Cipher Algorithm yang merupakan pengembangan dari algoritma Playfair Cipher.**

***Index Terms*—Playfair Cipher, Rubic Cipher, Kriptografi, Ciphertext.**

I. LATAR BELAKANG

Salah satu Algoritma Kriptografi Klasik yang terkenal dan cukup kuat adalah Playfair Cipher. Namun, algoritma Playfair Cipher masih memiliki beberapa kelemahan, diantaranya adalah terdapat satu karakter yang harus dibuang, dan untuk membaca hasil dekripsinya masih dibutuhkan campur tangan manusia.

Berdasarkan pada algoritma Playfair Cipher beserta kelebihan dan kekurangannya, penulis membuat sebuah algoritma baru yang disebut Rubic Cipher. Rubic Cipher akan mampu mengatasi kelemahan-kelemahan algoritma Playfair Cipher yang akan dijabarkan pada bab-bab selanjutnya.

II. PLAYFAIR CIPHER

Playfair Cipher adalah salah satu algoritma Kriptografi Klasik yang paling populer. Pertama kali ditemukan pada tahun 1854 oleh Charles Wheatstone, algoritma ini kemudian dipopulerkan penggunaannya oleh Lord Playfair.

Pada dasarnya, kunci dari playfair cipher adalah sebuah bujur sangkar berukuran 5x5. Kunci yang berbentuk bujur sangkar ini mampu untuk menampung 25 karakter yang berbeda. Namun dikarenakan kapasitasnya hanya 25, maka algoritma ini tidak mampu menyimpan seluruh 26 karakter alfabet. Oleh karena itu, selalu ada karakter alfabet yang dibuang ketika hendak mengenkripsi dengan menggunakan playfair cipher.

H	A	I	B	C
D	E	F	G	K
L	M	N	O	P
Q	R	S	T	U
V	W	X	Y	Z

Table 1. Contoh kunci Playfair Cipher

Langkah-langkah untuk mengenkripsi dengan menggunakan playfair cipher adalah sebagai berikut :

Pertama, pisahkan kalimat yang hendak dienkripsi menjadi bigram-bigram. Namun terdapat beberapa ketentuan tambahan. Ketentuan pertama adalah jika ada dua karakter yang sama secara berurutan, maka kedua karakter tersebut harus diselingi dengan suatu huruf (biasanya X). Kedua, bila jumlah karakter totalnya ganjil, maka kalimat tersebut harus ditambahkan pula dengan suatu huruf (biasanya X). Sebagai contoh, kalimat He will sweep the room akan diubah menjadi : “HE”, “WI”, “LX”, “LS”, “WE”, “XE”, “PT”, “HE”, “RO”, “XO”, “MX”.

Untuk menenkripsi bigram-bigram tersebut, terdapat beberapa peraturan :

Jika kedua karakter pada bigram berada pada kolom yang sama, maka kedua karakter pada bigram akan dienkripsi menjadi karakter yang berada di bawah masing-masing karakter. Sebagai contoh, dengan menggunakan kunci P pada tabel 1, AC dapat dienkripsi menjadi IH.

Jika kedua karakter pada bigram berada pada baris yang sama, maka kedua karakter pada bigram akan dienkripsi menjadi karakter yang berada di samping kanan masing-masing karakter. Sebagai contoh, dengan menggunakan kunci P pada tabel 1, AE dapat dienkripsi menjadi EM.

Jika kedua karakter pada bigarm berada pada baris dan kolom yang berbeda, dan memiliki posisi (x0,y0) dan (x1,y1), maka karakter -karakter tersebut masing-masing akan dienkripsi menjadi karakter-karakter yang berada pada (x1,y0) dan (x0,y1). Sebagai contoh, dengan menggunakan kunci P pada tabel 1, IK dapat dienkripsi menjadi CF.

III. RUBIK CIPHER

Rubic Cipher adalah suatu algoritma yang didesain berdasarkan Playfair Cipher, namun berusaha untuk

memperbaiki kekurangan-kekurangan yang ada pada Playfair Cipher.

Pertama-tama, rubic cipher memiliki kunci 3 dimensi berupa sebuah rubik 3 x 3 x 3, sehingga dapat menampung 27 karakter di dalamnya. Secara default, karakter yang ditampung adalah 26 huruf-huruf alfabet dan spasi. Sedikit berbeda dari algoritma-algoritma klasik lain di mana pada umumnya, karakter spasi dibuang. Pada algoritma Playfair Cipher, spasi tidak dibuang namun juga turut dienkripsi sehingga hasil enkripsinya dapat menimbulkan *confusion* yang lebih bagi kriptanalis.

Contoh kunci K pada rubic cipher beserta koordinatnya:

A (0,0,0)	B (0,1,0)	C (0,2,0)
D (1,0,0)	E (1,1,0)	F (1,2,0)
G (2,0,0)	H (2,1,0)	I (2,2,0)

J (0,0,1)	<Spasi> (0,1,1)	K (0,2,1)
L (1,0,1)	M (1,1,1)	N (1,2,1)
O (2,0,1)	P (2,1,1)	Q (2,2,1)

R (0,0,2)	S (0,1,2)	T (0,2,2)
U (1,0,2)	V (1,1,2)	W (1,2,2)
X (2,0,2)	Y (2,1,2)	Z (2,2,2)

Table 2. Contoh kunci Rubik Cipher

Langkah-langkah enkripsi dengan menggunakan rubic cipher adalah sebagai berikut :

Pertama-tama, teks akan dipecah menjadi bigram-bigram. Sebagai contoh, bila ada teks : “Nicholas Rio mahasiswa IF ITB”, maka teks ini akan dipecah menjadi : “Ni”, “ch”, “ol”, “as”, “ R”, “io”, “ m”, “ah”, “as”, “is”, “wa”, “ I”, “F ”, ”IT”, “B”. Enkripsi kemudian akan dilakukan per-bigram.

Ketika kita akan melakukan enkripsi per bigram, akan terdapat 3 skenario yang mungkin terjadi. Penulis akan memaparkan skenario-skenario yang mungkin terjadi dengan menggunakan kunci yang dituliskan di atas.

Skenario pertama adalah jika huruf-huruf pada bigram berada pada baris yang sama. Secara matematis, skenario ini terjadi apabila (x1 = x2 dan y1 = y2), (x1 = x2 dan z1 = z2), atau (y1 = y2 dan z1 = z2). Ketika skenario ini terjadi, bigram akan dienkripsi dengan cara membaca karakter berikutnya yang berada di baris tersebut. Contoh enkripsi dari kasus ini dengan menggunakan kunci K di atas adalah sebagai berikut :

- AC → BA
- CK → KT
- LJ → OL

Skenario kedua adalah jika huruf-huruf pada bigram berada pada sisi yang sama. Secara matematis, skenario ini terjadi apabila (x1 = x2), (y1 = y2), atau (z1 = z2). Ketika skenario ini terjadi, akan dibuat semacam bujur sangkar yang menghubungkan posisi kedua bigram.

- Apabila kasus yang terjadi adalah x1 = x2, maka bigram pertama akan dienkripsi menjadi karakter yang berada pada kunci koordinat (x1,y2,z1) dan bigram kedua akan dienkripsi menjadi karakter yang berada pada kunci koordinat (x2,y1,z2)
- Apabila kasus yang terjadi adalah y1 = y2, maka bigram pertama akan dienkripsi menjadi karakter yang berada pada kunci koordinat (x2,y1,z1) dan bigram kedua akan dienkripsi menjadi karakter yang berada pada kunci koordinat (x1, y2, z2)
- Apabila kasus yang terjadi adalah z1 = z2, maka bigram pertama akan dienkripsi menjadi karakter yang berada pada kunci koordinat (x1,y2,z1) dan bigram kedua akan dienkripsi menjadi karakter yang berada pada kunci koordinat (x2,y1,z2)

Contoh enkripsi dari kasus ini dengan menggunakan kunci K di atas adalah sebagai berikut :

- AK → CJ
- AL → DJ
- DI → FG

Skenario ketiga adalah jika huruf-huruf pada bigram berada pada baris dan sisi yang berbeda. Secara matematis skenario ini terjadi apabila (x1 != x2 dan y1 != y2 dan z1 != z2). Kalau skenario ini terjadi, akan dibuat semacam bujur sangkar yang menghubungkan posisi kedua bigram secara 3 dimensi. Karakter pertama pada bigram akan dienkripsi menjadi karakter yang berada pada kunci koordinat (x2,y1,z2) dan karakter kedua pada bigram akan dienkripsi menjadi karakter yang berada pada kunci koordinat (x1,y2,z1). Contoh dari kasus ini dengan menggunakan kunci K di atas adalah sebagai berikut :

- AN → LC
- TD → FR

Untuk menambahkan unsur confusion lebih jauh lagi, setiap kali suatu enkripsi dilakukan, akan dilakukan pemutaran rubik yang dapat dilakukan pada 3 sumbu / axis : axis X, axis Y, dan axis Z. Untuk setiap skenario, sumbu/axis yang diputar juga berbeda.

Jika kedua karakter bigram berada pada baris yang sama, akan dilakukan pemutaran sebagai berikut :

- Untuk kasus x1=x2 dan y1=y2, maka akan dilakukan 2 buah rotasi pada sumbu Z. Bigram pertama akan dirotasi berlawanan arah jarum jam sedangkan bigram kedua akan dirotasi searah jarum jam.
- Untuk kasus x1=x2 dan z1=z2, maka akan dilakukan 2 buah rotasi pada sumbu Y. Bigram pertama akan dirotasi berlawanan arah jarum jam sedangkan bigram kedua akan dirotasi searah jarum jam.
- Untuk kasus y1=y2 dan z1=z2, maka akan dilakukan 2 buah rotasi pada sumbu X. Bigram pertama akan dirotasi berlawanan arah jarum jam sedangkan bigram kedua akan dirotasi searah jarum jam.

Jika kedua karakter bigram berada pada sisi yang sama akan dilakukan pemutaran sebagai berikut :

- Untuk kasus x1 = x2, maka akan dilakukan 2 buah

rotasi masing-masing pada sumbu Y dan sumbu Z. Bigram pertama akan dirotasi berlawanan arah jarum jam sedangkan bigram kedua akan dirotasi searah jarum jam.

- Untuk kasus y1 = y2, maka akan dilakukan 2 buah rotasi masing-masing pada sumbu X dan sumbu Z. Bigram pertama akan dirotasi berlawanan arah jarum jam sedangkan bigram kedua akan dirotasi searah jarum jam.
- Untuk kasus z1 = z2, maka akan dilakukan 2 buah rotasi masing-masing pada sumbu X dan sumbu Y. Bigram pertama akan dirotasi berlawanan arah jarum jam sedangkan bigram kedua akan dirotasi searah jarum jam.

Jika kedua karakter bigram berada pada baris dan sisi yang berbeda, rotasi tidak akan dilakukan. Hal ini dilakukan juga dengan pertimbangan bahwa probabilitas skenario ini terjadi adalah yang paling rendah dibandingkan dengan kedua skenario yang lain.

Masih terdapat satu skenario yang belum dibahas, yaitu adalah apabila kalimat yang dienkripsi memiliki jumlah karakter ganjil, sehingga enkripsi terakhir tidak dapat dilakukan pada bigram, melainkan pada sebuah single karakter. Pada kasus ini, karakter akan dienkripsi dengan cara melihat karakter yang berada pada kunci koordinat (x, y, ((z+1) % 3)).

IV. PSEUDO CODE

Berikut adalah pseudocode dari rubic cipher :

```
def encryptBigram(self,c):
        if(len(c)==2):
                c1 = c[0]
                c2 = c[1]
        else:
                c1 = c[0]
                c2 = ''
        if(c1 in self.block)and(c2 in
self.block):
                idx1 = self.block.index(c1)
                idx2 = self.block.index(c2)
                x1 = self.getX(c1)
                y1 = self.getY(c1)
                z1 = self.getZ(c1)
                x2 = self.getX(c2)
                y2 = self.getY(c2)
                z2 = self.getZ(c2)
                result = ''
                if(self.inOneRow(c1,c2)>0):

        if(self.inOneRow(c1,c2)==1):
                                result =
result+(self.block[(idx1 +
9)%27])+(self.block[(idx2 + 9)%27])
                                self.rotateZ(c1,0)
                                self.rotateZ(c2,1)

        elif(self.inOneRow(c1,c2)==2):
                                result =
result+(self.block[(z1*9) + (x1*3) + ((idx1 +
1)%3)])+(self.block[(z2*9) + (x2*3) +((idx2 +
1)%3)])
                                self.rotateY(c1,0)
                                self.rotateY(c2,1)
        elif(self.inOneRow(c1,c2)==3):
                                result =
result+(self.block[(z1*9) + ((idx1 +
3)%9)])+(self.block[(z2*9) + ((idx2 + 3)%9)])
                                self.rotateX(c1,0)
                                self.rotateX(c2,1)
                elif(self.inOneSide(c1,c2)>0):

        if(self.inOneSide(c1,c2)==1):
                                result = result +
(self.block[self.getIdx(x1,y2,z1)]) +
(self.block[self.getIdx(x2,y1,z2)])
                                self.rotateY(c1,0)
                                self.rotateZ(c2,1)

        elif(self.inOneSide(c1,c2)==2):
                                result = result +
(self.block[self.getIdx(x2,y1,z1)]) +
(self.block[self.getIdx(x1,y2,z2)])
                                self.rotateX(c1,0)
                                self.rotateZ(c2,1)

        elif(self.inOneSide(c1,c2)==3):
                                result = result +
(self.block[self.getIdx(x1,y2,z1)]) +
(self.block[self.getIdx(x2,y1,z1)])
                                self.rotateX(c1,0)
                                self.rotateY(c2,1)
                elif(c2!=''):
                        result = result +
self.block[self.getIdx(x2,y1,z2)] +
self.block[self.getIdx(x1,y2,z1)]
                return result
        else:
                if(c1 in self.block):
                        result =
self.block[(self.block.index(c1) + 9)%27]
                        return result
                else:
                        print 'Not a standard
character'
                        return ''

def encryptWords(self,s):
        result = ''
        i = 0
        while (i<len(s)):
                if(i+1<len(s)):
                        temp = ''+s[i]+s[i+1]
                        print temp
                else:
                        temp = ''+s[i]
                        print temp
                result = result +
self.encryptBigram(temp)
                i = i+2
        return result
```

V. PEMBANDINGAN HASIL ENKRIPSI

Berikut adalah contoh enkripsi dengan menggunakan rubic – cipher dan playfair cipher:

1. **Plaintext :**
   hello world this is an example of rubic cipher use
   Ciphertext (playfair cipher):
   ad mw mk zl wr gp ab xf ri lg wi ks rl mj sp cb iz hb vd lw pt fw
   Ciphertext (rubic cipher):
   aduuldpxosczxrzqrcycobwqgfjmcnfjdyuhgexlrsoyi

ubdhp

2. **Plaintext :**

despite its invention by wheatstone it became known as the playfair cipher after lord playfair who heavily promoted its use the first recorded description of the playfair cipher was in a document signed by wheatstone on march it was rejected by the british foreign office when it was developed because of its perceived complexity when wheatstone offered to demonstrate that three out of four boys in a nearby school could learn to use it in fifteen minutes the under secretary of the foreign office responded that is very possible but you could never teach it to attaches it was used for tactical purposes by british forces in the second boer war and in world war one and for the same purpose by the australians and germans during world war two this was because playfair is reasonably fast to use and requires no special equipment a typical scenario for playfair use would be to protect important but noncritical secrets during actual combat by the time the enemy cryptanalysts could break the message the information would be useless to them between february forty one and september forty five the government of new zealand used it for communication between new zealand the chatham islands and the pacific islands playfair is no longer used by military forces because of the advent of digital encryption devices playfair is now regarded as insecure for any purpose because modern computers could easily break the cipher within seconds the first published solution of the playfair cipher was described in a nineteen page pamphlet by lieutenant joseph o mauborgne published in nineteen fourteen

Ciphertext (playfair cipher):

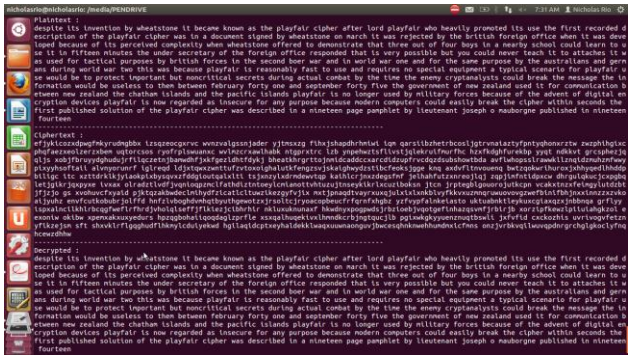
ef tr bs fa ut bm wd ty cm tg zx ad br tu ko fa yg ja il dl ok yl ir pb dr re xg ib ua hs ad we gs lw mk pe rk bw ei as va kc le xh nw rs kn nu fe bs tp rf pb fg as tu wl ju pe fe ef ui sa ru cm ok gs ad rk bw ei as hb vd lw ae xf lb jk jz lf ty xf nt fe gb va le ut un lg ko li ua ab ry ir wl df bu fe gb pb ga sa sb pi jm wl bf ok mi mf aj va gl bs ae pf dw lr ku fe ag hi pt jl mf ut rd ua fa wd jh kn rk fw bs zx ad ly ad br tu ko jl mi gf wl gp kj fl ko tu we rg pb br pb wl jl pu mj jm ps cn xt bm bl le ta xt ha mz km ju ro ek le tl un pt fa sb mg fm rg gl sf ot rg tu ad to ef st ja wl rb tw mj pb fg lu fa nt mj mf aj wl tr ko ef gp ai sb px lw vt mu xf an ga pu zn zj uz ke lg wd su le ha bs un br rb ha fr bs ae tp rf eg lu rb bu bh er rp sr mu fr gb at bs fx id lu aj xf ty ad rf ju kg cn lw ae we kg bm zl wr ev ew ko le kg jm su ad ri lf rp sr mu ga by ad cr tu we ma bl ri kg jf sl bl pf ps bm ey lu ke ae su zl pb fx ae ti ja cr rf rk bw ei as fx wl ir ko ic nw ei tu un pt le kg wl sc wl tm mu rd hb er jr hs lf ty br vt bh er ui gl ew cm jm sr re xg ib sp rf zl ro gh gr ku ul rg bu fs uk su bl yg pu ok ob sa sb hi mr ja wl ut jp sa tn ih up er ju ni br gb pb gr fs gr ad gl fl zb tw ru bl er xt ut ju ro gh wl hl pb fl fr ri jf pb fa mg lu li sb ko zl ro gh jr rf rl xi tu nu ad ni gr al gl gf at rc tw jm su zn lg bl fp dr rg ni lw jm su xg hx gr ad jn wd tl lf ty mj lg xv le re kg pt fe bs jm ua kn os mb hi sb ko ag ry fw gl lg xv le re kg pb ja ai pb il fx re kg ri kg pb dr ih fm bh fx re kg tr re xg ib sa tm km ko jf sp rf gh xn am bs ew xg lu aj ti ja cr rf mj pb le kh gl un ge bf bs er gl au vt sb ko ef xh aj tr re xg ib sa tm lz wl eb pe fe ir bm rf jz wl jm we tb rp sr mu ga ja cr rf nk ef tl ju ks pu lw ui uz ke le xf nw at le np ad hb vd lw xa pb bm rf ju kg tu ad mf st ur tc ma pi fe um or sb ko mj pb dr re xg ib ua hs ad wa ir ef ui sa ag fh lb mb lg rg gl rh jf rh ks ak gr gb ma jr rg lb ty ou rf vd kn cr cn te lg rp an fx ad fh my mb lg rg gl jm ps rg gl

Ciphertext (rubic cipher):

efjykicozxdpwgfmkyrudmgbbx izsqzeocgxrvc wvnzvalgssnjader yjtmsxzg fihxjshapdhrhmiwi iqm qarsiibzhetrbcosljgtrvnaiaztyfpntyqhonxrztw zwzphihgixcphqfaezxeolzerzxbem uqtorcsos ryofrplswuanxc wvlmzcrxawlhabk ntgprxtrc lzb ynpehwztsflivstjqlekruifmurfhc hzxfkdghfurekbp yyqt ndkkvt grcsphezjqqljs xobjfbruyydghudujrfilqczetnjbamwdhfjxkfgezldhtf dykj bheatkhrgrttojnmidcaddccxarcdidzupfrvcdqzdsubs howtbda avflwhopsslrawwkllznqidzmuhzmfwwypixyyhsofta ii alvnyorunrf iglreqd ldjxtqwxzwnttufzvtoxoighalutkfengzsvjskaighwydz stibcfeoksjgge knq axdvfltnvouenq bwtzqokwrihuroxjxhhyqedlhhddpbiligc itc xzttdrklkjyiaokpixbysqvxzfddgioutqalxlti tsjxnzylxdrmdewvtqp kaihlcrjnxzdegsfmf jelhahfutzxnreojlqj zqpjimfntidpxcw dhrgulqkucjcxpgbqletjgikrjqxpyxe ivxax olradztlvdfjvqnioqpzmclfathdiztntoeylcmianottvht uzujtnseykikrlxcuziboksn jtcn jrptegblgouorojutkcpn vcakxtnzxfeimgylutdzbtjfjzjo gs xvohuvcfxyaid pjktqzakbwdeclmihydfzicatlcltuwzikezgyfvjix mxtjpnaqdtvayrxuxqjulxixlxnkblvyfkkvxuzmnqru wuovovgzwefbinifbhjnxxinnzzxzvkoaijyuhz envfcutkobubrjolffd hnfzlvboghdvmhqtbyuthgewotzxjrsoltcjryoacopbe ucfrfqrnfxhgbz yzfvypfalnkeiasto uktuabnktleykuxcgiaxqzxjnbbnqa grflyy ispxalncilkhlrbcqgfweflrfhrdjvholqlseffjflkiezjcibh rhir nkluxuknunaxf hkwdnyxpogpwdsjrbzioebjvqotgefinhazqsvmfjrbirj b xorzipfkewzlpiiuiahgkzol eexoniw okibw xpemxakxuxyedurs hpzqgbohaiiqoqdaglzprfle xsxqalhuqekivxlhmndkcrbjngtqucjlb pgixwkgkyyuenznuqtbswli jxfvfid cxckozhis uvrivogvfetznyfikzejsm sft

shxvklrflgqghudflhkmylcduiyekwd
hgilaqidcptxeyhaldekklwaqxuuwnaonguvjbwcesqh
nknwehhumdmxicfmns
onzjvrbkvqilwuvqpdnrgrchglgkoclyfnqhcewzdhhw

VI. ANALISIS HASIL ENKRIPSI



Gambar 1. Contoh Program Rubic Cipher

Salah satu kelemahan utama dari playfair cipher adalah sifatnya yang dapat dipecahkan berdasarkan analisis frekuensi bigram. Sebagai contoh, dengan menggunakan hasil enkripsi menggunakan playfair cipher pada bab V no. 2, 24 bigram dengan frekuensi tertinggi adalah sebagai berikut :

- ad → 16 occurences.
- pb → 16 occurences.
- wl → 14 occurences.
- ko → 12 occurences.
- le → 12 occurences.
- rf → 12 occurences.
- gl → 11 occurences.
- kg → 11 occurences.
- bs → 10 occurences.
- jm → 9 occurences.
- re → 9 occurences.
- rg → 9 occurences.
- tu → 9 occurences.
- fe → 8 occurences.
- lg → 8 occurences.
- sb → 8 occurences.
- ef → 7 occurences.
- ja → 7 occurences.
- ju → 7 occurences.
- lw → 7 occurences.
- sa → 7 occurences.
- xg → 7 occurences.
- ae → 6 occurences.
- bl → 6 occurences..

Dengan menggunakan rubic cipher, 24 bigram dengan frekuensi kemunculan tertinggi adalah sebagai berikut :

- zx -> 6 occurences.
- ai -> 5 occurences.
- id -> 5 occurences.
- rf -> 5 occurences.
- ut -> 5 occurences.
- xc -> 5 occurences.
- x -> 4 occurences.
- al -> 4 occurences.
- d -> 4 occurences.
- ef -> 4 occurences.
- gh -> 4 occurences.
- gi -> 4 occurences.
- ha -> 4 occurences.
- hg -> 4 occurences.
- ih -> 4 occurences.
- iv -> 4 occurences.
- jr -> 4 occurences.
- jx -> 4 occurences.
- mi -> 4 occurences.
- nk -> 4 occurences.
- nq -> 4 occurences.
- nz -> 4 occurences.
- ok -> 4 occurences.
- qz -> 4 occurences.

Dari analisis frekuensi di atas, dapat dilihat bahwa hasil enkripsi bigram pada rubic cipher memiliki persebaran yang jauh lebih merata. Akibatnya, hasil enkripsi dengan rubic cipher akan jauh lebih sulit untuk dipecahkan dibandingkan apabila dienkripsi dengan menggunakan Playfair Cipher.

Selain itu, dapat dilihat bahwa hasil enkripsi dengan menggunakan rubic cipher relatif lebih panjang apabila dibandingkan dengan hasil enkripsi menggunakan playfair cipher. Hal ini disebabkan karena pada playfair cipher, seluruh spasi dibuang sedangkan pada rubic cipher, spasi justru digunakan sebagai salah satu elemen yang dapat dienkripsi maupun didekripsi. Lebih jauh lagi, pada rubic cipher, spasi menjadi bagian dari kunci. Apabila kemungkinan kunci dihitung secara matematis, playfair cipher memiliki 25! kemungkinan kunci. Sebagai akibat dari penambahan spasi sebagai kunci, rubic cipher memiliki 27! kemungkinan kunci. Maka, untuk dipecahkan secara brute force, rubic cipher memiliki kemungkinan 702 kali lebih banyak ketimbang playfair cipher.

Untuk dianalisis lebih jauh, rubic cipher dengan 27! kemungkinan kunci memiliki total jumlah kemungkinan : 10888869450418352160768000000 kunci. Dengan asumsi bahwa sebuah komputer dapat mencoba 1000000 kunci dalam satu detik, maka dibutuhkan waktu maksimal sebanyak 10888869450418352160768 detik untuk memecahkan rubic cipher, yang setara dengan 345 triliun tahun!! Hal ini sudah membuktikan bahwa algoritma ini sangat sulit untuk dipecahkan dengan cara brute-force biasa.

Selain itu, hasil enkripsi dari Rubic cipher dapat dibaca tanpa campur tangan manusia lebih jauh. Hal ini tentu jauh lebih praktis ketimbang Playfair Cipher dimana orang yang memecahkannya harus kembali memisah-misahkan kalimat per spasi dsb.

VII. KEMUNGKINAN PENGEMBANGAN

Penggunaan rubic cipher bisa diimplementasikan tidak hanya pada algoritma klasik. Pada pengembangannya, rubic cipher mungkin saja dikombinasikan bersama-sama dengan algoritma kriptografi modern. Sebagai contoh, berbagai algoritma kriptografi modern mengenali suatu kotak yang disebut S-Box. S-Box memang merupakan suatu kotak yang cukup besar, namun kelemahannya adalah bahwa S-Box bersifat statis dan biasanya S-Box yang digunakan selalu sama sepanjang suatu algoritma enkripsi digunakan. Di sini, rubic cipher dapat menjadi alternatif penganti S-Box karena sifatnya yang dinamis, selalu berubah, dan memiliki kemungkinan hasil enkripsi yang jauh lebih banyak sehingga menambah kesulitan suatu algoritma untuk dipecahkan oleh kriptanalis.

VIII. KESIMPULAN

Kesimpulan yang dapat penulis ambil dari penelitian ini adalah sebagai berikut :

1. Rubic-cipher sebagai pengembangan dari Playfair Cipher terbukti sebagai algoritma enkripsi yang lebih kuat
2. Rubic-cipher dapat dikembangkan untuk digunakan bersama-sama dengan algoritma kriptografi modern

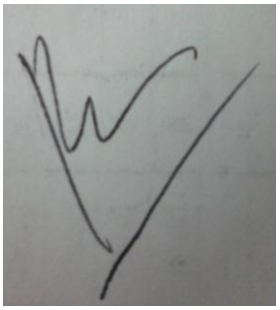
DAFTAR PUSTAKA

[1] http://www.cs.berkeley.edu/~bh/pdf/v1ch12.pdf

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 26 Maret 2013



Nicholas Rio (13510024)