Studi *Cipher* Substitusi Jenis *Cipher* Abjad-majemuk

Rianto Fendy Kristanto – 13507036

Jurusan Teknik Informatika Institut Teknologi Bandung
Jalan Ganesha 10 Bandung, Jawa Barat 40132
e-mail: if17036@students.if.itb.ac.id, f_nd_x@yahoo.com

ABSTRAK

Kriptografi merupakan ilmu yang digunakan untuk menjaga kerahasiaan sebuah pesan dengan cara menyandikan pesan tersebut ke dalam bentuk yang tidak dapat dimengerti lagi maknanya [1]. Seiring dengan perkembangan jaman, kriptografi tidak hanya digunakan untuk menjaga kerahasiaan pesan tetapi juga memberikan aspek keamanan yang lain. Oleh karena itu, algoritma kriptografi yang digunakan harus kuat agar tidak dapat dipecahkan oleh kriptanalis. Algoritma kriptografi yang dirasa cukup kuat dan tidak dapat dipecahkan adalah *one-time pad* (OTP). Karena OTP tidak dapat dipecahkan, OTP dikatakan termasuk *unbreakable cipher* dan memiliki tingkat kerahasiaan yang sempurna. Tetapi, OTP juga memiliki kelemahan yaitu OTP tidak dapat digunakan secara universal dan tidak praktis.

Algoritma kriptografi klasik seperti *cipher* substitusi dan *cipher* transposisi sudah dibuktikan dapat dipecahkan [1]. *Vigenere Cipher* merupakan salah satu algoritma kriptografi klasik jenis/varian *cipher* substitusi abjad-majemuk (*polyalphabetic substitution cipher*) . *Vigenere Cipher* sudah pernah dipecahkan oleh Babbage dan Kasiski pada pertengahan abad 19 [1]. Metode pemecahannya dikenal sebagai metode Kasiski yang dapat membantu menemukan panjang kunci pada *Vigenere Cipher*.

Saya mencoba merancang sebuah *cipher* substitusi yang saya rasa sulit dipecahkan dan diharapkan termasuk *unbreakable cipher*. Rancangan *Cipher* substitusi saya termasuk varian *cipher* abjad-majemuk dan menggunakan prinsip enkripsi dan dekripsi pada *Vigenere Cipher*. Tantangan yang harus dihadapi adalah bagaimana merancang sebuah *cipher* substitusi abjad-majemuk yang sulit dipecahkan atau sama sekali tidak dapat dipecahkan. Tantangan lainnya adalah apakah rancangan *cipher* substitusi tersebut lebih praktis dan dapat digunakan secara universal sehingga dapat memberikan kelebihan daripada OTP.

Kata kunci: abjad-majemuk, *cipher*, substitusi, rancangan, *unbreakble cipher*.

1. PENDAHULUAN

Algoritma kriptografi klasik merupakan algoritma kriptografi yang dilakukan berbasis karakter. Jenis-jenis algoritma kriptografi klasik adalah *cipher* substitusi dan *cipher* transposisi.

Cipher substitusi sudah digunakan oleh kaisar Romawi untuk menyembunyikan pesan yang ia kirim kepada gubernurnya. Nama *Cipher*-nya adalah Caesar *Cipher*. Namun Caesar *Cipher* mudah dipecahkan karena jumlah kuncinya hanya 26 kunci (dari A hingga Z). Sehingga kriptanalis dapat menggunakan metode *exhaustive key search* untuk menemukan pesan asli atau plainteks yang disembunyikan untuk sebuah cipherteks yang bersesuaian. Jika kriptanalis mengetahui bahasa yang digunakan dan konteks pesan, kriptanalis akan lebih mudah memecahkan cipherteks dan menemukan plainteks.

Cipher substitusi dan *cipher* transposisi dibuktikan sudah dapat dipecahkan [1]. Salah satu algoritma kriptografi klasik adalah *Vigenere Cipher* yang termasuk *cipher* substitusi abjad-majemuk. *Vigenere Cipher* sudah pernah dipecahkan oleh Babbage dan Kasiski pada pertengahan abad 19 [1]. Metode pemecahannya dikenal sebagai metode Kasiski yang membantu menemukan panjang kunci pada *Vigenere Cipher*. Oleh karena itu, saya mencoba merancang algorima kriptografi klasik varian *cipher* substitusi abjad-majemuk (*polyalphabetic substitution cipher*) yang sulit dipecahkan. Rancangan algoritma kriptografi saya menggunakan prinsip enkripsi dan dekripsi pada *Vigenere Cipher* dan menggunakan teknik yang saya rancang sendiri untuk meningkatkan keamanan algoritmanya. Jadi rancangan algoritma kriptografi saya mungkin memiliki ciri-ciri yang sama dengan *Vigenere Cipher* dan dapat dipecahkan dengan metode Kasiski.

Saya harapkan rancangan algoritma kriptografi saya termasuk *unbreakable cipher*. *Unbreakable cipher* merupakan sebuah *cipher* yang tidak dapat dipecahkan. Salah satu contohnya adalah *one-time pad* (OTP). Sehingga, *unbreakable cipher* memiliki tingkat kerahasiaan yang sempurna. Oleh karena itu, saya mencoba menganalisis apakah rancangan algoritma kriptografi saya masih dapat dipecahkan dengan metode Kasiski, atau cukup sulit dipecahkan bahkan tidak sama sekali dapat dipecahkan.

2. DASAR TEORI

Kriptografi adalah ilmu dan seni untuk menjaga kerahasiaan pesan dengan menyandikannya ke dalam bentuk yang tidak dapat dimengerti lagi maknanya [1]. Seiring dengan perkembangan jaman, kriptografi tidak hanya digunakan untuk merahasiakan pesan tetapi juga memberikan aspek keamanan lainnya.

Algoritma Kriptografi adalah aturan untuk enkripsi (*enciphering*) dan dekripsi (*deciphering*) [1]. Kekuatan kriptografi terletak pada kekuatan algoritmanya. Kekuatan algoritma kriptografi dilihat dari banyaknya kerja yang dibutuhkan untuk memecahkan data cipherteks menjadi plainteksnya. Kerja ini dapat diekivalenkan dengan waktu.

Inti dari kriptografi adalah menjaga kerahasiaan plainteks (atau kunci, atau keduanya) dari penyadap atau kriptanalis. Sebuah algoritma kriptografi dikatakan aman bila memenuhi tiga kriteria berikut [1]:

1. Persamaan matematis yang digunakan pada algoritma kriptografi sangat kompleks sehingga algoritma kriptografi tidak mungkin dipecahkan secara analitik
2. Biaya untuk memecahkan cipherteks melampaui nilai informasi yang terkandung di dalam cipherteks tersebut.
3. Waktu yang diperlukan untuk memecahkan cipherteks melampaui lamanya waktu informasi tersebut harus dijaga kerahasiaannya.

2.1 Terminologi Kriptografi

Beberapa istilah dan definisinya yang dipakai dalam kriptografi:

a. Pesan: data atau informasi yang dapat dibaca dan dimengerti maknanya. Nama lain dari pesan adalah plainteks.
b. Cipherteks: pesan yang sudah tersandi sehingga pesan tidak dapat dimengerti maknanya.
c. Kunci: parameter yang digunakan untuk transformasi enkripsi dan dekripsi.
d. Enkripsi: proses menyandikan plainteks menjadi cipherteks. Enkripsi disebut juga *enciphering*.
e. Dekripsi: proses mengembalikan cipherteks menjadi plainteks. Dekripsi disebut juga *deciphering*.
f. Penyadap: orang yang mencoba menangkap pesan selama ditransmisikan. Nama lain penyadap adalah *enemy*, *eavesdropper*, *interceptor*, dan lain-lain.
g. Kriptanalisis: ilmu dan seni untuk memecahkan cipherteks menjadi plainteks tanpa mengetahui unci yang diberikan.
h. Kriptanalis: pelaku kriptanalisisis, yaitu orang yang memecahkan cipherteks menjadi plainteks.

2.2 Enkripsi dan Dekripsi

Pada Kriptografi, proses enkripsi dan dekripsi dapat dituliskan dalam notasi matematis. Misalkan:

$$P = plainteks$$
$$C = cipherteks \quad (1)$$

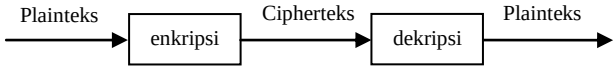
Fungsi enkripsi yang memetakan P ke C adalah:

$$E(P) = C \quad (2)$$

Fungsi dekripsi yang memetakan C ke P adalah:

$$D(C) = P \quad (3)$$

Gambar proses enkripsi dan dekripsi:

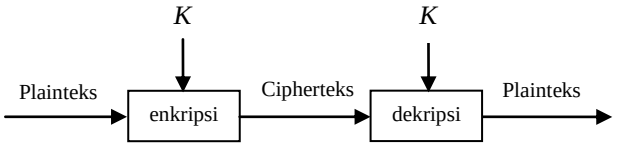


Gambar 1. Proses enkripsi dan dekripsi

Jika enkripsi dan dekripsi menggunakan kunci *K*, maka fungsi enkripsi dan dekripsi menjadi:

$$E_K(P) = C$$
$$D_K(C) = P \quad (4)$$

Gambar proses enkripsi dan dekripsi yang masing-masing menggunakan kunci K adalah:



Gambar 2. Proses enkripsi dan dekripsi dengan kunci *K*

2.3 *Cipher* Substitusi

Cipher substitusi termasuk algoritma kriptografi klasik. Idenya adalah menggantikan sebuah atau lebih huruf pada plainteks dengan sebuah atau lebih huruf pada plainteks dengan aturan tertentu. Aturan tersebut bergantung cara proses enkripsi dan dekripsi. *Cipher* subtitusi memiliki beberapa varian atau jenis. Jenis-jenis *Cipher* substitusi adalah:

1. *Cipher* substitusi abjad-tunggal (*monoalphabetic substitution cipher*). Jenis *cipher* substitusi ini sering juga disebut *cipher* substitusi sederhana.
   Ide *cipher* substitusi abjad-tunggal adalah menggantikan satu karakter pada plainteks menjadi

satu karakter pada cipherteks dengan aturan tertentu. Fungsi *ciphering*-nya merupakan fungsi satu ke satu.

2. *Cipher* substitusi homofonik (*homophonic substitution cipher*)
   Ide *cipher* substitusi homofonik adalah menggantikan satu karakter pada plainteks menjadi satu atau lebih karakter pada cipherteks.
   Fungsi *ciphering*-nya merupakan fungsi satu ke banyak.
3. *Cipher* substitusi abjad-majemuk (*polyalphabetic substitution cipher*). Jenis *cipher* substitusi ini dapat disebut sebagai cipher substitusi ganda. *Cipher* substitusi abjad-majemuk merupakan *cipher* substitusi abjad-tunggal yang menggunakan kunci berbeda-beda. Karena itu, *cipher* substitusi abjad-majemuk memiliki periode *m*, *m* merupakan panjang kunci.
4. *Cipher* substitusi poligram (*polygram substitution cipher*).
   Ide *cipher* substitusi poligram adalah menggantikan sebuah blok karakter dengan sebuah blok cipherteks. Blok terdiri dari satu atau lebih karakter. Misalnya AAA diganti menjadi BCD tau PAP, dan lain-lain.

2.4 *Vigenere Cipher*

Vigenere Cipher merupakan salah satu *cipher* yang terkenal. *Vigenere Cipher* termasuk *cipher* substitusi abjad-majemuk. *Vigenere Cipher* dipublikasikan pada tahun 1856 dan dapat dipecahkan oleh Babbage dan Kasiski pada pertengahan abad 19.

Proses enkripsi dan dekripsi pada *Vigenere Cipher* menggunakan bujursangkar *Vigenere*. Kolom paling kiri menyatakan huruf-huruf kunci, baris paling atas menyatakan huruf-huruf plainteks. Bujursangkar *Vigenere* tidak dimasukkan ke dalam makalah karena ukuran gambarnya terlalu besar.

Misalkan huruf plainteks adalah *p*, huruf kunci adalah *k* dan proses enkripsi plainteks *p* menjadi cipherteks *c*. Cipherteks *c* didapatkan dari isi bujursangkar pada baris yang sama dengan *p* (huruf plainteks-nya) dan kolom yang sama dengan *k* (huruf kunci-nya). Sedangkan proses dekripsi cipherteks *c* menjadi plainteks *p*, plainteks *p* didapatkan dari baris yang sama dengan cipherteks *c* dengan kolom yang sama dengan kunci *k* yang bersesuaian.

2.5 Metode Kasiski

Metode Kasiski merupakan metode yang digunakan untuk memecahkan *Vigenere Cipher*. Metode Kasiski membantu untuk menemukan panjang kunci pada *Vigenere Cipher*. Prinsipnya adalah sebagai berikut:

1. Menemukan perulangan seluruh kriptogram yang terkandung pada cipherteks *C*.
2. Menghitung jarak antara kriptogram yang berulang tersebut.
3. Mencari atau menghitung faktor pembagi dari jarak tersebut. Faktor pembagi merupakan panjang kunci yang mungkin.
4. Tentukan irisan dari himpunan faktor pembagi tersebut. Nilai irisan mungkin adalah panjang kunci, misalnya *M*.

Setelah menemukan panjang kunci *M*, kita mencari kata kunci dengan menggunakan metode analisis frekuensi atau *exhaustive key search*. Tetapi, kita kelompokan dahulu cipherteks dengan aturan setiap huruf ke-*M* dikelompokan menjadi sebuah cipherteks C_n. Karena masing-masing huruf pada kelipatan *M* akan dienkripsikan dengan huruf kunci yang sama.

Setelah kata kunci ditemukan, kita dapat mendekripsikan chiperteks *C* menjadi sebuah plainteks dengan proses dekripsi pada *Vigenere Cipher*.

2.6 *Unbreakble Cipher*

Unbreakable cipher merupakan istilah yang digunakan untuk cipher yang tidak dapat dipecahkan. Syarat untuk merancang *unbreakable cipher* adalah:

1. Kunci harus dipilih secara acak
2. Panjang kunci harus sama dengan plainteks yang akan dienkripsikan.

One-time pad (OTP) merupakan salah satu cipher yang tidak dapat dipecahkan dan memiliki tingkat kerahasiaan yang sempurna. Kunci yang digunakan pada OTP dibangkitkan secara acak dan panjang kuncinya sama dengan panjang plainteks yang akan dienkripsikan. Kelemahan OTP adalah tidak dapat digunakan secara universal dan tidak praktis. Ketidakpraktisan OTP adalah:

1. Semakin besar ukuran plainteks, semakin besar ukuran kunci.
2. Kunci yang digunakan untuk enkripsi harus diberikan oleh pengirim pesan kepada penerima pesan karena kunci dibangkitkan secara acak. Usaha yang dilakukan untuk melindungi pengiriman kunci sangat besar.

3. RANCANGAN ALGORITMA KRIPTOGRAFI

Rancangan algoritma kriptografi saya menggunakan prinsip enkripsi dan dekripsi pada *Vigenere Cipher*. Saya menggunakan ide saya untuk meningkatkan tingkat keamanan algoritma kriptografi yang saya rancang. Idenya adalah menggunakan lebih dari sebuah kunci untuk melakukan enkripsi dan dekripsi. Saya akan menggunakan

tiga kunci untuk melakukan enkripsi dan dekripsi. Aturan dalam pemilihan tiga kunci tersebut adalah:

1. Misalkan kunci pertama adalah K_1 dengan panjang m_1, kunci kedua adalah K_2 dengan panjang m_2, dan kunci ketiga adalah K_3 dengan panjang m_3.
2. Panjang kunci K_1 tidak boleh sama dengan panjang kunci K_2, panjang kunci K_1 tidak boleh sama dengan panjang kunci K_3, dan panjang kunci K_2 tidak boleh sama dengan panjang kunci K_3.
3. Panjang kunci K_2 ditambahkan dengan panjang kunci K_3 tidak boleh sama dengan panjang kunci K_1.

Aturan pemilihan tiga kunci tersebut dapat dituliskan dalam notasi matematis:

$$m_1 \neq m_2,\ m_1 \neq m_3,\ m_2 \neq m_3$$
$$m_1 \neq (m_2 + m_3) \qquad (5)$$

Penggunaan tiga kunci saja masih dirasa kurang memberikan keamanan, maka saya menyisipkan teks semu, mislanya *T*, pada plainteks *P* yang akan dienkripsi. Tujuan penyisipan teks semu *T* adalah untuk mengelabuhi kriptanalis terhadap pesan asli atau plainteks *P* yang terkandung pada cipherteks *C* nantinya. Aturan penyisipan teks semu *T* pada plainteks *P* adalah:

Sisipkan satu karakter t_i dari teks semu T pada plainteks P setiap karakter ke-k. Ulangi penyisipan t_{i+1} untuk kelipatan k berikutnya. Jika seluruh karakter pada teks semu T sudah disisipkan pada plainteks P, ulangi penyisipan mulai dari i=0.

Contoh penyisipan teks semu *T* pada plainteks *P* dan *T* dan *P* merupakan teks berbahasa inggris. Misalkan teks semu *T* adalah *opinion*, dan plainteks *P* adalah *abcdefghijklmnopqrstuvwxyz*. Hasil penyisipan setiap karakter ke-1 adalah:

aobpcidneifognhoipjiknlimonnnoopppqirnsitounvowpxiynzi

Hasil penyisipan setiap karakter ke-3 adalah:

abcodefpghiijklnmnoipqrostunvwxoyz

Hasil penyisipan setiap karakter ke-5 adalah:

abcdeofghijpklmnoipqrstnuvwxyiz

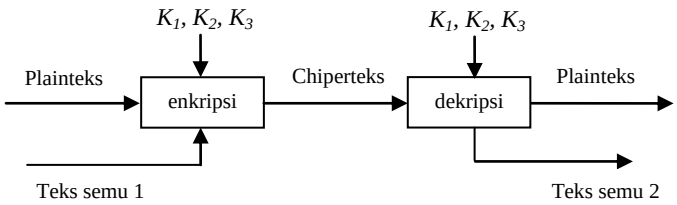
Proses enkripsinya akan menggunakan tiga buah kunci, yaitu K_1,K_2, dan K_3. Proses enkripsi plainteks *P* dengan penyisipan teks semu *T* setiap karakter ke-*k* adalah sebagai berikut:

1. Sisipkan teks semu *T* pada plainteks *P* setiap karakter ke-*k* maka didapatkan plainteks baru yang akan didekripsikan yaitu *Pnew*.
2. Enkripsikan seluruh karakter pada *Pnew* dengan kunci K_1 menggunakan prinsip enkripsi pada *Vigenere Cipher* sehingga mendapatkan cipherteks *C*.
3. Enkripsikan cipherteks *C* dengan 2 buah kunci K_2 dan K_3 dengan syarat kunci K_2 untuk menenkripsikan plainteks *P* dan kunci K_3 untuk menenkripsikan teks semu *T* (karakter *k*+1 dan kelipatannya pada ciphertkes *C*). Proses enkripsi dengan K_2 ataupun K_3 masih menggunakan proses enkripsi pada *Vigenere Cipher*. Hasil enkripsinya adalah *Cnew* dan *Cnew* merupakan hasil akhir enkripsi.

Proses dekripsi cipherteks C menjadi plainteks P harus mengetahui tiga buah kunci (K_1, K_2, dan K_3) yang digunakan pada proses enkripsi dan *k* penyisipan teks semu *T* pada plainteks *P* sebelum melakukan enkrispi. Jadi pengirim pesan harus memberi tahu K_1, K_2, K_3 dan *k*. Proses dekripsinya cipherteks *C* adalah sebagai berikut:

1. Dekripsikan cipherteks *C* dengan 2 buah kunci K_2 dan K_3 dengan syarat enkripsi karakter kelipatan *k*+1 menggunakan K_3 dan enkripsi karakter lainnya menggunakan K_2. Proses dekripsi dengan K_2 ataupun K_3 menggunakan proses dekripsi *Vigenere Cipher*. Hasil dekripsinya adalah *P*.
2. Dekripsikan seluruh karakter pada *P* dengan K_1. Proses dekripsinya masih menggunakan proses dekirpsi pada *Vigenere Cipher*. Hasil dekripsinya adalah *Pnew*.
3. Pisahkan *Pnew* menjadi 2 teks yang berbeda yaitu P_1 dan P_2. P_2 terdiri dari karakter *k*+1 dan kelipatannya dari *Pnew* dan karakter selain karakter *k*+1 dan kelipatannya pada *Pnew* dimasukkan pada P_1. P_1 merupakan plainteks asli dan P_2 merupakan teks semu yang disisipkan. P_2 dapat tidak dihiraukan setelah proses dekripsi.

Gambar proses enkripsi dan dekripsi rancangan algoritma kriptografi saya adalah:



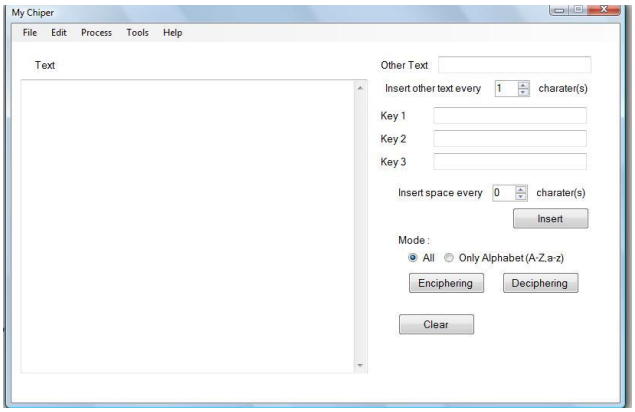
Gambar 3. Proses enkripsi dan dekripsi rancangan algoritma kriptografi saya.

4. PROGRAM

Saya juga membuat program untuk melakukan enkripsi dan dekripsi rancangan algoritma kritpografi saya. Saya menulis program dalam bahasa C# dan menggunakan

.NET Framework 3.5 SP1. *Tools* yang saya gunakan adalah Microsoft® Visual Studio Team System 2008.

Tampilan program yang saya buat adalah sebagai berikut:



Gambar 4. Tampilan Program

Program yang saya buat saya namakan *MyChiper*. *MyChiper* dapat melakukan proses enkripsi dan dekripsi sesuai dengan rancangan algoritma kriptografi saya. *MyChiper* menyediakan pilihan mode enkripsi dan dekripsi. Pilihan mode pertama adalah mode *All*, yaitu melakukan proses enkripsi atau dekripsi kepada seluruh karakter yang ada. Karakter yang akan dienkripsikan atau didekripsikan adalah karakter yang memiliki notasi ASCII 0-127, karakter dengan representasi ASCII selain 0-127 tidak akan diproses. Pilihan mode kedua adalah mode *Only Alphabet*, yaitu melakukan proses enkripsi dan dekripsi pada alfabet saja, yaitu A-Z dan a-z.

Untuk proses enkripsi, teks semu yang akan disisipkan pada plainteks *Text* dapat diisikan pada *TextBox Other Text*. Bilangan *k* atau penyisipan karakter teks semu pada plainteks setiap karakter ke-*k* dapat diisikan pada isian "*Insert other text every ... character(s)*".

Contoh keluaran program:

1. Plainteks:

> Like the Japanese fish problem, the best solution is simple.
> It was observed by L. Ron Hubbard in the early nineteen fifty's.
> "Man thrives, oddly enough, only in the presence of a challenging environment."- L. Ron Hubbard.

Teks semu: *fish*
Penyisipan teks semu setiap 2 karakter
Kunci 1: *my*
Kunci 2: *mine*
Kunci 3: *yours*

Hasil enkripsi:

> Jgpics rcfcr Hpynsylccqrc pdgsqfc nrpmpzjsckc, rrfpc szccqrr qpmjssrcgmrl pgqs qcgkrnjpc.s
> cGrr upyqs mczqrcpptcsb czwr Jp. sPmcl rFspzzsypcb rglp rsfcc cryppjws lcglrcrpccsl cdgrdrpw'sq.c
> r
> "pKysl crfrpgptcsq,c mrbbpjws cclmrsepf,s mcljrw pgls rcfcr nppcsqcclarc pmds yc arfypjjsclcegrlep csltcgprmlpkcslrc."r- pJ.s Pcmlr Fpszszycpbr.

Hasil dekripsi (dengan kunci 1,kunci 2, kunci 3 yang sama dengan proses enkripsi dan aturan penyisipan 2 karakter) :

> Like the Japanese fish problem, the best solution is simple.
> It was observed by L. Ron Hubbard in the early nineteen fifty's.
> "Man thrives, oddly enough, only in the presence of a challenging environment."- L. Ron Hubbard.

2. Plainteks sama dengan plainteks pada contoh 1. Jadi plainteks tidak ditampilkan lagi.

Teks semu: *flower*
Penyisipan teks semu setiap 2 karakter
Kunci 1: *my*
Kunci 2: *mine*
Kunci 3: *yours*

Hasil enkripsi:

> Jgpicv ryfcg Hoynbylpcqvc ydggqfo nbpmpzjvcky, grfoc bzcpqrv qymjgsrogmbl pgqv qygkgnjoc.b
> pGrv uyyqg mozqbcpptcvb yzwg Jo. bPmpl vFsyzzgypob bglp rvfcy cgypojwb lpglvcryccgl odgbdrpw'vq.y
> g
> "oKybl prfvpgytcgq,o mbbbpjwv cylmgseof,b mpljvw yglg rofcb nppcvqcylagc omdb yp avfyyjjgcloegblep cvltygpgmlokcblrp."v- yJ.g Pomlb Fpszvzyypbg.

Hasil dekripsi dengan menggunakan kunci 1, kunci 2, kunci 3 yang sama dengan proses enkripsi dan penyisipan 2 karakter adalah:

> Like the Japanese fish problem, the best solution is simple.
> It was observed by L. Ron Hubbard in the early nineteen fifty's.
> "Man thrives, oddly enough, only in the presence of a challenging environment."- L. Ron Hubbard.

5. ANALISIS

Selain membuat program, saya juga menganalisis kekuatan algoritma dari rancangan algoritma kriptografi saya. Saya menganalisis kekuatan algoritma rancangan algoritma kriptografi saya dengan cara apakah cipherteks yang dihasilkan masih dapat dipecahkan dengan metode Kasiski atau tidak.

Dari contoh keluaran program, saya mendapatkan bahwa chiperteks yang dihasilkan memiliki keunikan tersendiri.

Misalkan saya memperhatikan kemunculan karakter p pada chiperteks contoh 1 dan contoh 2. Kemunculan karakter p pada chiperteks contoh 1 ada sebanyak 38 buah dan pada karakter ke-2, 12, 23, 32, 34, 44, 54, 65, 76, 84, 94, 95, 105, 107, 114, 119, 125, 134, 135, 146, 157, 163, 172, 174, 184, 195, 204, 214, 215, 225, 235, 247, 254, 258, 266, 269, 275, 282.

Sedangkan kemunculan huruf p pada chiperteks contoh 2 ada sebanyak 29 buah pada karakter ke- 2, 18, 32, 34, 49, 65, 79, 94, 95, 107, 109, 119, 125, 134, 140, 157, 168, 172, 184, 199, 214, 215, 230, 247, 254, 264, 269, 275, 282.

Sementara ini, saya dapat menyimpulkan bahwa hasil enkripsi plainteks *P* untuk penyisipan teks semu T_1 akan mendapatkan cipherteks C_1 yang berbeda dengan hasil enkripsi untuk plainteks *P* yang sama dengan penyisipan teks semu T_2 akan mendapatkan C_2.

Selanjutnya saya mencoba memperhatikan kemunculan karakter terbanyak pada cipherteks contoh 1 dan contoh 2. Karakter yang muncul paling banyak pada cipherteks contoh 1 adalah C dengan jumlah 52 buah. Sedangkan karakter yang muncul paling banyak pada cipherteks contoh 2 adalah G dengan jumlah 32 buah.

Tabel 1. Karakter yang muncul paling banyak pada ciphterteks contoh 1 dan cipherteks contoh 2

Urutan	Cipherteks contoh 1	Cipherteks contoh 2
1	C (jumlah = 52)	G (jumlah = 32)
2	R (jumlah = 39)	P (jumlah = 29)
3	P (jumlah = 38)	Y (jumlah = 28)
4	S (jumlah = 32)	C (jumlah = 24)
5	L (jumlah = 18)	B (jumlah = 23)
6	G (jumlah = 13)	V (jumlah = 19)

Ternyata karakter yang paling banyak muncul untuk urutan ke-1 hingga ke-6 pada cipherteks contoh 1 dan contoh 2 masih mengandung kemiripan. Karakter yang mirip adalah C,P, dan G walaupun jumlahnya tidak sama dan urutan tertingginya masih tidak sama.

Sementara ini, saya menyimpulkan bahwa kemunculan karakter yang paling banyak masih kurang bisa disamarkan. Tetapi jumlah karakter yang muncul paling banyak dapat disamarkan.

Kemudian faktor pembagi yang dimungkin menjadi panjang kunci sulit dicari. Pada cipherteks contoh 1, faktor pembagi dari kemunculan karakter C, P, dan G tidak dapat ditemukan. Kemunculan karakter C dan G tidak dituliskan pada makalah ini. Mungkin kriptanalis menganggap panjang kunci yang mungkin adalah 2 atau kelipatannya. Alasannya adalah 2 merupakan salah satu faktor pembagi yang paling sering ditemukan.

Selanjutnya saya akan memperhatikan *trigraph* yang sering muncul pada cipherteks contoh 1 dan contoh 2.

Tabel 2. *Trigraph* yang muncul paling banyak pada cipherteks contoh 1 dan cipherteks contoh 2.

Urutan	Cipherteks contoh 1	Cipherteks contoh 2
1	PCS(jumlah = 4)	BLP (jumlah = 3)
2	SQC (jumlah = 4)	JGP (jumlah = 2)
3	SRC (jumlah = 3)	BPM (jumlah = 2)
4	RCP (jumlah = 3)	PMP (jumlah = 2)
5	SRC (jumlah = 3)	MPL (jumlah = 2)

Trigraph yang muncul paling banyak pada cipherteks contoh 1 tidak sama dengan trigraph yang muncul paling banyak pada cipherteks contoh 2.

Misalnya saya perhatikan kemunculan *trigraph* PCS pada cipherteks contoh 1 muncul setiap 44, 76, 215, 247. Sedangkan kemunculan trigraph SQC pada chiperteks contoh 1 muncul setiap 68,159,177,217. Sedangkan kemunculan N-*graph* lainnya misalnya kemunculan 5-*graph* untuk SRCFC sebanyak 2 buah terletak pada posisi 5 dan 207 dan kemunculan 5-*graph* RCFCR sebanyak 2 buah pada posisi 6 dan 208. Jadi dimungkinkan panjang kuncinya adalah 202 dan 202 memang merupakan kelipatan 2.

Misalkan kriptanalis masih menganggap 202 merupakan panjang kunci, sekarang saya akan mencari panjang kunci sebenarnya. Kunci 1: *my* memiliki panjang 2, Kunci 2: *mine* memiliki panjang 4, Kunci 3: *yours* memiliki panjang 5. Karena penyisipan teks semu: *fish* sebanyak 2 karakter maka panjang kunci dari gabungan 3 kunci tersebut adalah 15 karakter didapatkan dari kata kunci:

Kunci 1 mymymymymymymym
Kunci 2 + miyneomiunermis
Kunci 3

Kunci 3 disisipkan setiap karakter ke-2 pada kunci 2 karena penyisipan teks semu juga dilakukan setiap karakter ke-2. Kata kunci gabungan kunci 1, kunci 2, dan kunci 3 tidak dituliskan dalam makalah ini.

Kesimpulan yang saya dapatkan adalah panjang kunci yang sebenarnya tidak mungkin ditemukan dengan metode Kasiski karena tidak ada faktor pembagi 15, panjang kunci sebenarnya, dari kandidat faktor pembagi yang ditemukan dengan metode Kasiski. Walaupun panjang kunci sudah ditemukan, kriptanalis masih harus menembak kata kunci

dan menghilangkan teks semu yang juga dienkripsikan. Kesulitan yang saya ciptakan dari rancangan algoritma kriptografi saya adalah:

1. Sulit untuk menentukan panjang kunci dengan metode Kasiski.
2. Kriptanalis masih harus menghilangkan teks semu yang ikut dienkripsikan.
3. Keamanan algoritma kriptografi ini tinggi karena dienkripsikan dengan 3 buah kunci. Jadi kriptanalis susah menembak kata kunci gabungan dari 3 kunci tersebut.

6. KESIMPULAN

Kesimpulan yang saya dapatkan dari makalah ini adalah:

1. Kekuatan algoritma kriptografi dapat ditingkatkan dengan cara memperumit proses enkripsi dan dekripsi algoritma kriptografi. Prosesnya dapat berupa proses matematis, penggunaan kunci lebih dari satu buah, atau penyisipan teks semu yang saya lakukan pada rancangan algoritma kriptografi saya.
2. Setiap algoritma kriptografi memiliki kelemahan dan kekuatan-nya masing-masing. Hal yang saya temukan dalam makalah ini adalah peningkatan kekuatan algoritma dengan menggunakan lebih dari satu kunci mengakibatkan usaha untuk mengirimkan kunci-kunci tersebut dari pengirim pesan kepada penerima pesan menjadi lebih besar atau mahal.
3. Kekuatan rancangan algoritma kriptografi saya adalah:
    a. Kunci yang digunakan tidak hanya satu, melainkan tiga buah. Dengan begitu, keamanan algoritma-nya juga meningkat.
    b. Penyisipan teks semu pada setiap karakter ke-*k* dapat membingungkan atau mengelabuhi kriptanalis.
4. Kelemahan rancangan algoritma kriptografi adalah:
    a. Kunci yang digunakan tiga kunci, maka usaha yang digunakan untuk mengirimkan tiga buah kunci lebih mahal daripada usaha untuk mengirimkan satu kunci.
    b. Pengirim pesan juga harus memberikan *n*, *n* merupakan bilangan untuk menyisipkan teks semu setiap *n*-kali karakter pada plainteks yang asli.

REFERENSI

[1] Munir, Rinaldi, “Diktat Kuliah IF5054 Kriptografi”, Program Studi Teknik Informatika, 2005.