Vide Noir Cipher

Adriano Milyardi - 13509010
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
Codezero91@yahoo.com

***Abstrak*— *Vigenère Cipher* merupakan salah satu metode enkripsi yang sudah dikenal cukup lama. Pada jaman perang dahulu, enkripsi digunakan untuk mengirimkan informasi-informasi vital dengan aman, pada jaman sekarang enkripsi tetap dimanfaatkan, yaitu untuk menjaga keamanan data, file yang dikhususkan untuk orang tertentu saja, dan masih banyak kegunaan-kegunaan penting lainnya. Namun *Vigenère Cipher* sendiri sudah berhasil dipecahkan, ada banyak sekali cara untuk memecahkan *Vigenère Cipher* dan cara yang paling populer contohnya adalah Metode Kasiski. Melihat hal tersebut, penulis ingin meningkatkan keamanan dari *Vigenère Cipher* yaitu dengan cara membuat sebuah algoritma baru yang menggabungkan berbagai unsur yang telah ada misalnya penerapan *One Time Pad*, yaitu untuk membuat enkripsi yang tidak dapat dipecahkan dengan kunci yang acak dan panjang kunci sama dengan panjang teks.**

Kata Kunci*—Enkripsi, Metode Kasiski, *One Time Pad*, *Vigenère Cipher

I. PENDAHULUAN

Kriptografi, secara umum dapat dikatakan sebagai ilmu dan seni dalam menjaga kerahasiaan dari berita (bruce Schneier – "Applied Cryptography"). Sebenarnya selain dari pengertian diatas, kriptografi dapat dikatakan pula sebagai ilmu yang mempelajari teknik-teknik matematika yang berhubungan erat dengan aspek keamanan informasi seperti kerahasiaan data, keabsahan data, integritas data, dan juga autentikasi data (A. Menezes, P. van Oorschot and S. Vanstone – "Handbook of Applied Cryptography"). Meskipun demikian, tidak semua aspek dari keamanan informasi dapat ditangani oleh kriptografi.

Istilah kriptografi (*cryptography*) berasal dari bahasa Yunani yaitu cryptos yang berarti rahasia (*secret*) dan graphein artinya tulisan (*writing*). Jadi kriptografi dapat diartikan sebagai tulisan rahasia (secret writing). Kriptografi tidak hanya ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan, integritas data, nirpenyangkalan, otentikasi tetapi juga sekumpulan teknik lain yang berguna.

Konsep dari kriptografi sendiri sudah lama digunakan oleh manusia misalnya pada peradaban Mesir dan Romawi walaupun masih sangat sederhana. Prinsip-prinsip yang mendasari kriptografi yakni:

- *Confidelity* (kerahasiaan) adalah layanan yang berguna agar isi pesan yang dikirimkan tetap rahasia dan tidak diketahui oleh pihak lain (kecuali pihak pengirim, pihak penerima / pihak-pihak memiliki ijin). Umumnya hal ini dilakukan dengan cara membuat suatu algoritma matematis yang mampu mengubah data hingga menjadi sulit untuk dibaca dan dipahami, sehingga hanya pihak otoritas yang memiliki kunci rahasia yang dapat mengubah data tersebut agar mudah dipahami.
- *Data integrity* (keutuhan data) merupakan layanan yang mampu mengenali/mendeteksi adanya manipulasi (penghapusan, pengubahan atau penambahan) data yang tidak sah (oleh pihak lain).
- *Authentication* (keotentikan) merupakan layanan yang berhubungan dengan identifikasi. Dua pihak yang saling berkomunikasi harus saling memperkenalkan diri. Informasi yang dikirimkan melalui kanal harus diautentikasi keaslian, isi datanya, waktu pengiriman, dan lain-lain.
- *Non-repudiation* (anti-penyangkalan) yaitu layanan yang dapat mencegah suatu pihak untuk menyangkal aksi yang dilakukan sebelumnya (menyangkal bahwa pesan tersebut berasal dirinya).

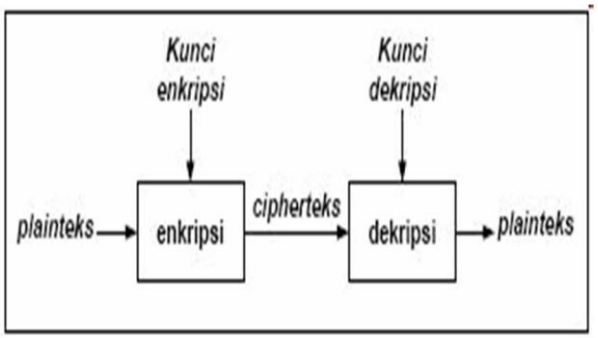
Berbeda dari kriptografi klasik yang menitikberatkan kekuatan pada kerahasiaan dari algoritma yang digunakan (apabila algoritma yang digunakan telah diketahui oleh orang lain maka pesan sudah pasti "bocor" dan dapat diketahui isinya oleh siapa saja yang mengetahui algoritma tersebut), kriptografi modern lebih fokus pada kerahasiaan dari kunci yang digunakan pada algoritma tersebut sehingga algoritma tersebut dapat saja disebarkan ke kalangan masyarakat umum tanpa takut kehilangan kerahasiaan bagi para pemakainya.

Berikut adalah istilah-istilah yang digunakan dalam bidang kriptografi :

- **Plaintext** (M) adalah pesan yang hendak dikirimkan (berisi data asli), yang dapat dimengerti langsung oleh manusia.
- **Ciphertext** (C) adalah pesan ter-enkrip (tersandi) yang merupakan hasil enkripsi, tidak bisa langsung dimengerti manusia.

- **Enkripsi** (fungsi E) adalah proses pengubahan *plaintext* menjadi *ciphertext*.
- **Dekripsi** (fungsi D) adalah proses kebalikan dari enkripsi yakni mengubah *ciphertext* menjadi *plaintext*, sehingga hasilnya berupa data awal yang asli.
- **Kunci** adalah suatu bilangan yang dirahasiakan yang digunakan dalam proses enkripsi dan dekripsi.

Kriptografi sendiri memililki dua proses utama yakni proses enkripsi dan proses dekripsi. Seperti yang telah dijelaskan di atas, proses enkripsi mengubah *plaintext* menjadi *ciphertext* (dengan menggunakan kunci tertentu) sehingga isi informasi pada pesan tersebut sukar dimengerti.



Gambar 1. Diagram proses enkripsi dan dekripsi algoritma simetris

Kunci memegang peranan yang sangat penting dalam proses enkripsi dan dekripsi (selain juga algoritma yang digunakan) sehingga kerahasiaannya sangatlah penting, apabila kunci tersebut berhasil diketahui oleh orang lain, maka isi dari pesan dapat terbongkar.

Secara matematis, proses enkripsi merupakan pengoperasian fungsi E (enkripsi) menggunakan e (kunci enkripsi) pada M *(plaintext)* sehingga dihasilkan C *(ciphertext)*, notasinya :

$$\mathbf{E_e(M) = C} \qquad (1)$$

Sedangkan untuk proses dekripsi, merupakan pengoperasian fungsi D (dekripsi) menggunakan d (kunci dekripsi) pada C *(ciphertext)* sehingga dihasilkan M *(plaintext)*, notasinya :

$$\mathbf{D_d(C) = M} \qquad (2)$$

Di mana fungsi dekripsi D memetakan *ciphertext* P ke plainteks awalnya Adapun fungsi *cipher* yang merupakan komposisi antara enkripsi dan dekripsi dapat ditulis sebagai berikut:

$$\mathbf{D_d(E_e(M)) = M} \qquad (3)$$

II. VIGENÈRE CIPHER

A. Enkripsi dan Dekripsi

Vigenère cipher merupakan metode enkripsi tulisan yang pertama kali diciptakan oleh seorang yang bernama Blaise de Vigenère pada tahun 1553 dalam bukunya yang berjudul "*La cifra del. Sig. Giovan Battista Bellaso"*. Metode ini berjenis kunci simetrik dengan menggunakan metode subtitusi polialfabetik dengan mengaplikasikan *Caesar Cipher* yaitu dengan mensubtitusikan huruf pada plainteks dengan kata kunci yang berpadanan letaknya.

Secara matematis, fungsi enkripsi dan fungsi dekripsi dengan mengaplikasikan *Vigenère cipher* dapat dituliskan sebagai berikut:

$$\boldsymbol{C_i = (P_i + K_i) \bmod 26} \qquad (4)$$
$$\boldsymbol{P_i = (C_i - K_i) \bmod 26} \qquad (5)$$

Di mana,
Pi : karakter plainteks
Ki : karakter kunci
Ci : karakter chiperteks

Nomor 4 merupakan fungsi enkripsi yang menghasilkan *ciphertext* (C_i) dan nomor 5 adalah fungsi dekripsi yang menghasilkan *plaintext* (P_i).

Enkripsi dan dekripsi pada *Vigenère cipher* menggunakan bantuan tabel berukuran 26 x 26 berisi huruf alfabet yang biasa disebut sebagai *tabula recta*.

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

Gambar 2. *Tabula recta*

Pada kasus dimana panjang kunci lebih pendek daripada panjang plainteks, maka kunci diulang secara periodik. Misalkan panjang kunci adalah 20 karakter, maka 20 karakter pertama dienkripsi dengan (4) setiap karakter ke-*i* menggunakan kunci k_i. Untuk 20 karakter

berikutnya, kembali menggunakan pola enkripsi yang sama. Misalkan untuk sebuah plainteks alphabet yang berisikan tulisan "livealifestayfoolish" dan kunci yang bertuliskan "test", maka bisa didapat ciphertext dengan menggunakan (4) sebagai berikut:

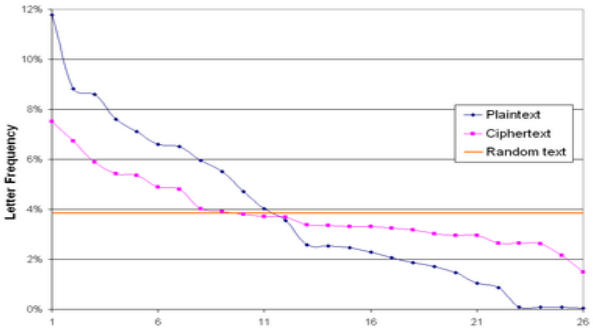
Pi: livealifestayfoolish
Ki: testtesttesttesttest
Ci: emnxtpayxwltrjghemka

Dari hasil enkripsi di atas dapat dilihat bahwa belum tentu satu huruf yang sama dienkripsi menjadi huruf yang sama pula, oleh karena hal ini, algoritma *Vigenère cipher* dapat disimpulkan cukup berhasil mengurangi frekuensi kemunculan pengulangan huruf dalam sebuah teks.

B. Kriptanalisis pada Vigenère cipher

Kita tinjau telebih dahulu ide utama dari *Vigenère cipher*, seperti semua polialfabetik enkripsi lainnya adalah untuk menyamarkan kemunculan frekuensi huruf cipherteks yang bersesuaian dengan plainteks. Sebagai contoh, apabila dengan analisis frekuensi ditemukan bahwa L adalah huruf terbanyak dalam cipherteks dan kriptanalis mengetahui jika plainteks menggunakan Bahasa Inggris maka kriptanalis dapat menyimpulkan bahwa huruf L merupakan huruf E yang telah terenkripsi karena dalam Bahasa Inggris huruf E adalah huruf yang paling sering muncul, setelah menemukan huruf E tersebut maka sisa dari teks akan dapat dipecahkan, hanya masalah waktu saja. Maka, dengan digunakannya *Vigenère cipher* hal ini dapat dikurangi karena satu huruf bisa saja dienkripsi menjadi beberapa huruf lain.

Kelemahan utama dari *Vigenère cipher* terletak pada kuncinya yang berulang, apabila seorang kriptanalis dapat menebak dengan tepat panjang kunci, maka cipherteks akan dengan mudah dipecahkan. Memang benar bahwa penggunaan *Vigenère cipher* dapat mengurangi keterlihatan frekuensi pengulangan huruf, akan tetapi, meksipun demikian sedikit dari pola kalimat akan tetap tersisa, hal ini dapat terjadi apabila kunci terlalu pendek atau monoton berulang.



Gambar 3. Grafik perbandingan frekuensi kemunculan huruf

Pada tahun 1863, seorang bernama Friedrich Kasiski dapat memecahkan kode enkripsi dengan Vigenère cipher yang akhirnya namai Metode Kasiski atau *Kasiski Examination* atau *Kasiski Test*, yaitu dengan mengambil kelemahan sisa pola yang berulang, misalnya sebuah plainteks hendak dienkripsi dengan kunci sepanjang 4 karakter yaitu "ABCD":

Pi: **CRYPTO**ISSHORTFOR**CRYPTO**GRAPHY
Ki: ABCDABCDABCDABCDABCDABCDABCD
Ci: **CSASTP**KVSIQUTGQU**CSASTP**IUAQJB

Dari contoh diatas dapat dilihat bahwa kata **CRYPTO** secara tidak sengaja terenkripsi menjadi kumpulan kata yang sama yaitu **CSASTP**. Maka secara intuitif kita dapat menyimpulkan bahwa kata **CRYPTO** secara kebetulan dienkripsi dengan sekuens kunci yang sama. Metode Kasiski menyimpulkan bahwa jarak antara kedua kumpulan huruf pada suatu cipherteks memiliki kemungkinan besar adalah merupakan kelipatan dari panjang kunci yang digunakan. Dalam contoh di atas jarak antara kedua kumpulan huruf **CSASTP** adalah 16 karakter, maka bisa ditarik kesimpulan kunci memiliki panjang karakter sebesar 1, 2, 4, 8, atau 16. Dan hipotesis tersebut benar karena panjang kunci adalah 4 karakter yaitu "ABCD".

Metode lain untuk kriptanalisi *Vigenère cipher* adalah penghitungan analisis frekuensi. Sekali waktu panjang kunci enkripsi diketahui, maka cipherteks dapat dipecahkan dengan membaginya ke dalam beberapa kolom dengan masing-masing kolom berkorelasi dengan huruf pada kunci. Tiap kolom berisi plainteks dan kemudian hanya dibutuhkan penggeseran untuk mencoba kunci apa yang tepat dan cipherteks telah terdekripsi.

III. ALGORITMA PENDUKUNG

One Time Pad

One Time Pad merupakan algoritma kriptografi yang ditemukan oleh Major Joseph Mauborgne pada tahun 1917, yang sampai sekarang merupakan algoritma kriptografi sempurna yang belum dapat dipecahkan. Algoritma ini masih termasuk ke dalam algoritma kriptografi simetri.

Agar cara-cara kriptanalisis di atas dapat dihindari, terutama Metode Kasiski, konsep *Vigenère Cipher* dapat diperkuat menjadi sebuah *cipher* yang tidak dapat dipecahkan atau *unbreakable cipher*. Kecuali tentunya menggunakan jika kriptanalis menggunakan metode *brute force* yang selalu menghasilkan jawaban, maka pengirim pesan harus memperhatikan berbagai macam hal, salah satunya bahwa kunci harus diacak dan panjang kunci harus sama dengan panjang plainteks. Kedua syarat tersebut mengakibatkan plainteks yang sama tidak selalu menghasilkan cipherteks yang sama. Akibat dari panjang kunci yang sama dengan plainteks maka tidak ada pengulangan dari kunci yang menjadi kelemahan utama.

One Time Pad berisi deretan karakter-karakter kunci

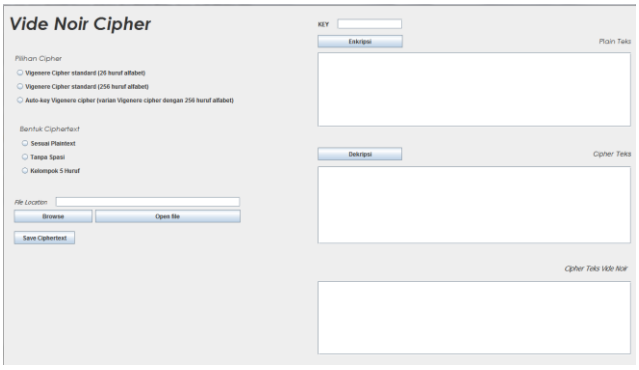
yang dibangkitkan secara acak. Satu *pad* hanya digunakan sekali saja untuk mengenkripsi pesan, setelah itu pad yang telah digunakan dihancurkan supaya tidak dipakai kembali untuk mengenkripsi pesan yang lain. Kunci itu hanya dimiliki oleh pembuat pesan dan penerima pesan.

Metode ini pada akhirnya membuat sebuah cipherteks memiliki resistansi penuh terhadap Metode Kasiski yang memanfaatkan pola pengulangan huruf untuk mendekripsinya. Meskipun *One Time Pad* merupakan algoritma yang aman, namun algoritma ini juga kelemahan yang cukup fatal yang mengakibatkan tidak banyak digunakan dalam praktek, yaitu tidak mangkus, karena panjang kunci sama dengan panjang pesan, apabila pesan yang dikirim cukup panjang, maka kunci juga akan mengikuti panjang pesan tersebut sehingga membuat kunci tersebut sulit untuk disimpan, selain itu pendistribusian kunci juga akan merepotkan yaitu bagaimana kunci tersebut dikirmkan secara aman dan tentunya tidak mungkin melalui jalur yang sama dengan jalur pengiriman pesan. Kunci harus dikirim karena pembangkitan kunci adalah acak dan sepanjang pesan serta yang mengetahui hanyalah sang pembuat pesan, agar penerima pesan bisa menerima pesan dengan baik, maka kunci tersebut harus dikirimkan melalui jalur yang aman dan biasanya keamanan itu relatif mahal dan lambat.

IV. VIDE NOIR CIPHER

Vide Noir cipher adalah algoritma yang dibuat oleh penulis yang merupakan pengembangan dari *Vigenère cipher* dengan menggabungkan beberapa konsep, yaitu pembangkitan kunci secara random dan *One Time Pad*. Algoritma ini memiliki ide utama sebagai berikut:

- Pembangkitan kunci secara acak dengan mengambil karakter acak dari plainteks yang ada.
- Pembangkitan kunci menerapkan prinsip *One Time Pad*, yaitu panjang dari kunci sama dengan panjang teks.
- Setelah teks dan kunci dibuat, proses selanjutnyasama dengan *Vigenère Cipher*. Hasil ciphertext dari vigenere cipher tersebut lalu diubah ke bentuk ASCII-nya.



Gambar 4. Tampilan program *Vide Noir Cipher*

A. Pembangkitan Kunci

Tahap awal dari enkripsi dengan *Vide Noir Cipher* ini adalah pembangkitan kunci dengan menggunakan fungsi random yang ada pada bahasa pemrogaman Java. Bilangan yang dirandom memiliki range dari bilangan pada posisi pertama sampai bilangan pada posisi terakhir yang ada pada plain teks.

Panjang kunci yang dibangkitkan harus sama dengan panjang dari plainteks yang hendak dienkripsi karena bertujuan menerapkan konsep *One Time Pad*.

B. Perubahan bentuk ciphertext ke bentuk ASCII

Setelah kunci berhasil dibangkitkan secara random, tahap berikutnya adalah proses enkripsi *Vigenère cipher*. Dari proses enkripsi tersebut akan dihasilkan ciphertext yang tidak dapat dimengerti langsung oleh manu.

Setelah ciphertext berhasil didapatkan, masing-masing karakter dari ciphertext tersebut akan dikonversi ke bentuk ASCII-nya dengan tiga digit angka.

V. PERBANDINGAN KEAMANAN KEDUA ALGORITMA

Agar dapat melihat perbedaan tingkat keamanan antara kedua algoritma, dapat digunakan sebuah sumber plainteks yang sama yang dienkripsi dengan kedua algoritma tersebut. Kemudian akan dilakukan kriptanalisi terhadap kedua cipherteks yang dihasilkan agar dapat membandingkan tingkat keamanan dari kedua algoritma tersebut.

A. Plainteks

Romance of the Three Kingdoms

Romance of the Three Kingdoms, written by Luo Guanzhong in the fourteenth century, is a historical novel based on the events in the turbulent years near the end of the Han Dynasty and the Three Kingdoms era of Chinese history, starting in one hundred and sixty nine and ending with the reunification of the land in two hundred eighty.

The story (part historical, part legend, and part myth) chronicles the lives of feudal lords and their retainers, who tried to replace the dwindling Han Dynasty or restore it. While the novel actually follows literally hundreds of characters, the focus is mainly on the three power blocs that emerged from the remnants of the Han Dynasty, and would eventually form the three states of Wei, Shu, and Wu. The novel deals with the plots, personal and army battles, intrigues, and struggles of these states to achieve dominance for almost a hundred years. This novel also gives readers a sense of how the Chinese view their history in a cyclical lens. The famous opening lines of the novel summarize this view: It is a general truism of this world that anything long divided will surely unite, and anything long united will surely divide.

Romance of the Three Kingdoms is acclaimed as one of the Four Great Classical Novels of Chinese literature; it

has a total of eight hundred thousand words and nearly a thousand dramatic characters (mostly historical) in one hundred twenty chapters. The novel is among the most beloved works of literature in East Asia, and it is arguably the most widely read historical novel in late imperial and modern China.

B. Cipherteks dengan Vigenère Cipher

Cipherteks di bawah adalah hasil pengenkripsian dengan *Vigenère cipher* biasa yang menggunakan kata kunci "handynasty".

Yozhnpl os ahr Ahele Xpntkozz

Rbtaaje bm tul Tuyer Riandbts, jyigaea iy Ybo Tbaaghbug vu tul fbbrgleaah plngbrl, ps n oifaoepcns nbcey iafld bu tul eilngz ia ahr aueiuylng fenys alae ahr lnq vf goe Uhn Qfnnztl hnq ahr Ahele Xpntkozz eeh os Jhvuefl hvztbyy, faaeaian ia vnr ouakrrk aak svetl uial aak eakian wvah goe eluapfvjagpoa vf goe yhnq pn gdo ubnqyeq litotl.

Ahr ztbyy (chrg oifaoepcns, pnyt ylgrud, nud chrg tygo) cuyoapcyls goe ypvrz os mehkay soeks nud goevy rraavueez, wuv tepeq ao elpyhcr ahr kwvudypnt Oaa Kyahsgf oe yefaoel ig. Dhvse goe avvrs apaunsll moysojz lvaeehlyf hhudeldf vf poaehcglrf, ahr mopbs vz mnpnyf oa ahr ahele cvwry byvcf ahna ezlrtld syoz ahr yezuaaas bm tul Hnu Dluafay, nud jvuyk eilngbaysy svrz ahr ahele faagls bm Wrp, Sub, aak Wh. Ahr uoill qlayz wvah goe csogz, prysbuay hnq hrzf bnatyls, vutepghls, nud farhngyls bm tulsr ztnaef ao njhvlvr kozpnnucr moe hlzvsg h hhudeld llaez. Tups avvrs ayzo tpvrz rrhdrys n zeaze bm hbd tul Cupnrze ipej ahrpr upsgvrl pn n jypsiphl ylnf. Ahr mazvuf vpruian lvuef vf goe avvrs shtmnyiml tups ipej: Pt vz a tlnryay arhpsz vf goif doesd goag hnlahvug yvnt kiipdrk wvsl fbrrsy huigl, aak aaftupnt soan uaptrk wvsl fbrrsy qpvvke.

Evmnucr vf goe Gorrl Kvugqvmf ps njcyhizld nz oal os ahr Mohy Gelag Jlnzsvjay Uoillf vf Poialsr siglrnauel; ig oaf h tbaay vf rpgua hhudeld goohzaak wbydf hnq uenyll h tuvufhnq krntagpc poaehcglrf (tofall oifaoepcns) ia vnr ouakrrk tjlngf cuhpglrf. Ahr uoill vz azvnt ahr tofa brsoild jvrxz os siglrnauel ia Lafa Afpa, nud va if hrtbaosy goe zvsg diqlll yenk hvztbyiphl avvrs ia sagl izweepay hnq toqlra Jhvua.

C. Cipherteks dengan Vide Noir Cipher

Sedangkan untuk penenkripsian dengan menggunakan *Vide Noir Cipher* dengan kunci yang sama adalah sebagai berikut:

08609810110511511610203211412003211211711903207210710510
12103207809711011104119102113010067114098118103118101032
11212103212012210103206910610011911903208911811811710411
81131190440321101071191161201151020321021150320721081190320
73105121114102115099112098032097118032102118105032110115
10210710211412111812011503211810910110711710711604403211
81020321080321081091191161151140981101151140321140981131140
09903210211710911510303209809703210410410503211410011812
10911103210709703210810410703210110211810010912210910310
03212111810110711903211810610511703211909912103210409711
90321181200321201181190320841091170320901061010971191071020
03211710511703210712210503206912210110111403207811810111
70610410310203210511811003210310203206711710911011410410
90321050990971080981051060440321200971051221030981141210320
11311303209911311203211109910111912110911903210510210003200
11309810510809903211810910310103210111511903210511810612200
10211103211412212011703209709712003211710510410309812211600
11210410210911311003210010203210710711803210111611611403200
11309803210410610203210211711510811810510403210510910312100
10810804601006510612003210610411810110303204011611510511100
03211711210612110311612211709711104403209811511611903210000
10409712011712004403209710311703211611511410903211310012000
11704103211810612210710110911209912210803211410810103210800
11810311810703210911503210611510211911112103211109812211100
10803211411811203211610412109809903211811412009709810210100
11910304403210611811603210411411210910303210910303211809700
11612110110310503210411710103211911210810410810211312110300
03207610009703207909910410810811510303210610203210510910200
12211309911403209811404603208211210012110403211011709703200
11211910510310103210411912010311710212211203212210310809900
10310611903212109812010311210212211910303210411709810810600
11911310003209910503212011511611410111211012111811804403200
10809811203210710211010511803211610703210211009810311611200
03211110303212011610803210912210110512103211410211010710500
03211111910210210803210311211110303210510510510110511810800
03210611411110503209711511503210711811711409712111912103200
10312003209709711503207610511203206609910110911912010904400
03210009810003211009911912210303211411810512212010411112100
11210903212112210112203211610410503209709910510112003210600
12209911911411803210412003208612009904403207411709804403200
11511810803209010804603206511211403211510410909710103211900
11811610810803210809810911203209710411903210010011812010000
04403211610912010911909811112203211211710803211711412011300
03211210011609810010511804403210111311610110510812011910800
04403212111511103209712011811710911711210810703212211403200
11610209911911403210610810110410409703211911503210109912100
11810110611903211911610009711810210110212003212011910203200
09911210211912110103211503210411911411909911210003211410500
11611411704603207611711811603210109811010512203210410110800
11503211110911611811803210810612111410810210803211103210900
11209811711303210712103210211911203211411210903207309711900
10611510010903210911010911003212009912111610703210911011800
10309810112103211909703210503211110211309909711211512203200
10212011710204603207610709703210211511310409810603211911600
10811810511412003211210511710110303210310603212011812003200
12211911810711603210210412111711010611211310803209712110300
11803212110812110105803207311003210512203211603212111410200
10411810809903210912111711610010003211112003210111210810600
03210610411912110203211211809911603211011811911910710711000
10303210110310812103211211611210911310510403209809711611200
03211810310712012111503211210211212010504403211610610703200
11811010309712112209811103211111809810303209811010911911500
10803211909812212203210812012209912212103211710710309810800
10904603201008610911312110811711203210011503212010911503200
07211210712110503207911611712112210210912203211711503211700
10110610110811611410110803211012203210212111903211811003200
10910811403208410011911403207510111909709803207211012111800
10610510611709703207911510312010011803210911003206910811600
10110811911403210012011909711410310910410112005903209810100
03210411011903212103210412211411110803211511503211811910400
10211603210010910110211909710003210710809912111911610211900
03210411510811711803211910211203211411209911710809903211100
03210311209811709710112110603210411811810712111011102032000
10911711511409710310911912211603204011610310810909710603200
09809811110309811212211812110804103210309803210410809903200
10412111411712212009803211610710409710810203209912209710400
12011510710304603208409710903210811111111810603210910603200
11110510410311003210310210903212212210010403210411911210200
11409911703210411111711311903210511903209711110911911611100
10312111211803209809803207209710011403208310611212104403200
10411010003211810303209811703210011712210810110211510303200
10412111203211611510411603210710911911511110303210711210200
11103210011211011411110511211610510003211409912210811203200

10111803211210110310503210510212011212211911610903210112110403212011111309912009803206511511211711604601 0

D. Kunci yang degenerate

ensifrbdswnsodteudsaieityldpvttabtesalcmssonioeheertoaeoseuwricoyeglocvsimoeieltmnuieltirratvnnleeeaeatlsgenvnreuuodnnoaenirltwcnsagllecsoitnareteifiddvudnthseosmmhwlraerhuvrrselsnandnrogtuneensaaneanpibuisnrlfhiintesidodlhirthitisaytlseietaefteigrsivrenhttdenttulnhmecapardrttgoioonryafieeeeearsnhctrohniesrvnhrfscrsadmscdsduthuatresatefentciwrenrvtyeaanlrsyneoltondniltrimaautleneatsafonofoahidtseweneeeonattduiuilaednleultzivoringclntyvivndunwcincthuemuuorusarsnentecyfoliaaoissnlodvltaenuuedsulfrlodlstnttiratemhtsneucrrgrnlrdtnionewencrieaawhlotriealdgsshtoeicyermeeodoarocodnaemenoneotlnnaaehvratrgcdndtszturnhsiidrhinftrwttrtatpttihasosheleiguiooophiualsodaisedwdanafdstyflieeagoehslmayyenrseodideearnaostfrsifrdtsiocetiglsacetllatetacsnnbrnseohtteieyrdufyohotoulocmwtyityiigtowolirfirevultffdnnnaonimhorsnsouthnsdwasethriehiaereahaoseeotmiaginnminshrhhrydddduiauahtsnsdelrthallraslidrntfncwocaniyddcaatsysmlueneefsiedmttnuvsheetwhvaihrroidhoahaedoiatootdiyoarcltiieyeyyslpnefooitueelhswrahmauchtllfainhrlshitenopcaensaifcydrahupbeltsdyicelrhenspdwagtnnttlaneyolyoaenrobyawsrcfwareoeetstleurdwsmelcdaeoninaielgeevyyugdknsaaetsibhsttplutwnnyrtyayotyyaeerityaodnshasaseotoatiyatryerowtthnyinllogserwyrladgeurpgtsconeyrtodalysrhyhaanntcddtreehiorlhepaoetoditlflwhvyarhriseoehewieeneatiliotbelelanygoylhhtwhsoertrrooeitraiewerlnsiwritircsuasheunfaeylledhunaldylsceenweieeeedtrhrnsiinaoyilteetrvounofhitaiiohrreeohaftlvsrntetynhrohslnhrlheetesraidefyeernoaenmnisboitaslaegcsreyrmlivhtartieemhetaaottdnannimtcsiuaeaniflsehaenrtaohtdnttloelinaeecaiyiupdehserodhtrmdneisuiennlsutnrnaeooenulaorwyfusdnrsnaecistspnth

E. Kriptanalisis pada Vigenère Cipher

Kriptanalisis akan dilakukan dengan memanfaatkan Metode Kasiski dan *Frequency Analysis*, sebagai alat bantu untuk memudahkan penghitungan penulis menggunakan program *CryptoHelper* yang disediakan di situs kuliah.

Dengan menggunakan konsep yang telah dituliskan di bab sebelumnya, dapat dilakukan kriptanalisis sebagai berikut:

- Analisis kemunculan
  - Salah satu Trigraph yang terbanyak muncul adalah "ahr" yaitu sebanyak 16 kali, masing-masing terdapat pada posisi 9, 129, 150, 174, 297, 405, 495, 513, 546, 594, 621, 792, 819, 1014, 1176, 1191.
  - Bigraph yang paling banyak muncul adalah "ah" sebanyak 27 kemuncululan di posisi 9,12,84,129,150,174,177,252,297,405,422,495,513,516,531,546,594,597,621,636,792,819,903,1014,1077,1176,1191
  - Huruf yang paling sering muncul adalah "a" dengan 138 kemunculan
- Penentuan panjang kunci
  - Urutan huruf terdekat dari Trigraph adalah 1176 dan 1191, sehingga kemungkinan panjang kunci adalah 1, 3, 5, dan 15 karakter.
- Asumsi dan nalar
  - Panjang kunci diasumsikan 3 karakter terlebih dahulu karena 1 karakter dianggap terlalu pendek dan tidak mungkin digunakan sebagai kunci.
  - Dari hasil analisis kemunculan didapat frekuensi kemunculan satu huruf dan dua huruf yang paling sering muncul, dan diketahui bahwa teks dalam Bahasa Inggris.
  - Mengelompokkan cipherteks karena panjang kunci sudah dapat ditentukan.
  - Asumsikan huruf yang paling sering muncul adalah "E" dan tiga huruf yang sering muncul adalah "THE"
  - Jika kita mengambil asumsi kata "ahr" sebagai "the" maka kita dapat mencoba mensubtitusi huruf-huruf yang ada dari kata tersebut.
  - Dengan menggunakan nalar, kita dapat menerka huruf-huruf lain yang belum lengkap.
  - Cara yang sama dilanjutkan terus menerus sampai akhirnya terdapat beberapa teks yang dapat dibaca dan diasumsikan sebagai kata tertentu dan didapatkan kunci yang sepanjang 3 karakter dan bertuliskan "han"
- Tingkat kebenaran
  - Panjang kunci adalah 3 karakter adalah benar.
  - Kunci dari ciphertext adalah "han".

Jadi, dapat disimpulakn bahwa pengenkripsian dengan *Vigenère Cipher* sekarang ini tidak lagi aman karena sudah adanya metode yang dapat digunakan secara efektif untuk melakukan penyerangan terhadap kunci, panjang kunci, dan langsung ke cipherteksnya. Dari Contoh di atas terbukti jika panjang kunci memengaruhi tingkat kesulitan kriptanalisis, semakin pendek akan semakin mudah untuk dipecahkan.

E. Kriptanalisis pada Vide Noir Cipher

Kriptanalisis dengan memanfaatkan Metode Kasisiki dan *Frequency Analysis* tidak memungkinkan untuk algoritma ini, karena diterapkannya *One Time Pad*.

Penulis telah menggunakan aplikasi yang sama yaitu *CryptoHelper* untuk melakukan analisis kemunculan, namun karena ciphertext akhir berupa angka program tidak berhasil mengeluarkan hasil frekuensi kemunculan.

Dengan memanfaatkan One Time Pad dapat dipastikan bahwa hasil enkripsi tidak dapat diserang oleh Metode Kasiski dan *Frequency Analysis* karena persentase pengulangan huruf yang kecil dan sulit untuk ditebak, terlebih lagi frekuensi kemunculan dari angka yang tidak mudah dihitung langsung menggunakan program *CryptoHelper*. Kriptanalis dipaksa harus membuat terlebih dahulu program yang membantu mengubah ciphertext yang dalam betuk ASCII ke bentuk hurufnya

terlebih dahulu untuk membantu proses kriptanalisis.

VI. KESIMPULAN

Melihat berbagai perbandingan dari hasil percobaan diatas, penulis dapat menganalisis dan menyimpulkan:

- Tingkat keamanan *Vide Noir Cipher* ini lebih baik jika dibanding *Vigenère cipher* karena terbukti memiliki ketahanan terhadap Metode Kasiski dan *Frequency Analysis.* Hal ini disebabkan karena algoritma ini mengurangi adanya pola kata yang berulang dan panjang kunci akan sulit untuk ditentukan.
- Karena ciphertext bebentu angka-angka, jauh lebih sulit untuk dimengerti oleh para kriptanalis, dan para kriptanalis perlu membuat program bantuan untuk mencari kunci.
- Kekurangan yang dimiliki oleh algoritma baru ini adalah membutuhkan biaya dan waktu yang cukup mahal untuk membangkitkan kunci acak karena digunakannya konsep *One Time Pad*. Apabila teks yang hendak dienkripsi cukup besar maka membutuhkan waktu untuk pembangkitan kunci yang cukup besar pula.
- Karena kunci yang disimpan sepanjang plaintext, maka butuh alokasi memori yang cukup besar juga.

VII. UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih terutama kepada Tuhan Yang Maha Esa karena anugerah yang diberikan-Nya makalah ini dapat diselesaikan. Penulis juga mengucapkan terima kasih kepada Bapak Ir. Rinaldi Munir, M.T. sebagai dosen pengajar kuliah IF3058 Kriptografi karena berkat kuliah dan referensi yang diberikan oleh beliau makalah ini dapat disempurnakan.

REFERENCES

[1] Murphy, Sean dan Fred Piper. 2002. *Cryptography: A Very Short Introduction*. Oxford University Press, ch 4.
[2] http://en.wikipedia.org/wiki/Vigenere_cipher
[3] http://supapri.wordpress.com/2009/07/10/otp-one-time-pad/

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, 19 Maret 2012

ttd

Adriano Milyardi - 13509010

LAMPIRAN

A. ALGORITMA Vide Noir CIPHER

```
public class Vigenerecipher extends kripto1 {

   private String key;
   private int mode = 0;

   public Vigenerecipher(String _plaintext, String _ciphertext, String _key, int _mode) {
      super(_plaintext, _ciphertext);
      key = _key.toLowerCase();
      mode = _mode;
   }

   public String Getkey(){
      return key;
   }

   public void Generatekey(){
      int Length = plaintext.length();
      Random generator = new Random();
      int i = 0;
      char[] Generatedkey = new char[Length];


      while (i<Length){
       int randomIndex = generator.nextInt(Length);
       if ((plaintext.charAt(randomIndex) > 96) && (plaintext.charAt(randomIndex) < 123)) {
       Generatedkey [i] = (char)plaintext.charAt(randomIndex);
       i++;
       }
      }
      key = String.valueOf(Generatedkey);
   }

   public String Converttoascii(){
     int[] code = new int[ciphertext.length()];
     String ascii = new String();
     int i = 0;

     while ((i<ciphertext.length())){
     code[i] = (int)ciphertext.charAt(i);
     System.out.println(code[i]);
     i++;
     }
     i = 0;
     while (i<ciphertext.length()){
     if (code[i] < 100){
     ascii += '0';
     ascii +=  code[i];
     } else {
     ascii +=  code[i];
     }
     i++;
     }
     System.out.println(ascii);
     return ascii;
   }
```

```
//Enkripsi 26 alphabet dengan key berulang
public void  encrypt26r() {
int i = 0;
int ec = 0;
int Length = plaintext.length();
char[] ciphertxt;

ciphertxt = new char[Length];
System.out.println(Length);

//Menangani bukan alphabet besar atau kecil
while (i<Length){
   if ((!((plaintext.charAt(i) > 96) && (plaintext.charAt(i) < 123))) &&
     (!((plaintext.charAt(i) > 64) && (plaintext.charAt(i) < 91))))
   {
      ciphertxt [i] = (char) plaintext.charAt(i);
   }
   //Menangani alphabet kecil
   else if (plaintext.charAt(i)>96) {
      int plntxt = plaintext.charAt(i) - 97;
      int kytxt = key.charAt(ec % key.length()) - 97;
      int ctxt = (plntxt + kytxt) % 26;
      ciphertxt[i] = (char) (ctxt + 97);
      ec++;
   }
   //Menangani alphabet besar
   else if ((plaintext.charAt(i)>64) && (plaintext.charAt(i)<96)) {
      int plntxt = plaintext.charAt(i) - 65;
      int kytxt = key.charAt(ec % key.length()) - 97;
      int ctxt = (plntxt + kytxt) % 26;
      ciphertxt[i] = (char) (ctxt + 65);
      ec++;
   }
   i++;
   }
ciphertext = String.valueOf(ciphertxt);
}

//Dekripsi 26 aphabet dengan key berulang
public void  decrypt26r() {
int i = 0;
int ec = 0;
int Length = ciphertext.length();
char[] plaintxt = ciphertext.toCharArray();


//Menangani bukan alphabet besar atau kecil
while (i<Length){
   if((!(ciphertext.charAt(i) > 96)) && (!(ciphertext.charAt(i) < 123 )) &&
     (!(ciphertext.charAt(i) > 64)) && (!(ciphertext.charAt(i) < 91 )))
   {
      plaintxt [i] = (char) ciphertext.charAt(i);
   }
   //Menangani alphabet kecil
   else if (ciphertext.charAt(i)>96) {
      int ctxt = ciphertext.charAt(i) - 97;
      int kytxt = key.charAt(ec % key.length()) - 97;
      int plntxt = (ctxt - kytxt);
      if (plntxt < 0) {
         plntxt = plntxt + 26;
```

```
        }
        plaintxt[i] = (char) (plntxt + 97);
        ec++;
    }
    //Menangani alphabet besar
    else if ((ciphertext.charAt(i)>64) && (ciphertext.charAt(i)<96)) {
        int ctxt = ciphertext.charAt(i) - 65;
        int kytxt = key.charAt(ec % key.length()) - 97;
        int plntxt = ctxt - kytxt;
        if (plntxt < 0) {
            plntxt = plntxt + 26;
        }
        plaintxt[i] = (char) (plntxt + 65);
        ec++;
    }
    i++;
    }
plaintext = String.valueOf(plaintxt);
}
```