PENGGABUNGAN *VIGENERE CIPHER* DENGAN *HILL CIPHER* PADA PENGKODEAN *PLAINTEXT* DENGAN KUNCI BERTAHAP

SKRIPSI

Oleh
Rizka Rahmawati
NIM 131010101028

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2017

PENGGABUNGAN *VIGENERE CIPHER* DENGAN *HILL CIPHER* PADA PENGKODEAN *PLAINTEXT* DENGAN KUNCI BERTAHAP

SKRIPSI

diajukan guna memenuhi tugas akhir dan memenuhi salah satu syarat
untuk menyelesaikan Program Studi Matematika (S1)
dan mencapai gelar Sarjana Sains

Oleh
Rizka Rahmawati
NIM 131810101028

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2017

PERSEMBAHAN

Skripsi ini saya persembahkan untuk :

1. orang tua saya Bapak Wartono dan Ibu Tri Sunaryati yang selalu memberikan doa dan kasih sayang serta cinta untuk putri tercintanya.
2. Seluruh dosen matematika dan guru dari sekolah dasar sampai perguruan tinggi yang telah memberikan ilmu dan membimbing dengan penuh kesabaran.
3. almamater tercinta Jurusan Matematika FMIPA Universitas Jember, SMAN 1 Genteng, SMPN 1 Genteng, SDN 1 Genteng, dan TK Pertiwi Genteng.
4. teman-teman Atlas'13 yang selalu membantu dan memberi dukungan.

MOTO

"*Everybody is a genius. But if you judge a fish by its ability to climb a tree, it will spend its whole life believing that it is stupid*".[1]

"*Just believe in yourself. Even if you don't, pretend that you do and at some point, you will*".[2]

[1] Albert Einstein
[2] Venus Williams

PERNYATAAN

Saya yang bertanda tangan dibawah ini :

Nama : Rizka Rahmawati

NIM : 131810101028

menyatakan dengan sesungguhnya bahwa karya ilmiah yang berjudul "Penggabungan *Vigenere Cipher* dengan *Hill Cipher* pada Pengkodean *Plaintext* dengan Kunci Bertahap" adalah benar-benar hasil karya sendiri, kecuali kutipan yang telah disebutkan sumbernya, belum pernah diajukan di institusi manapun, dan bukan karya jiplakan. Saya bertanggung jawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa ada tekanan dan paksaan dari pihak manapun serta bersedia mendapat sanksi akademik jika ternyata di kemudian hari pernyataan ini tidak benar.

Jember, Juni 2017

Yang menyatakan,

Rizka Rahmawati

NIM 131810101028

SKRIPSI

PENGGABUNGAN *VIGENERE CIPHER* DENGAN *HILL CIPHER* PADA PENGKODEAN *PLAINTEXT* DENGAN KUNCI BERTAHAP

Oleh

Rizka Rahmawati
NIM 131810101028

Pembimbing

Dosen Pembimbing Utama : Ahmad Kamsyakawuni, S.Si, M.Kom.

Dosen Pembimbing Anggota : Kusbudiono, S.Si,M.Si.

PENGESAHAN

Skripsi berjudul "Penggabungan *Vigenere Cipher* dengan *Hill Cipher* pada Pengkodean *Plaintext* dengan Kunci Bertahap" telah diuji dan disahkan pada :

hari, tanggal :

tempat : Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Tim Penguji:

Ketua,	Anggota I,
Ahmad Kamsyakawuni, S.Si, M.Kom.	Kusbudiono, S.Si, M.Si
NIP. 197211291998021001	NIP. 197704302005011001
Anggota II,	Anggota III,
Kosala Dwidja Purnomo, S.Si., M.Si.	Drs. Rusli Hidayat, M.Sc.
NIP. 196908281998021001	NIP. 196610121993031001

Mengesahkan
Dekan,

Drs. Sujito, Ph.D.

NIP. 196102041987111001

RINGKASAN

Penggabungan *Vigenere Cipher* dengan *Hill Cipher* pada Pengkodean *Plaintext* dengan Kunci Bertahap; Rizka Rahmawati, 131810101028; 61 Halaman; Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Di era yang terus berkembang ini, teknologi informasi juga ikut berkembang. Banyak kegiatan manusia yang dibantu dengan teknologi informasi, misalnya saja bertransaksi, berbisnis, atau bertukar pesan yang tentunya membutuhkan keamanan agar data informasi tidak diketahui oleh sembarang orang. Untuk itu dibutuhkan suatu teknik penyembunyian data yang disebut dengan kriptografi.

Dalam kriptografi sendiri, terdapat beberapa macam algoritma diantaranya yang cukup terkenal adalah *Vigenere Cipher*. *Vigenere Cipher* merupakan algoritma *polyalphabetic* dengan kunci simetris. Kunci pada algoritma *vigenere* berulang hingga panjangnya sama dengan *plaintext*. Namun perulangan kunci tersebut menyebabkan algoritma *vigenere cipher* dapat dipecahkan. Selain itu karakter yang dapat digunakan pada *vigenere cipher* berbatas pada huruf alfabet saja. Untuk mengatasi kelemahan ini, akan digunakan kunci baru yang dibentuk dari proses enkripsi kunci mengunakan algoritma *Caesar Cipher, Vigenere Cipher,* dan *Hill Cipher*. Selain itu untuk meningkatkan keamanan data, proses enkripsi *plaintext* akan digabungkan dengan algoritma *Hill Cipher*.

Pada penelitian ini, data yang akan digunakan yaitu data teks yang termasuk dalam *ASCII Printable Character*. Proses enkripsi dilakukan dengan membagi *Plaintext* menjadi *plaintext* ganjil dan *plaintext* genap. *Plaintext* ganjil akan dienkripsi dengan *Vigenere Cipher* sedangkan *plaintext* genap akan dienkripsi dengan *Hill Cipher*. Kunci yang digunakan yaitu *final key* yang dibentuk dengan menggunakan algoritma *Caesar Cipher, Vigenere Cipher,* dan *Hill Cipher*. *Final*

Key tersebut akan digunakan untuk proses enkripsi dan dekripsi. Hasil dari proses enkripsi kedua algoritma tersebut digabungkan kembali menjadi *ciphertext*.

Proses enkripsi mengasilkan karakter *ciphertext* yang acak dan tidak memiliki perulangan karakter. Selain itu, analisis frekuensi dan metode kasiski tidak mampu memecahkan kunci bahkan *plaintext* dari metode yang diajukan.

PRAKATA

Puji syukur kehadirat Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul "Penggabungan *Vigenere Cipher* dengan *Hill Cipher* Pada Pengkodean *Plaintext* Dengan Kunci Bertahap". Tugas akhir ini disusun guna memenuhi salah satu syarat untuk menyelesaikan pendidikan strata satu (S1) Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Pada kesempatan ini penulis mengucapkan terima kasih atas bantuan dan bimbingan dalam penyusunan tugas akhir ini, terutama kepada yang terhormat :

1. Drs. Sujito, Ph.D., selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember;
2. Ahmad Kamsyakawuni, S.Si, M.Kom., selaku Dosen Pembimbing Utama dan Kusbudiono, S.Si, M.Si., selaku Dosen Pembimbing Anggota;
3. Kosala Dwidja Purnomo, S.Si., M.Si. dan Drs. Rusli Hidayat, M.Sc. selaku Dosen Penguji yang telah memberikan kritik dan saran yang membangun demi kesempurnaan skripsi ini;
4. Orang tua tercinta Ibu Tri Sunaryati dan Bapak Wartono yang selalu memberikan dukungan dan doa;
5. Dosen dan Karyawan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember;
6. Sahabat "Housemate" dan "Atlas13" yang telah memberikan banyak kenangan dan dukungan.

Semoga bimbingan, bantuan, dan dorongan beliau dibalas berlipat ganda oleh Tuhan yang Maha Esa. Selain itu, penulis juga menerima segala kritik dan saran dari semua pihak demi kesempuranaan penyusunan tugas akhir ini. Akhirnya penulis berharap, semoga tugas akhir ini dapat bermanfaat.

Jember, Juni 2017

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

DAFTAR LAMPIRAN

Halaman

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Seiring dengan berkembangnya zaman, teknologi informasi pun ikut berkembang dengan pesatnya. Perkembangan teknologi yang semakin pesat memberikan perubahan besar terhadap penggunanya. Berbagai kegiatan manusia kini dibantu dengan teknologi informasi diantaranya bertukar pesan, bertransaksi, bahkan berbisnis yang tentunya membutuhkan keamanan untuk menjaga rahasia data informasi. Untuk mencegah terjadinya penyalahgunaan suatu data yang bersifat rahasia, maka dibutuhkan suatu teknik penyandian yang dapat menyembunyikan pesan sehingga pesan rahasia tidak dapat dibaca oleh sembarang orang.

Kriptografi merupakan ilmu mengenai teknik penyandian pesan. Menurut Schneier (1996), kriptografi merupakan ilmu dan seni untuk menjaga pesan agar tetap aman. Sedangkan menurut Munir (2006), kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan, integritas data, serta otentikasi. Berdasarkan pendapat tersebut dapat disimpulkan bahwa kriptografi merupakan ilmu yang mempelajari keamanan informasi secara matematis.

Algoritma *Vigenere Cipher* merupakan salah satu algoritma kriptografi dengan kunci simetris yang terkenal. *Vigenere Cipher* merupakan bentuk substitusi *polyalphabetic* dengan berbagai keterbatasan dimana hanya dapat digunakan untuk mengenkripsi teks. Teknik cipher ini menggunakan tabel vigenere untuk proses enkripsi abjad dengan kunci pada *Vigenere Cipher* berulang hingga panjang sama dengan *plaintext*. Perulangan pada kunci tersebut menyebabkan algoritma *Vigenere Cipher* dapat dipecahkan dengan analisis frekuensi maupaun metode kasiski.

Pada tahun 2016, Garg dkk telah membahas *Vigenere Cipher* dengan *Stream Cipher* dimana *plaintext* dibagi menjadi sisi ganjil dan genap kemudian sisi ganjil dienkripsi dengan *Vigenere Cipher* dan sisi genap dienkripsi dengan

Stream Cipher. Hasil dari penelitian tersebut dibandingan dengan algoritma *Vigenere Cipher* yang ada. Dari penelitian tersebut terlihat bahwa *Vigenere Cipher* dengan *Stream Cipher* lebih aman karena menggabungkan 2 algoritma dan telah memuat karakter yang lebih banyak. Namun apabila kriptanalis telah menemukan pola enkripsi serta metode yang digunakan akan dengan mudah menemukan kunci bahkan *plaintext*.

Pada penelitian ini, penulis ingin mengajukan suatu metode modifikasi algoritma *Vigenere* yaitu dengan menggabungkan *Vigenere Cipher* dan *Hill Cipher* dimana bentuk modifikasi terletak pada kunci. Kunci yang akan digunakan para proses enkripsi dan dekripsi disebut *final key* yang didapatkan dari proses enkripsi kunci awal dengan menggunakan metode *Caesar Cipher*, *Vigenere Cipher,* dan *Hill Cipher*. Penulis menggunakan *Hill Cipher* karena algoritma tersebut diciptakan agar tidak dipecahkan dengan analisis frekuensi yang mana berlawanan dengan *Vigenere* Cipher.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, maka rumusan masalah pada penelitian ini adalah:

a. Bagaimana mengenkripsi pesan menggunakan algoritma *Vigenere Cipher* dan *Hill Cipher* dengan modifikasi kunci?
b. Bagaimana mendekripsi pesan menggunakan algoritma *Vigenere Cipher* dan *Hill Cipher* dengan modifikasi kunci?
c. Bagaimana memodifikasi kunci sebelum digunakan untuk enkripsi?
d. Bagaimana analisis keamanan dari metode yang diajukan?

1.3 Batasan Masalah

Adapun batasan masalah pada penelitian ini yaitu:

a. Kunci matriks yang digunakan pada *Hill Cipher* adalah 3 x 3.
b. Modifikasi *Vigenere Cipher* dan *Hill Cipher* dilakukan pada kunci untuk membentuk kunci baru sebelum digunakan untuk enkripsi teks .

1.4 Tujuan Penelitian

Adapun tujuan pada penelitian ini yaitu:

a. Untuk mengenkripsi pesan menggunakan algoritma *Vigenere Cipher* dan *Hill Cipher* dengan modifikasi kunci.
b. Untuk mendekripsi pesan menggunakan algoritma *Vigenere Cipher* dan *Hill Cipher* dengan modifikasi kunci.
c. Untuk memodifikasi kunci sebelum digunakan untuk enkripsi.
d. Untuk menganalisis keamanan dari metode yang diajukan.

1.5 Manfaat Penelitian

Manfaat yang diharapkan pada penelitian ini yaitu dapat menggabungkan modifikasi algoritma *Vigenere Cipher* dan *Hill Cipher* pada sekali enkripsi. Selain itu penambahan karakter dan bentuk modifikasi yang diterapkan pada kuncinya dapat membuat penyembunyian data informasi menjadi lebih aman. Modifikasi algoritma tersebut membuat pesan menjadi lebih sulit untuk dipecahkan oleh metode serangan analisis frekuensi maupun metode kasiski.

BAB 2. TINJAUAN PUSTAKA

2.1 Kriptografi

Kriptografi (*Cryptography*) berasal dari bahasa Yunani, yaitu *kryptos* dan *grapho* yang artinya tulisan tersembunyi. Kriptografi adalah ilmu yang mempelajari untuk teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi, seperti kerahasiaan data, keabsahan data, integritas data, serta autentikasi data (Menezes dkk, 1996). Kriptografi dapat pula diartikan sebagai ilmu atau seni menjaga keamanan pesan.

Terdapat beberapa komponen yang ada pada kriptografi yaitu *plaintext*, *ciphertext*, *key*, dan *algorithm*. *Plaintext* merupakan data informasi yang dapat terbaca sedangkan *ciphertext* merupakan data yang sudah diubah menjadi tidak terbaca. *Key* merupakan kunci yang digunakan dalam proses kriptografi dan *algorithm* adalah metode yang digunakan untuk melakukan proses kriptografi. Proses pada kriptografi ada 2 yaitu enkripsi dan dekripsi. Dalam kriptografi, enkripsi adalah proses mengubah informasi menggunakan algoritma agar tidak terbaca siapapun kecuali mereka yang memiliki pengetahuan khusus atau yang biasanya disebut dengan kunci. Proses untuk membalikkannya disebut dengan dekripsi. Gambar 2.1 merupakan ilustrasi proses enkripsi dan dekripsi dalam kriptografi

Gambar 2.1 Aliran Proses Enkripsi dan Dekripsi

Berdasarkan perkembangannya, kriptografi terbagi menjadi 2 yaitu kriptografi klasik dan modern. Kriptografi klasik sudah digunakan sejak era komputerisasi dan kebanyakan teknik ini menggunakan kunci simetris. Metode yang digunakan yaitu substitusi dan transposisi. Teknik substitusi menggantikan karakter dalam *plaintext* menjadi karakter lain yang hasilnya adalah *ciphertext* sedangkan transposisi mengubah *plaintext* menjadi *ciphertext* dengan cara permutasi karakter. Kombinasi keduanya secara kompleks adalah yang melatar

belakangi terbentuknya berbagai macam algoritma kriptografi modern. Kriptografi modern memiliki tingkat kesulitan yang kompleks dan kekuatan kriptografinya ada pada *key* atau kuncinya. Algoritma ini menggunakan pengolahan simbol biner karena berjalan mengikuti operasi komputer digital. Sehingga membutuhkan dasar berupa pengetahuan terhadap matematika untuk menguasainya.

Kunci dalam kriptografi dibedakan menjadi 2 jenis yaitu simetris dan asimetris. Algoritma simetris menggunakan kunci yang sama untuk proses enkripsi maupun dekripsi sehingga disebut juga algoritma kunci tunggal. Selain itu algoritma ini disebut juga *private key* atau kunci rahasia karena kunci tersebut bersifat rahasia. Algoritma asimetris menggunakan kunci yang berbeda untuk proses enkripsi dan dekripsi. Kunci yang digunakan untuk proses enkripsi disebut *public key* dan dapat diketahui orang lain sedangkan kunci yang digunakan untuk dekripsi disebut *private key* dan bersifat rahasia.

2.2 ASCII (*American Standart Code for Information Interchange*)

American Standard Code for Information Interchange atau ASCII merupakan suatu standar international kode dan simbol yang bersifat universal. Kode ASCII digunakan komputer untuk menunjukkan teks. Terdapat 256 kode pada ASCII yang dikelompokkan kedalam beberapa bagian. Bagian yang sering digunakan disebut *ASCII Printable Character* (32-127) merupakan karakter yang terdapat pada keyboard komputer yang berupa huruf, angka, dan simbol. Kode ASCII *Printable Character* dapat dilihat pada Tabel 2.1.

Tabel 2.1 Kode *ASCII Printable Character*

ASCII Code	***Character***	***ASCII Code***	***Character***	***ASCII Code***	***Character***
32	Spasi	64	@	96	~
33	!	65	A	97	a
34	”	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d

ASCII Code	*Character*	*ASCII Code*	*Character*	*ASCII Code*	*Character*
37	%	69	E	101	e
38	&	70	F	102	f
39	’	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_	127	DEL

2.3 *Vigenere Cipher*

Vigenere Cipher merupakan algoritma dengan kunci simetris karena menggunakan kunci yang sama untuk proses enkripsi dan dekripsi. *Cipher* ini termasuk *cipher* klasik abjad majemuk karena setiap huruf dienkripsikan dengan huruf yang berbeda. *Vigenere cipher* termasuk bentuk pengembangan dari *Caesar cipher*. Penemu *Vigenere Cipher* Giovan Battista Bellaso menuliskan metode ini dalam bukunya yang berjudul *La Cifra del Sig* dan disempurnakan oleh diplomat Perancis Blaise de Vigenere sehingga banyak orang yang meyebutkan bahwa Vigenere penemu *cipher* tersebut sehingga algoritma ini dikenal dengan *Vigenere Cipher*.

Vigenere Cipher merupakan teknik kriptografi sederhana yang menggunakan karakter huruf sebagai kunci enkripsi. Algoritma ini juga merupakan *polyalphabetic substition cipher*. Karakter yang digunakan pada *cipher* ini adalah A, B, C, …, Z yang dikonversikan ke dalam angka yaitu 0, 1, 2, …, 25 seperti pada Tabel 2.2.

Tabel 2.2 Tabel Konversi Karakter ke Angka

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

P	Q	R	S	T	U	V	W	X	Y	Z
15	16	17	18	19	20	21	22	23	24	25

Kunci yang digunakan pada proses enkripsi *Vigenere Cipher* dituliskan secara berulang hingga setiap karakter pesan yang akan di enkripsi memiliki pasangan. Selanjutnya masing-masing karakter diubah menjadi angka dan disubstitusikan dengan kunci pasangannya. Hasil substitusi dari kunci dan *plaintext* di modulo 26 dan akan membentuk karakter baru. Proses modulo tersebut dilakukan untuk membatasi karakter karena pada *Vigenere Cipher* hanya digunakan 26 karakter.

Tabel 2.3 Contoh Enkripsi *Vigenere Cipher* (Kester, 2012)

Plaintext	C	R	Y	P	T	O	G	R	A	P	H	Y
	2	17	24	15	19	14	6	17	0	15	7	24
Key	L	U	C	K	L	U	C	K	L	U	C	K
	11	20	2	10	11	20	2	10	11	20	2	10
Ciphertext	N	L	A	Z	E	I	I	B	L	J	J	I
Mod 26	13	11	0	25	4	8	8	1	11	9	9	8

Pada contoh Tabel 2.3, karakter pesan "C" bernilai 2 dienkripsikan dengan kunci "L" bernilai 11 menghasilkan ciphertext "N" bernilai 13. Karakter baru tersebut didapatkan dari hasil substitusi nilai pada masing-masing karakter yaitu 2 dan 11 . Karakter "R" bernilai 17 dienkripsi dengan kunci "U" bernilai 20 menghasilkan ciphertext "L" bernilai 11. Karakter baru tersebut didapatkan dari substitusi nilai "R" dan "U" yaitu 17 dan 20 dengan di modulo 26 menghasilkan 11. Proses enkripsi pada *Vigenere Cipher* dituliskan seperti pada persamaan (2.1).

$$C_i = EK(P_i) = (P_i + K_i) mod\ 26 \quad (2.1)$$

Dan proses dekripsinya

$$C_i = (P_i + K_i)$$

$$(C_i - K_i) = P_i$$

Sehingga persamaan proses dekripsinya

$$P_i = DK(C_i) = (C_i - K_i) mod\ 26 \quad (2.2)$$

Dimana

P = $P_0 \dots P_n$ adalah *plaintext*

C = $C_0 \dots C_n$ adalah *ciphertext*

K = $K_0 \dots K_n$ adalah kunci

Permasalahan yang ada pada *Vigenere Cipher* yaitu kunci yang digunakan tidak aman karena menggunakan kunci yang berulang-ulang. Selain itu algoritma *Vigenere Cipher* ini hanya digunakan untuk karakter berhuruf besar / kapital sehingga tidak data digunakan untuk mengenkripsi simbol, huruf kecil, maupun angka (Kester, 2012).

2.4 *Hill Cipher*

Hill Cipher merupakan salah satu algoritma kriptografi dengan kunci simetris dan merupakan kripto *polyalphabetic* yang diperkenalkan oleh Lester S.Hill yang termuat dalam papernya "*Cryptography in Algebraic Alphabet*". Teknik kriptografi ini diciptakan dengan maksud menciptakan *cipher* yang tidak dapat dipecahkan dengan menggunakan analisis frekuensi. Setiap *plaintext* tidak selalu menghasilkan *ciphertext* yang sama dikarenakan menggunakan perkalian matriks pada proses enkripsi dan dekripsinya. *Hill cipher* menggunakan matriks *n* x *n* sebagai kunci untuk melakukan proses enkripsi dan dekripsi. Dasar teori matrik yang digunakan dalam *Hill Cipher* adalah perkalian dan invers matriks. Perhitungan matematis dasar dari teknik *Hill Cipher* adalah aritmatika modulo terhadap matriks dengan karakter yang digunakan yaitu A, B, C, …, Z yang dikonversikan ke dalam angka seperti pada Tabel 2.2.

Proses enkripsi maupun dekripsi pada *Hill Cipher* bergantung pada matriks kuncinya. Jika matrik kunci yang digunakan adalah 2 x 2 maka proses enkripsi dilakukan setiap 2 karakter pada *plaintext* dan jika matrik kuncinya 3 x 3 maka proses enkripsi dan dekripsi dilakukan setiap 3 karakter. Apabila jumlah karakter pesan yang akan dienkripsi kurang yaitu bersisa 1 atau 2 karakter, maka pada proses enkripsi pesan terakhir ditambahkan sebarang karakter untuk memenuhi perhitungan matriks. Jika matriks kunci adalah *K*, maka

$$K = \begin{bmatrix} K_{11} & K_{12} & \dots & K_{1n} \\ K_{21} & K_{22} & \dots & K_{2n} \\ \dots & \dots & \dots & \dots \\ K_{n1} & K_{n2} & \dots & K_{nn} \end{bmatrix}$$

Matriks *K* yang menjadi kunci harus merupakan matriks *invertible* yaitu *multiplicative invers* K^{-1} sehingga $K.K^{-1}=1$ karena kunci K^{-1} merupakan kunci yang digunakan untuk dekripsi.

Proses enkripsi pada *Hill Cipher* adalah

$$C = (K . P) \ mod \ 26 \qquad (2.3)$$

Proses enkripsi untuk $n= 3$ dapat dituliskan sebagai berikut :

$C_1 = (K_{11}.P_1 + K_{12}.P_2 + K_{13}.P_3) \ mod \ 26$

$C_2 = (K_{21}.P_1 + K_{22}.P_2 + K_{23}.P_3) \ mod \ 26$

$C_3 = (K_{31}.P_1 + K_{32}.P_2 + K_{33}.P_3) \ mod \ 26$

Contoh : Akan dienkripsikan *plaintext* "ILOVEALJABARGEOMETRI" menggunakan algoritma *Hill Cipher* dengan kunci $K=\begin{bmatrix}3 & 1 & 0\\ 1 & 1 & 1\\ 0 & 1 & 2\end{bmatrix}$ (Utama, 2015).

Penyelesaian : Konversi *plaintext* ke angka menjadi 9 7 13 15 25 4 16 2 9 3 18 8 22 24 6 14 9 16 7 7 24 dan karena kunci 3x3 maka untuk memenuhi perkalian matriks, bagi *plaintext* dengan masing-masing 3 karakter sehingga 9 7 13, 15 25 4, 16 2 9, 3 18 8, 22 24 6, 14 9 16, 7 7 24 (ditambahkan karakter yang sama dengan karakter terakhir untuk memenuhi perkalian)

Maka proses enkripsinya

$$\text{ILO} = C_1 = \left(\begin{bmatrix}3 & 1 & 0\\ 1 & 1 & 1\\ 0 & 1 & 2\end{bmatrix} \text{x} \begin{bmatrix}8\\ 11\\ 14\end{bmatrix}\right) \text{mod } 26 = \begin{bmatrix}9\\ 7\\ 13\end{bmatrix} = \text{JHN}$$

$$\text{VEA} = C_2 = \left(\begin{bmatrix}3 & 1 & 0\\ 1 & 1 & 1\\ 0 & 1 & 2\end{bmatrix} \text{x} \begin{bmatrix}21\\ 4\\ 0\end{bmatrix}\right) \text{mod } 26 = \begin{bmatrix}15\\ 25\\ 4\end{bmatrix} = \text{PZE}$$

$$\text{LJA} = C_3 = \left(\begin{bmatrix}3 & 1 & 0\\ 1 & 1 & 1\\ 0 & 1 & 2\end{bmatrix} \text{x} \begin{bmatrix}11\\ 9\\ 0\end{bmatrix}\right) \text{mod } 26 = \begin{bmatrix}16\\ 20\\ 9\end{bmatrix} = \text{QUJ}$$

$$\text{BAR} = C_4 = \left(\begin{bmatrix}3 & 1 & 0\\ 1 & 1 & 1\\ 0 & 1 & 2\end{bmatrix} \text{x} \begin{bmatrix}1\\ 0\\ 17\end{bmatrix}\right) \text{mod } 26 = \begin{bmatrix}3\\ 18\\ 8\end{bmatrix} = \text{DSI}$$

$$\text{GEO} = C_5 = \left(\begin{bmatrix}3 & 1 & 0\\ 1 & 1 & 1\\ 0 & 1 & 2\end{bmatrix} \text{x} \begin{bmatrix}6\\ 4\\ 14\end{bmatrix}\right) \text{mod } 26 = \begin{bmatrix}22\\ 24\\ 6\end{bmatrix} = \text{WYG}$$

$$\text{MET} = C_6 = \left(\begin{bmatrix}3 & 1 & 0\\ 1 & 1 & 1\\ 0 & 1 & 2\end{bmatrix} \text{x} \begin{bmatrix}12\\ 4\\ 19\end{bmatrix}\right) \text{mod } 26 = \begin{bmatrix}14\\ 9\\ 16\end{bmatrix} = \text{OJQ}$$

$$\text{RII} = C_7 = \left(\begin{bmatrix}3 & 1 & 0\\ 1 & 1 & 1\\ 0 & 1 & 2\end{bmatrix} \text{x} \begin{bmatrix}17\\ 8\\ 8\end{bmatrix}\right) \text{mod } 26 = \begin{bmatrix}7\\ 7\\ 24\end{bmatrix} = \text{HHY}$$

Sehingga didapatkan *ciphertext* "JHNPZEQUJDSIWYGOJQHHY"

Sedangkan proses dekripsinya adalah

$$C = (K . P) \ mod \ 26$$

$$K^{-1} . C = K^{-1} . K . P$$

$$K^{-1} . C = 1 . P$$

$$P = (K^{-1} . C) \ mod \ 26$$

Sehingga persamaan proses dekripsinya

$$P = (K^{-1} . C) \ mod \ 26 \qquad (2.4)$$

Dimana

P = *Plaintext*

C = *Ciphertext*

K = Kunci

Pada contoh halaman 10 didapatkan *ciphertext* "JHNPZEQUJDSIWYGOJQ HHY" sehingga untuk proses dekripsinya

$$K = \begin{bmatrix} 3 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix} \qquad K^{-1} = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 6 & -3 \\ 1 & -3 & 2 \end{bmatrix}$$

$$JHN = P_1 = \left(\begin{bmatrix} 1 & -2 & 1 \\ -2 & 6 & -3 \\ 1 & -3 & 2 \end{bmatrix} x \begin{bmatrix} 9 \\ 7 \\ 13 \end{bmatrix} \right) \text{mod } 26 = \begin{bmatrix} 8 \\ 11 \\ 14 \end{bmatrix} = \text{ILO}$$

$$PZE = P_2 = \left(\begin{bmatrix} 1 & -2 & 1 \\ -2 & 6 & -3 \\ 1 & -3 & 2 \end{bmatrix} x \begin{bmatrix} 15 \\ 25 \\ 4 \end{bmatrix} \right) \text{mod } 26 = \begin{bmatrix} 21 \\ 4 \\ 0 \end{bmatrix} = \text{VEA}$$

$$QUJ = P_3 = \left(\begin{bmatrix} 1 & -2 & 1 \\ -2 & 6 & -3 \\ 1 & -3 & 2 \end{bmatrix} x \begin{bmatrix} 16 \\ 20 \\ 9 \end{bmatrix} \right) \text{mod } 26 = \begin{bmatrix} 11 \\ 9 \\ 0 \end{bmatrix} = \text{LJA}$$

$$DSI = P_4 = \left(\begin{bmatrix} 1 & -2 & 1 \\ -2 & 6 & -3 \\ 1 & -3 & 2 \end{bmatrix} x \begin{bmatrix} 3 \\ 18 \\ 8 \end{bmatrix} \right) \text{mod } 26 = \begin{bmatrix} 1 \\ 0 \\ 17 \end{bmatrix} = \text{BAR}$$

$$WYG = P_5 = \left(\begin{bmatrix} 1 & -2 & 1 \\ -2 & 6 & -3 \\ 1 & -3 & 2 \end{bmatrix} x \begin{bmatrix} 22 \\ 24 \\ 6 \end{bmatrix} \right) \text{mod } 26 = \begin{bmatrix} 6 \\ 4 \\ 14 \end{bmatrix} = \text{GEO}$$

$$OJQ = P_6 = \left(\begin{bmatrix} 1 & -2 & 1 \\ -2 & 6 & -3 \\ 1 & -3 & 2 \end{bmatrix} x \begin{bmatrix} 14 \\ 9 \\ 16 \end{bmatrix} \right) \text{mod } 26 = \begin{bmatrix} 12 \\ 4 \\ 19 \end{bmatrix} = \text{MET}$$

$$HHY = P_7 = \left(\begin{bmatrix} 1 & -2 & 1 \\ -2 & 6 & -3 \\ 1 & -3 & 2 \end{bmatrix} x \begin{bmatrix} 7 \\ 7 \\ 24 \end{bmatrix} \right) \text{mod } 26 = \begin{bmatrix} 17 \\ 8 \\ 8 \end{bmatrix} = \text{RII}$$

Hasil akhir dari proses dekripsi adalah "ILOVEALJABARGEOMETRII". Sehingga apabila dikurangi karakter yang ditambahkan akan didapatkan "ILOVEALJABARGEOMETRI".

2.5 *Caesar Cipher*

Caesar Cipher merupakan salah satu algoritma *cipher* tertua dan yang paling diketahui dalam perkembangan ilmu kriptografi. *Cipher* ini digunakan oleh Julius Caesar untuk mengirimkan pesan kepada para gubenurnya sehingga algoritma *cipher* ini dikenal dengan *Caesar Cipher*. *Caesar Cipher* merupakan salah satu kriptografi klasik substitusi *monoalphabetic* dimana proses enkripsi dilakukan dengan menjumlahkan tiap-tiap karakter pesan dengan kunci sehingga karakter bergesar dan membentuk karakter baru yang disebut *ciphertext*. Karakter yang digunakan pada *Caesar Cipher* yaitu A, B, C, …, Z dan masing-masing karakter dikonversi menjadi 0, 1, 2, …, 25 seperti pada Tabel 2.2 . Kunci yang digunakan pada *Caesar Cipher* bernilai integer sehingga untuk proses enkripsinya setiap karakter bergeser ke kanan dalam jumlah pergeseran yang sama dengan kuncinya sedangkan untuk proses dekripsi *ciphertext* dieliminasikan dengan kunci sehingga setiap karakter bergeser ke kiri sebanyak kuncinya.

Misalkan untuk susunan abjad (alfabet) disubstitusikan dengan $k = 3$, maka tabel substitusinya

Tabel 2.4 Tabel Substitusi Alfabet

Plaintext																									
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
Ciphertext																									
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

Contoh : Akan dienkripsikan “KRIPTOGRAFI” dengan $k = 3$

Penyelesaian :

K = 10 + 3 = 13 = N
R = 17 + 3 = 20 = U
I = 8 + 3 = 11 = L
P = 15 + 3 = 18 = S
T = 19 + 3 = 22 = W
O = 14 + 3 = 17 = R
G = 6 + 3 = 9 = J
R = 17 + 3 = 20 = U
A = 0 + 3 = 3 = D
F = 5 + 3 = 8 = I
I = 8 + 3 = 11 = L

Sehingga hasil enkripsinya “NULSWRJUDIL”

Secara sistematis proses enkripsi *Caesar Cipher* adalah

$$C_i = E(P_i) = (P_i + k) \bmod 26 \quad (2.5)$$

Dan proses dekripsinya

$$P_i = D(C_i) = (C_i - k) \bmod 26 \quad (2.6)$$

Dimana *P* = *Plaintext*

C = *Ciphertext*

K = Kunci

Bentuk modifikasi pada kunci dengan memanfaatkan metode *Caesar Cipher* dimana kunci digeser sebanyak letak posisi karakternya (Prabowo, 2015). Misalkan pesan “KRIPTOGRAFI” dikonversi ke angka dan disubstitusikan dengan posisinya maka akan menjadi “LTLTYUNZJPT”.

K	R	I	P	T	O	G	R	A	F	I	
10	17	8	15	19	14	6	17	0	5	8	
1	2	3	4	5	6	7	8	9	10	11	+
11	19	11	19	24	20	13	25	9	15	19	
L	T	L	T	Y	U	N	Z	J	P	T	

Modifikasi ini membuat hasil substitusinya tidak menghasilkan karakter pesan yang sama seperti pada proses enkripsi *Caesar Cipher* yang ada. Seperti pada contoh pesan KRIPTOGRAFI, karakter R dan I yang muncul dua kali menghasilkan *ciphertext* yang sama yaitu U dan L. Namun pada modifikasi *Caesar Cipher*, karakter R menghasilkan *ciphertext* yang berbeda yaitu T dan Z sedangkan karakter I menghasilkan *ciphertext* L dan T.

2.6 Metode Babbage-Kasiski

Metode Babage-Kasiski memanfaatkan bahwa bahasa inggris tidak hanya mengandung perulangan huruf, tetapi juga perulangan pasangan huruf atau triple huruf seperti TH,THE,NG,ING, dll. Perulangan tersebut juga merupakan kelemahan algoritma *vigenere cipher* yaitu memiliki kunci yang berulang sehingga pada *plaintext* panjang terdapat kemungkinan bahwa *plaintext* yang sama akan dienkrispi dengan karakter kunci yang sama sehingga akan ada

perulangan pada *ciphertext*. Metode ini ditemukan oleh Friedrich Kasiski yang berhasil memecahkan *Vigenere* Cipher pada 1863 namun sebelumnya Charles Babbage telah mengembangkan metode yang serupa pada 1854. Metode Babbage-Kasiski ini dapat digunakan untuk menemukan panjang kunci pada *vigenere cipher*. Langkah-langkah untuk mencari panjang kunci adalah sebagai berikut :

a. Menghitung semua kriptogram yang berulang.
b. Menghitung jarak antar kriptogram yang berulang.
c. Menghitung faktor pembagi dari jarak antar kriptogram yang berulang. Faktor pembagi tersebut dapat menjadi kemungkinan dari panjang kunci.
d. Menentukan irisan dari himpunan faktor persekutuan terbesar yang mana irisan tersebut didapatkan dari angka yang muncul pada semua faktor pembagi. Irisan faktor persekutuan terbesar tersebut adalah panjang kunci yang mungkin.

Misalkan saja terdapat *ciphertext* sebagai berikut :

LJVBQ STNEZ LQMED **LJV**MA MPKAU FAVAT **LJV**DA YYVNF JQLNP **LJV**HK VTRNF **LJV**CM LKETA **LJV**HU YJVSF KRFTT WEFUX VHZNP (Munir, 2010).

Pada contoh *ciphertext* tersebut, LJV berulang sebanyak 6 kali sehingga apabila digunakan untuk mencari irisan faktor persekutuan terbesar, maka hitung jarak pada masing-masing LJV.

Jarak LJV 1 – LJV 2 = 15
Jarak LJV 2 – LJV 3 = 15
Jarak LJV 3 – LJV 4 = 15
Jarak LJV 4 – LJV 5 = 10
Jarak LJV 5 – LJV 6 = 10

Faktor pembagi 15 adalah 1, 3, 5, 15
Faktor pembagi 10 adalah 1, 2, 5, 10

Sehingga irisan dari faktor pembagi LJV adalah 5, maka kemungkinan panjang kuncinya adalah 5. Selanjutnya kunci dapat dicari dengan menggunakan analisis frekuensi maupun metode lainnya.

2.7 Analisis Frekuensi *Vigenere Cipher*

Selain metode Babbage-Kasiski, perulangan kunci yang terdapat pada algoritma *Vigenere Cipher* juga dimanfaatkan kriptanalis pada teknik pemecahan analisis frekuensi. Pada pemecahan *ciphertext*, kriptanalis tidak sepenuhnya sempurna karena diambil nilai kemungkinan atau *probability* yang disebut *index of coincidence. Index of coincidence* ini menghubungkan suatu huruf dengan huruf yang lain yang dapat menghasilkan kunci. Rumus *index of coincidence* yaitu:

$$I_c(x) = \frac{\sum_{i=0}^{25} f_i(f_i-1)}{n(n-1)} \quad (2.7)$$

I_c = *index of coincidence*

f_i = frekuensi A, B, C, …, Z

n = banyaknya karakter pada *ciphertext*

Metode analisis frekuensi merupakan salah satu cara mencari panjang kunci dari *ciphertext Vigenere* yang masuk dalam *ciphertext only attack.* Metode ini membagi *ciphertext* ke dalam beberapa bagian yang kemudian dihitung jumlah karakternya menggunakan *index of coincidence* yang kemudian menjadi kemungkinan dari panjang kunci *ciphertext*.

Teknik analisis frekuensi dapat dilakukan sebagai berikut

a. Hitung karakter dari masing-masing baris dan kolom
b. Catat frekuensi tiap urutan
c. Hitung *index of coincidence* pada kolom dan baris
d. Hitung rata-rata dari setiap *index of coincidence* yang dihasilkan
e. Hasil *index of coincidence* terbesar merupakan kemungkinan dari panjang kunci

Setelah mengetahui kemungkinan panjang kunci, maka kemungkinan panjang kunci dapat digunakan sebagai acuan untuk menemukan kunci yang sebenarnya. Misalkan j adalah panjang kunci, maka untuk i=1, 2, …, j maka m_i merupakan penjabaran frekuensi A, B, C, …, Z dalam metode analisis frekuensi perbarisnya dimana p_0 adalah frekuensi kemunculan huruf dalam bahasa inggris atau bahasa Indonesia dan f_i adalah frekuensi huruf dalam *index of coincidence*.

$$\left.\begin{aligned} y_i(m_0) &= \frac{(p_0 f_0) + (p_1 f_1) + (p_2 f_2) + \cdots + (p_{25} f_{25})}{n/m} \\ y_i(m_1) &= \frac{(p_0 f_1) + (p_1 f_2) + (p_2 f_3) + \cdots + (p_{25} f_0)}{n/m} \\ y_i(m_2) &= \frac{(p_0 f_2) + (p_1 f_3) + (p_2 f_4) + \cdots + (p_{25} f_1)}{n/m} \\ &\vdots \\ y_i(m_{25}) &= \frac{(p_0 f_{25}) + (p_1 f_0) + (p_2 f_1) + \cdots + (p_{25} f_{24})}{n/m} \end{aligned}\right\} \quad (2.8)$$

Tabel 2.5 Frekuensi Kemunculan Huruf dalam Bahasa Inggris (Bawono, 2015) dan Bahasa Indonesia (Shah dkk, 2013)

Huruf	Frekuensi Bahasa Inggris(%)	Frekuensi Bahasa Indonesia(%)	Huruf	Frekuensi Bahasa Inggris(%)	Frekuensi Bahasa Indonesia(%)
A	0,082	0,2039	N	0,067	0,0933
B	0,015	0,0264	O	0,075	0,0126
C	0,028	0,0076	P	0,019	0,0261
D	0,043	0,05	Q	0,001	0,0001
E	0,127	0,0828	R	0,06	0,0464
F	0,022	0,0021	S	0,063	0,0415
G	0,02	0,0366	T	0,091	0,0558
H	0,061	0,0274	U	0,028	0,0462
I	0,07	0,0798	V	0,01	0,0018
J	0,002	0,0087	W	0,023	0,0048
K	0,008	0,0514	X	0,001	0,0003
L	0,04	0,0326	Y	0,02	0,0188
M	0,024	0,0421	Z	0,001	0,0004

$$M_g \approx \sum_{i=0}^{25} p_i^2 = p_0^2 + p_1^2 + p_2^2 + \cdots + p_{25}^2$$

$$= 0{,}082^2 + 0{,}015^2 + 0{,}028^2 + \ldots + 00{,}001^2$$

$$= 0{,}0656$$

Pada perhitungan *IC* akan ditemukan nilai *IC* tertinggi yang menjadi panjang kunci sehingga pada perhitungan y_i nilai frekuensi analisis frekuensi yang paling mendekati 0,0656 yang kemungkinan menjadi kunci (Bawono, 2015). Metode analisis frekuensi tidak selalu akurat karena metode ini menggunakan *probability* yang melihat hasil kemungkinan yang tinggi. Sehingga untuk lebih akuratnya lagi dapat dibandingkan dengan metode pemecahan kunci lainnya.

2.8 Kriptanalisis Terhadap Hill Cipher

Algoritma *Hill Cipher* diciptakan agar tidak dapat dipecahkan dengan analisis frekuensi yang merupakan teknik pemecahan kunci pada *Vigenere Cipher*. Kriptanalisis pada *hill cipher* sendiri akan sulit apabila hanya diketahui *ciphertext* saja (*ciphertext only attack*) karena *ciphertext* pada *hill cipher* tidak berpola dan setiap karakter dalam blok saling mempengaruhi karakter yang lain (Widyanarko, 2007). Apabila kunci matriks yang digunakan besar akan menambah tingkat kesulitan karena menyebabkan semakin banyaknya karakter yang mempengaruhi karakter yang dihasilkan.

Teknik kriptanalisis yang dapat digunakan untuk memecahkan kunci *hill cipher* yaitu *known-plaintext attack*. Teknik ini memerlukan potongan *plaintext* dan *ciphertext* yang saling berkorespondensi. Namun proses yang sulit yaitu untuk mencari panjang kunci pada *hill cipher*. Apabila kriptanalis telah mengetahui panjang kuncinya, akan sangat mudah kriptanalis menemukan kuncinya serta mendekripsikan pesan menjadi pesan asli. Dalam mencari panjang kunci, kriptanalis hanya dapat memperkirakan dan menebaknya sehingga apabila kunci sangat besar akan menambah tingkat kesulitan dan akan memakan waktu yang sangat lama untuk memecahkannya. Untuk mencari kunci *hill cipher* apabila telah diketahui potongan *plaintext* dan *ciphertext* yang saling berkorespodensi serta panjang kuncinya maka digunakan persamaan (2.9).

$$C = K.\ P$$
$$C.\ P^{-1} = K.\ P.\ P^{-1}$$
$$C.\ P^{-1} = K \qquad (2.9)$$

2.9 Invers Matriks Modulo

Pada algoritma *Hill Cipher*, kunci matriks berukuran *n* x *n* merupakan kunci yang akan digunakan pada proses enkripsi. Kunci matriks dikalikan dengan *plaintext* menghasilkan *ciphertext.* Sedangkan pada proses dekripsi digunakan invers modulo dari matriks kunci yang akan dikalikan dengan *ciphertext*.

Definisi 2.1 (Modulo) Misalkan m adalah bilangan bulat. Untuk $a, b \in \mathbb{Z}$ dapat ditulis $a \equiv b \ (mod\ m)$ dan dibaca "a kongruen dengan b modulo m" jika berlaku $m|a - b$ atau m habis membagi $a - b$.

Teorema 2.1 (Invers Modulo) Jika a adalah suatu bilangan Z_m maka bilangana^{-1}dalam Z_m disebut balikan atau invers perkalian dari a modulo m jika $a.a^{-1} = a^{-1}.a = 1 \ (mod\ m)$.

Dapat dibuktikan bahwa jika a dan m relatif prima atau tidak mempunyai faktor prima bersama, maka a mempunya invers modulo m.

Misalkan *A* adalah matriks kunci, maka langkah-langkah untuk mencari invers modulo dari kunci matriks *A* yaitu :

a. Mencari nilai determinan matriks *A*.
b. Menghitung adjoin dari maktriks *A*.
c. Mencari nilai invers dari determinan matriks *A*. Apabila menggunakan modulo 95, maka 95 dan determinan matriks *A* harus relatife prima sehingga dapat ditemukan nilai invers dari determinan matriks *A*.
d. Mengalikan invers determinan matriks *A* dengan adjoin dari matriks *A*.
e. Menghitung modulo 95 pada langkah d. Hasil modulo dari langkah d merupakan invers modulo dari matriks *A* (Tomasouw, 2014).

BAB 3. METODE PENELITIAN

3.1 Data Penelitian

Data yang akan digunakan pada penelitian ini berupa pesan Teks. Pesan teks tersebut dapat berupa alfabet, angka, maupun simbol yang merupakan *ASCII Printable Character*. Pesan teks akan diubah ke dalam kode ASCII sebelum dienkripsi.

3.2 Langkah-Langkah Penelitian

Langkah-langkah yang dilakukan dalam penelitian ini adalah sebagai berikut :

a. Studi Literatur

   Tahap ini dilakukan untuk memahami teori-teori yang dibutuhkan pada penelitian. Teori yang digunakan yaitu algoritma *Vigenere Cipher*, algoritma *Caesar Cipher* sebagai algoritma yang digunakan pada modifikasi *Vigenere Cipher*, dan algoritma *Hill Cipher*.

b. Perancangan Program

   Pada langkah ini dilakukan perancangan desain GUI (*Guide User Interface)* menggunakan *software* Matlab 2009a. Proses perancangan meliputi *form layout*, tata letak *edit text*, dan *push button* serta pewarnaan tampilan agar lebih menarik.

c. Pembuatan Program

   Pembuatan program didasarkan pada algoritma yang digunakan pada proses enkripsi dan dekripsi. Proses enkripsi dan dekripsi menggunakan modifikasi *Vigenere Cipher* dan *Hill Cipher*. Kunci yang akan digunakan pada proses enkripsi dan dekripsi disebut *final key* yang didapatkan dari proses enkripsi menggunakan *Caesar Cipher*, *Vigenere Cipher*, dan *Hill Cipher*. Langkah-langkah dalam melakukan proses enkripsi maunpun dekripsi adalah sebagai berikut:

1) Enkripsi

a) Pemeriksaan *Plaintext*

Pada tahap ini, *plaintext* diperiksa agar pada saat proses enkripsi *hill* tidak terjadi penambahan karakter. Apabila karakter kurang, *plaintext* ditambahkan dengan karakter spasi. Sehingga terdapat *plaintext* baru yang sudah memuat tambahan spasi.

b) Pembentukan Kunci

Pada pembentukan kunci dilakukan sebanyak tiga tahapan, yaitu

i. Pembentukan Kunci 1

Pada pembentukan kunci 1 digunakan metode *Caesar Cipher* yang mana *plaintext* yang digunakan pada metode ini adalah kunci (*K*) dan posisi karakternya adalah *key (i)*. Kunci berulang hingga panjangnya setengah dari panjang pesan. Dengan memanfaatkan persamaan (2.5) dan jumlah karakter yang digunakan adalah 95 karakter maka secara sistematis pembentukan kunci pertama dituliskan seperti pada persamaan (3.1).

$$K1_i = E(K1_i) = (((K_i - 32) + i) \bmod 95) + 32 \quad (3.1)$$

ii. Pembentukan Kunci 2

Pada pembentukan kunci 2 digunakan metode *Vigenere Cipher* dengan *plaintext* merupakan hasil dari tahap pertama (*K1*) dan kunci (*K*). Dengan menggunakan persamaan (2.1), maka secara sistematis pembentukan kunci kedua dituliskan seperti pada persamaan (3.2).

$$K2_i = E(K2_i) = (((K1_i - 32) + (K_i - 32))mod\ 95) + 32 \quad (3.2)$$

iii. Pembentukan *Final Key*

Pada pembentukan *final key* digunakan metode *Hill Cipher* untuk membentuk *final key* (*FK*) yang akan digunakan untuk mengenkripsi pesan. Kunci pada tahap ini merupakan perulangan kunci (*K*) yang mana akan diambil 9 karakter dari belakang sebagai kunci matriks (*KM*) berukuran 3x3. Dengan menggunakan persamaan (2.3), maka secara sistematis pembentukan *final key* dituliskan pada persamaan (3.3).

$$FK_i = (((KM - 32)x(K2 - 32))mod\ 95\ ) + 32 \quad (3.3)$$

c) Pemeriksaan *Final Key*

Setelah pembentukan *FK*, dilakukan pemeriksaan *FK* untuk melihat apakah *FK* memiliki matriks invers. Apabila *FK* tidak memiliki invers, maka kunci awal *K* harus diganti hingga ditemukan *FK* yang memilik invers. Hal ini dilakukan supaya *ciphertext* dapat didekripsi.

d) Pembagian *Plaintext*

Sebelum dilakukan proses enkripsi, *plaintext* dibagi terlebih dahulu menjadi *plaintext* ganjil dan genap. *Plaintext* ganjil merupakan karakter yang terletak pada posisi ganjil sedangkan *plaintext* genap merupakan karakter yang terletak pada posisi genap. Sebagai contohnya dapat dilihat pada Tabel 3.1.

Tabel 3.1 Pembagian *Plaintext*

K	R	I	P	T	O	G	R	A	F	I
1	2	3	4	5	6	7	8	9	10	11

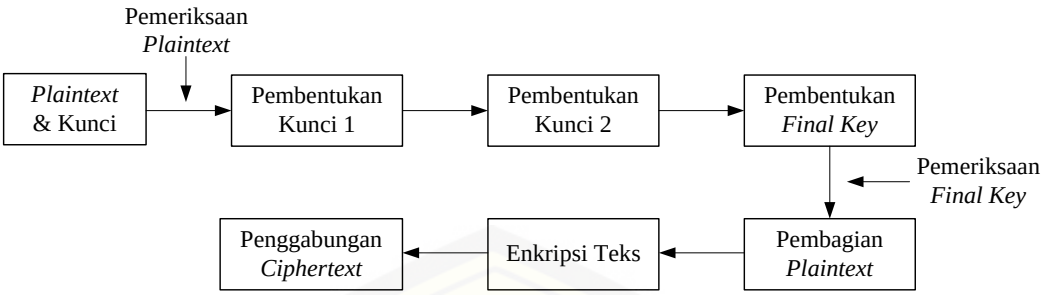
Plaintext ganjil pada Tabel 3.1 adalah KITGAI sedangkan *plaintext* genapnya adalah RPORF.

e) Proses Enkripsi

Proses enkripsi data dilakukan terhadap *plaintext* ganjil dan *plaintext* genap dengan metode yang berbeda. Pada *plaintext* ganjil digunakan persamaan (3.4) yaitu perluasan *Vigenere Cipher* dengan kunci yang digunakan menggunakan *final key*. Sedangkan *plaintext* genap digunakan persamaan (3.5) yaitu perluasan *Hill Cipher* yang juga menggunakan *final key*. Hasil akhir dari kedua proses enkripsi disatukan kembali dan menjadi *ciphertext*. Langkah-langkah pada proses enkripsi dapat dilihat pada Gambar 3.1.

$$C_i = EK(P_i) = (P_i + K_i) mod\ 95 \quad (3.4)$$

$$C = (K.P) mod\ 95 \quad (3.5)$$



Gambar 3.1 Proses Enkripsi

2) Dekripsi

a) Pembentukan Kunci

Pada tahap ini, dilakukan 3 tahapan untuk membentuk *final key*. Langkah-langkah untuk membentuk *final key*, sama seperti pembentukan kunci pada proses enkripsi. Sehingga *final key* yang digunakan pada proses enkripsi dan dekripsi adalah sama.

b) Pembagian *Ciphertext*

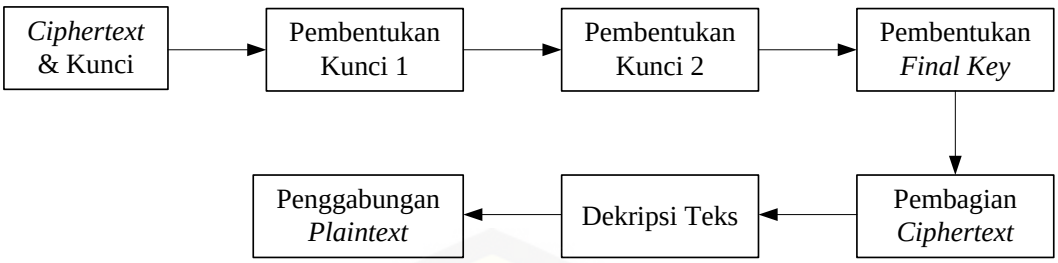
Sebelum dilakukan proses dekripsi, *ciphertext* dibagi menjadi *ciphertext* ganjil dan *ciphertext* genap. Sama seperti pada proses enkripsi, *ciphertext* ganjil terdiri dari *ciphertext* yang letak posisi karakternya ganjil sedangkan *ciphertext* genap yang letak posisi karakternya genap.

c) Proses Dekripsi

Ciphertext dibagi menjadi *ciphertext* ganjil dan genap. Kemudian dilakukan dekripsi menggunakan persamaan (3.6) pada *ciphertext* ganjil dan persamaan (3.7) pada *ciphertext* genap. Hasil akhir dari kedua dekripsi disatukan kembali menjadi *plaintext*. Langkah-langkah pada proses dekripsi dapat dilihat pada Gambar 3.2

$$P_i = DK(C_i) = (C_i - K_i) mod\ 95 \quad (3.6)$$

$$P = (K^{-1}.C) mod\ 95 \quad (3.7)$$



Gambar 3.2 Proses Dekripsi

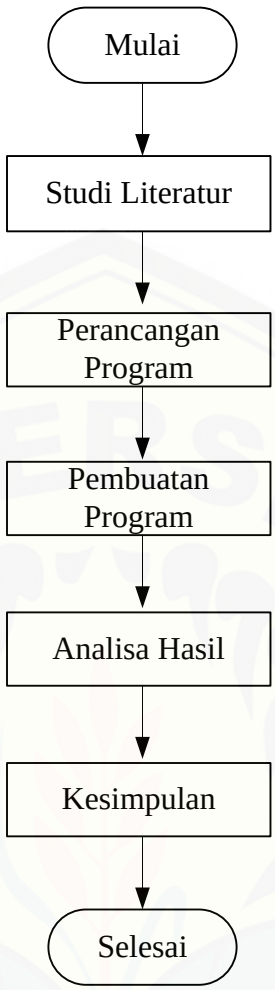
d. Analisis Hasil

Menguji dan menganalisa program yang dibuat telah berjalan sesuai metode yang digunakan serta membandingkan algoritma sebelum dan sesudah modifikasi. Selain itu dilakukan uji keamanan dengan menggunakan metode kasiski serta persamaan (2.7) dan (2.8) analisis frekuensi.

e. Kesimpulan

Mengambil kesimpulan dari penelitian yang dilakukan yaitu menganalisa hasil proses sebelum dan sesudah modifikasi serta perbedaan dari modifikasi kunci pada *ciphertext*.

Flowchart dari langkah-langkah penelitian yang dilakukan dapat diamati pada Gambar 3.3



Gambar 3.3 *Flowchart* Penelitian

BAB 5. PENUTUP

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan, dapat disimpulkan bahwa proses enkripsi menggunakan *final key* dapat menghilangkan kelemahan pada *Vigenere Cipher*. Selain itu penggabungan dengan *Hill Cipher* menambah tingkat keamanan dalam pemecahan algoritma tersebut. Proses dekripsi juga mampu mengembalikan *ciphertext* menjadi *plaintext* tanpa adanya perubahan. Pembentukan kunci baru dengan menggunakan 3 algoritma yang berbeda menghasilkan karakter kunci yang tidak memiliki pola perulangan. Perulangan kunci serta karakter yang berulang merupakan faktor utama kunci dapat dipecahkan. Pembentukan kunci baru serta penggabungan dengan *hill cipher* menyebabkan *ciphertext* tidak bisa dipecahkan dengan metode kasiski maupun analisis frekuensi.

5.2 Saran

Saran yang dapat diberikan untuk penelitian selanjutnya yaitu dapat menerapkan penggabungan algoritma pada gambar atau media lainnya. Misalkan pada citra RGB, maka proses pengenkripsian pada masing-masing kanal dilakukan dengan algoritma yang berbeda.

DAFTAR PUSTAKA

Bawono, H. R. C. 2015. *Kriptanalisis Pada Algoritma Vigenere*. *Skripsi*. Yogyakarta: Universitas Sanata Dharma.

Garg, S., S. Khera, dan A. Aggarwal. 2016. Extended Vigenere Cipher with Stream Cipher. *International Journal Engineering Science and Computing (IJESC)* 6(5): 5176-5180.

Kester, Q. 2012. A Cryptosystem Based on Vigenere Cipher with Varying Key. *Internation Journal of Advanced Research in Computer Engineering & Technology (IJARCET)* 1(10):108-113.

Menezes, A., P. van Oorschot, dan S. Vanstone. 1996. *A Handbook of Applied Cryptography*. Florida. CRC Press, Inc.

Munir, R. 2006. *Diktat Kuliah IF5054 Kriptografi.* Jakarta: Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika.

Munir, R. 2010. *Diktat Kuliah IF3058 Kriptografi.* Bandung: Departemen Teknik Informatika, Institut Teknologi Bandung.

Prabowo, H. 2015. *Enkripsi Teks Menggunakan Metode Vigenere Cipher dengan Pembentukan Kunci Tiga Tahap.* Skripsi. Semarang: Teknik Elektro, Universitas Negeri Semarang.

Schneier, B. 1996 . *Applied Cryptography 2nd.* New York. John Wiley & Son, Inc.

Shah, A, A. Z. Saidin, I. F. Taha, A. M. Zeki, dan Z. Bhatti. 2013. Similarities and Dissimilarities between Character Frequencies of Written Text of Melayu, English, and Indonesian Languages. *2nd International Conference on Advanced Computer Scienci Application and Technologie*. Kuching, Malaysia. 22-24 Desember: 192-194.

Tomasouw, B. P. 2014. Invers Matriks Kunci Pada Algoritma Chiper Hill. https://www.slideshare.net/bernypebo/invers-matriks-kunci-pada-algoritma-chiper-hill. [Diakses pada 10 Mei 2017].

Utama, M. 2015. *Penerapan Matriks dalam Kriptografi Hill Cipher*. Bandung: ITB.

Widyanarko, A. 2007. *Study dan Analisis Mengenai Hill Cipher, Teknik Kriptanalisis dan Upaya Penanggulangannya*. Bandung: TI ITB.

LAMPIRAN

LAMPIRAL A. (Perhitungan Kunci Menggunakan Metode Kasiski dan Analisis Frekuensi)

Ciphertext 2 (*Vigenere Cipher*)

IE**FFE**DUBDCQMFKTCVREROTQYNFZCWYMJEJBCABEQOWFJYRZB
LFZCLP**FFE**WMATKTYTEAKAKFCRYAUMLOFTYQWSYASTZFUICX
LOKDCATTKABQQTYQKSFSPAMQJYCALECKRHRFRHVKOUZFJYJT
STKTCIIYMUKTQNFIUECXGDFZMTTMPENTCNFZJYKTCTYULGTMJ
LVPJOEQJIEQQSIQKAZZCDSKKYJUBEZTYVVNCCFYCCFYNLVFCLP
MJOEQQODQQAPNYSKMPDPASHRHCBVOMMVMAECQZRZFWOYR
SCBFFAKKCAYICLCEMWYMRNFNMDPDCMRULSSKKYJUBEZMKO
BXGKVFFIJ**FFE**JQUOIPQTYMRAIQCAJUJYJMGDKAUAIPQMVCSITW
JYSQAODQQANMJLVHCNCALECULEJERUIZQIEFMSFYCTYULGPAS
CRZQEVQTEEUDMPNPERFFSRDCBCAAKVPZETMSSVUYMTALFZZC
DZZRHRFUACXGHVMBTFIYRUERHVESRWMAEFRUAKQPASATECA
LECKJOEQJYCALECKUHRXCSZZEIESYLFZCLZWCTYUQEMQLMVIF
OZEJIBQYLFZCLPUQLRZBCRZGSYULEFZRHVASTJUBECALECKJOE
QJYCALECKUHRXCLZWCTYUQTIKAACXGNXALCVMEAZZSNKUJT
YUQSFZETYMRDFQQNFFFAMQYRVENOEECRVMAHVERODAPRFIY
NVZBLVEQSZSLACIGLCDCATTQODQBAPQTEIKUHVDCEMQLTF**FF**
EF**FFE**IEGDVADTYQCAI**FFE**MQLTYQZLZZBWYMJEJIGLCNCASXCT
FECEDQROUMWAXMGNZEGNX

Berikut merupakan langkah-langkah untuk mencari kunci pada *ciphertext* 2.

a. Menghitung panjang kunci dengan Metode Kasiski

   Karakter berulang pada *Ciphertext* yang paling banyak yaitu FFE sehingga jarak antar FFE yaitu

   Jarak FFE 1 – FFE 2 = 52 Jarak FFE 3 – FFE 4 = 424

   Jarak FFE 2 – FFE 3 = 288 Jarak FFE 4 – FFE 5 = 4

Jarak FFE 5 – FFE 6 = 16

Hitung GCD pada jarak antar FFE, hasil dari GCD tersebut merupakan kemungkinan dari panjang kunci.

GCD(52,288,424,4,16) = 4

Sehingga kemungkinan panjang kuncinya yaitu 4 karakter.

b. Mencari Kunci Sebenarnya dengan Analisis Frekuensi

Karena panjang kemungkinan kunci adalah 4, maka bagi *ciphertext* menjadi 4 bagian menjadi m1, m2, m3, dan m4. Hitung frekuensi karakter pada masing-masing baris.

m1 =

IEDFVONWEAOYLLETTARMTSTIOAATSAYEHHUYTIUNEDTENYTG
LOISADYEVCCLLOOASDHBMEROCAALWNDMSYEOKIEOTAAYDA
MIYOALNEEUISTGCEEMESBKESMFDHAHTRHREAAEEOYEHSILLTE
MOILLLCSEHTEEOYEHLTTANCANTSTDNARORHORNLSALAOAEHE
TEEDTAETLWELATEOANN

Huruf	Frekuensi	Huruf	Frekuensi
A	25	N	11
B	2	O	17
C	6	P	0
D	9	Q	0
E	32	R	7
F	2	S	12
G	2	T	21
H	11	U	3
I	9	V	2
J	0	W	3
K	2	X	0
L	17	Y	10
M	7	Z	0

m2 =

EDCKRTFYJBWRFPWKEKYLYYZCKTBYFMCCRVZJKIKFCFTNFKYT
VEEIZSJZVFFVPEDPKPRVVCZYBKYCYFPRSJZBVJJIYIJJKIVTSDNVC
CJIEFYPRVEPRRCVTVTZZRCVFUVWFKSCCECCRZEFZYMVZBFPRR

YFVJCCECCRZYICXVZKYFYFFMVEVVDFVVZCCTDPIVMFFIVYIMY
ZYJCSFDUXZX

Huruf	Frekuensi	Huruf	Frekuensi
A	0	N	2
B	5	O	0
C	23	P	9
D	6	Q	0
E	11	R	13
F	22	S	5
G	0	T	8
H	0	U	2
I	10	V	24
J	11	W	3
K	12	X	3
L	1	Y	19
M	5	Z	16

m3 =
FUQTEQZMBEFZZFMTAFAOQAFXDTQQSQAKFKFTTYTIXZMTZTUM
PQQQZKUTNYYFMQQNMAHOMQFRFKIEMNDUKUMXFFQPMQUMA
PCWQQMHAUEZFYUAZQUNFDAPMUAZZFXMIEEMRQAAKQAKXZS
ZWUQIEQZUZZUZAUAKQAKXWUKXAMZUUZMQFQEEMEAIZESIDT
QQKDQFFEAQFQQZMINXEQMME

Huruf	Frekuensi	Huruf	Frekuensi
A	19	N	5
B	1	O	2
C	1	P	4
D	5	Q	31
E	14	R	2
F	19	S	3
G	0	T	10
H	2	U	17
I	7	V	0
J	0	W	3
K	11	X	8
L	0	Y	4
M	21	Z	20

m4 =

FBMCRYCJCQJBCFAYKCUFWSULCKQKPJLRROJSCMQUGMPCJCLJJ
JQKCKBYCCNCJQQYPSCMAZWSFCCMRMCLKBKGFFUQRCJGUQSJ
AQJCLLRQMCLSQTDPFCAZSYLCRUGBYRSAUPTLJJLUCEYCCQLFJ
YCQBGLRSBLJJLUCCQAGLESJQERQFYNCARPYBQLGCQBTUCLFFG
DCFLZBJGCCCRWGG

Huruf	Frekuensi	Huruf	Frekuensi
A	7	N	2
B	10	O	1
C	35	P	6
D	2	Q	18
E	3	R	12
F	12	S	10
G	11	T	3
H	0	U	10
I	0	V	0
J	19	W	3
K	7	X	0
L	18	Y	10
M	7	Z	3

Hitung pada masing-masing baris menggunakan persamaan (2.8) . Karakter pada masing-masing baris yang memiliki nilai IC mendekati 0,0656 merupakan kemungkinan karakter kunci pada baris tersebut.

$$m_1 = A = \frac{(0{,}082.25) + (0{,}015.2) + (0{,}028.6) + \cdots + (0{,}001.0)}{210} = 0{,}0687$$

$$m_1 = B = \frac{(0{,}082.2) + (0{,}015.6) + (0{,}028.9) + \cdots + (0{,}001.25)}{210} = 0{,}0344$$

$$\vdots$$

$$m_1 = Z = \frac{(0{,}082.0) + (0{,}015.25) + (0{,}028.2) + \cdots + (0{,}001.10)}{210} = 0{,}0371$$

Setelah perhitungan pada m1, m2, m3, dan m4 maka akan didapatkan karakter kunci A pada m1, R pada m2, M pada m3, dan Y pada m4 yang memiliki nilai IC mendekati 0,0656. Karakter tersebut merupakan kunci pada *ciphertext* 2. Sehingga didapatkan kunci yang sebenarnya yaitu ARMY.

Tabel *Index Coincidence*

	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		
m1	**0,0687**	0,0344	0,0274	0,0313	0,0434	0,032	0,0361
	0,0448	0,0342	0,0326	0,0380	0,0493	0,003	0,0399
	0,0345	0,0415	0,0379	0,0324	0,0374	0,0421	0,0366
	0,0313	0,0475	0,0369	0,0324	0,0371		
m2	0,0320	0,0411	0,0465	0,0310	0,0392	0,0407	0,0380
	0,0356	0,0349	0,0353	0,0409	0,0427	0,0344	0,0408
	0,0395	0,0341	0,0360	**0,0623**	0,0347	0,0285	0,0387
	0,0473	0,0306	0,0369	0,0476	0,0340		
m3	0,0367	0,0417	0,0386	0,0327	0,0349	0,0398	0,0395
	0,0341	0,0490	0,0363	0,0276	0,0353	**0,0689**	0,0369
	0,0291	0,0289	0,0453	0,0360	0,0379	0,0391	0,0358
	0,0344	0,0397	0,0454	0,0378	0,0412		
m4	0,0294	0,0373	0,0492	0,0329	0,0314	0,0399	0,0347
	0,0358	0,0352	0,0490	0,0364	0,0402	0,0365	0,0414
	0,0410	0,0343	0,0397	0,0382	0,0398	0,0280	0,0431
	0,0332	0,0327	0,0415	**0,0642**	0,0374		

LAMPIRAN B. (Skrip Pembentukan Kunci)

finalkey.m

```
function ek = finalkey(plain,key)
fk2='';
a=0;
n=length(plain);
m=length(key);
x=0;
p=1;
q=1;

%CEK BUAT HILL
zn=length(plain);
for zi=1:2:zn
    zgjl(zi)=plain(zi);
end

for zig=1:zn
    if (2*zig-1)>zn
        break
    else
        zgjl2(zig)=zgjl(2*zig-1);
    end
end

for zj=2:2:zn
    zgnp(zj)=plain(zj);
end

for zip=1:zn
    if (2*zip)>zn;
        break
    else
        zgnp2(zip)=zgnp(2*zip);
    end
end

ch32=32;
c32=char(ch32);
pzl=length(zgjl2);
pzp=length(zgnp2);
mpzp=mod(pzp,3);
plaintk2='    ';
[zx1,zx2]=size(plain);
zpk=0;
if mpzp==0
    plaintk2=plain;
elseif mpzp==1
        for zkk=zx2+1:zx2+4
```

```
            for z1k=1:4
                zpk=zpk+1;
                plain(zkk)=plaintk2(z1k);
            end
        end
elseif mpzp==2
    for zkk=zx2+1:zx2+2
            for z1k=1:2
                zpk=zpk+1;
                plain(zkk)=plaintk2(z1k);
            end
        end
end

if mpzp==0
    setkey=n;
elseif mpzp==1
    setkey=n+4;
elseif mpzp==2
    setkey=n+2;
end
pkk1,pkk2]=size(plain);
pp0=round(pkk2/m);
pp1=repmat(key,1,pp0);
pp3=round(pkk2/2);
pp2=pp1(1:pp3);
key=pp2;

%CAESAR CIPHER
pjgknc=length(key);
for lok=1:pjgknc
    asciiknc(lok)=double(key(lok))-32;
    fk1(lok)=(mod((asciiknc(lok)+lok),95)+32);
end
fk1=char(fk1)

%VIGENERE CIPHER
for i=1:length(fk1)
    p=double(fk1(i));
    p=p-32;
    k=double(key(i))-32;
    pcode=mod(p+k,95);
    pcode=pcode+32;
    plaincode(i)=pcode;
    fk2=sprintf('%s%s',fk2,char(pcode));
end
fk2
%HILL CIPHER
n1=length(key);
blkknc='';
for blk1=n1:-1:1
```

```
    bknc=key(blk1);
    blkknc=sprintf('%s%s',blkknc,char(bknc));
end
blkknc;
key=blkknc;

%KUNCI
fkasli=fk2;
[a,b]=size(key);
pjg1=0;
l=0;
if b<9
    for pjg=b+1:9
        pjg1=pjg1+1;
        key(pjg)=key(pjg1);
    end
end

for it=1:3 %kolom
    for ite=1:3 %baris
        l=l+1;
        mat(it,ite)=key(l);
    end
end
matt=double(mat)-32;
pjghill=length(fk2);
[a,b]=size(fk2);
ass=mod(pjghill,3);
ass1=3-ass;
pjgg=0;
if ass~=0
    ab=ass1;
    for i=b+1:(pjghill+ab)
        pjgg=pjgg+1;
        fk2(i)=fk2(pjgg);
    end
end
fk2;
aa=length(fk2)/3;

%PEMBAGIAN 3-3
l1=0;
for i=1:aa
    for j=1:3
        l1=l1+1;
        m(i,j)=fk2(l1);
    end
end
mm=m';
ek='';
for i2=1:aa
```

```
    k1=m(i2,1:3);
    k2=k1';
    k3=double(k2)-32;
    matt;
    mat3=round(matt*k3);
    enk=(mod(mat3,95));
    enk=enk+32;
    enk2=enk';
    ek=sprintf('%s%s',ek,char(enk2));
end
```

LAMPIRAN C. (Skrip Invers Matriks Modulo)

modinvers.m

```
function dekh=modinvers(deka,z)
dekb=det(deka);
dekc=round(det(deka)*inv(deka));
[G, C] = gcd(dekb,z);
if G==1
    dekf = mod(C,z);
    mod(dekb*dekf,z);
    msgbox('Success');
else
    msgbox('Invalid Value. Try Again', 'Error','error');
end
dekf;
dekg=dekf*dekc;
dekh=mod(dekg,z);
```

LAMPIRAN D. (Skrip Ekripsi Penggabungan *Vigenere Cipher* dan *Hill Cipher*)

envigell.m

```
function enkripsihasil=envigell(plain,ek)
plainteks=plain;
pjgteks=length(plainteks);

for i=1:2:pjgteks
    gjl(i)=plainteks(i);
end
for ig=1:pjgteks
    if (2*ig-1)>pjgteks
        break
    else
        gjl2(ig)=gjl(2*ig-1);
    end
end
%VIGENERE
vteks=gjl2;
vkey=ek;
vchip='';
for vi=1:length(vteks)
    vp=double(vteks(vi));
    vp=vp-32;
    vk=double(vkey(vi))-32;
    vpcode=mod(vp+vk,95);
    vpcode=vpcode+32;
    vplaincode(vi)=vpcode;
    vchip=sprintf('%s%s',vchip,char(vpcode));
end



%GENAP
for j=2:2:pjgteks
    gnp(j)=plainteks(j);
end
for ip=1:pjgteks
    if (2*ip)>pjgteks;
        break
    else
        gnp2(ip)=gnp(2*ip);
    end
end
%HILL
en1=length(ek);
eblkknc='';
for eblk1=en1:-1:1
```

```
    ebknc=ek(eblk1);
    eblkknc=sprintf('%s%s',eblkknc,char(ebknc));
end
eblkknc;
ek=eblkknc;
hfk2=gnp2;
hkey=ek;
gnp2asli=gnp2;
%KUNCI
[ha,hb]=size(hkey);
hpjg1=0;
hl=0;
if hb<9
    for hpjg=hb+1:9
        hpjg1=hpjg1+1;
        hkey(hpjg)=hkey(hpjg1);
    end
end

for hit=1:3 %kolom
    for hite=1:3 %baris
        hl=hl+1;
        hmat(hit,hite)=hkey(hl);
    end
end
hmat;
hmatt=double(hmat)-32;
hpjghill=length(hfk2);
[ha,hb]=size(hfk2);
hass=mod(hpjghill,3);
hass1=3-hass;
hpjgg=0;
if hass~=0
    hab=hass1;
    for hi=hb+1:(hpjghill+hab)
        hpjgg=hpjgg+1;
        hfk2(hi)=hfk2(hpjgg);
    end
end
haa=length(hfk2)/3;
%PEMBAGIAN 3-3
hl1=0;
for hi=1:haa
    for hj=1:3
        hl1=hl1+1;
        hm(hi,hj)=hfk2(hl1);
    end
end
hillk='';
for hi2=1:haa
    hk1=hm(hi2,1:3);
```

```
    hk2=hk1';
    hk3=double(hk2)-32;
    hmatt;
    hmat3=round(hmatt*hk3);
    henk=(mod(hmat3,95));
    henk=henk+32;
    henk2=henk';
    hillk=sprintf('%s%s',hillk,char(henk2));
end
hillk;
pdk=length(hillk);
pgnp2asli=length(gnp2asli);
if pdk>pgnp2asli
    pdk=pgnp2asli;
end
for i=1:pgnp2asli
    hillkfix(i)=hillk(i);
end
hillkfix;
%GABUNG
ga=double(vchip);
gb=double(hillkfix);
gp1=length(ga);
gp2=length(gb);
gp3=gp1-gp2;
if gp3~=0
    gb(end+1)=nan;
else
    gb=gb;
end
gr = [ga;gb];
gr = gr(:);
gr1=gr(~isnan(gr))';
enkripsihasil=char(gr1);
```

LAMPIRAN E. (Skrip Dekripsi Penggabungan *Vigenere Cipher* dan *Hill Cipher*)

devigell.m

```
function ggg=devigell(plain,ek)
pjgteks=length(plain);
%GANJIL
for i=1:2:pjgteks
    gjl(i)=plain(i);
end
for ig=1:pjgteks
    if (2*ig-1)>pjgteks
        break
    else
        gjl2(ig)=gjl(2*ig-1);
    end
end
vteks=gjl2;
vkey=ek;
vplain='';
for vi=1:length(vteks)
    vp=double(vteks(vi));
    vp=vp-32;
    vk=double(vkey(vi))-32;
    vpcode=mod(vp-vk,95);
    vpcode=vpcode+32;
    vplaincode(vi)=vpcode;
    vplain=sprintf('%s%s',vplain,char(vpcode));
end
%GENAP
for j=2:2:pjgteks
    gnp(j)=plain(j);
end
for ip=1:pjgteks
    if (2*ip)>pjgteks
        break
    else
        gnp2(ip)=gnp(2*ip);
    end
end
dn1=length(ek);
dblkknc='';
for dblk1=dn1:-1:1
    dbknc=ek(dblk1);
    dblkknc=sprintf('%s%s',dblkknc,char(dbknc));
end
ek=dblkknc;
hfk2=gnp2;
hkey=ek;
```

```
hdeek=double(gnp2)-32;
gnp2asli=gnp2;
[ha,hb]=size(hkey);
hpjg1=0;
hl=0;
if hb<9
    for hpjg=hb+1:9
        hpjg1=hpjg1+1;
        hkey(hpjg)=hkey(hpjg1);
    end
end
for hit=1:3 %kolom
    for hite=1:3 %baris
        hl=hl+1;
        hmat(hit,hite)=hkey(hl);
    end
end
hmatt=double(hmat)-32;
hpjghill=length(hfk2);
[ha,hb]=size(hfk2);
hass=mod(hpjghill,3);
hass1=3-hass;
hpjgg=0;
if hass~=0
    hab=hass1;
    for hi=hb+1:(hpjghill+hab)
        hpjgg=hpjgg+1;
        hfk2(hi)=hfk2(hpjgg);
    end
end
haa=length(hfk2)/3;
deka=hmatt;
dekb=det(deka);
dekc=round(det(deka)*inv(deka));
[G, C] = gcd(dekb,95);
if G==1
    dekf=mod(C,95);
end
dekf;
dekg=dekf*dekc;
dekh=mod(dekg,95);
hmm=dekh;
cek=mod(hmatt*hmm,95);
hbb=length(hdeek)/3;
hl1=0;
for hi=1:haa
    for hj=1:3
        hl1=hl1+1;
        hm(hi,hj)=hfk2(hl1);
    end
end
```

```
dekhmm=hm';
hdk='';
for hi2=1:haa
    hk1=hm(hi2,1:3);
    hk2=hk1';
    hk3=double(hk2)-32;
    hmm2=round(hmm*hk3);
    hdek=(mod(hmm2,95));
    hdek=hdek+32;
    hdek2=hdek';
    hdk=sprintf('%s%s',hdk,char(hdek2));
end
pdk=length(hdk);
pgnp2asli=length(gnp2asli);
if pdk>pgnp2asli
    pdk=pgnp2asli;
else
    pdk=pdk;
end
for ki=1:pdk
    hdkfix(ki)=hdk(ki);
end
ga=double(vplain);
gb=double(hdkfix);
gp1=length(ga);
gp2=length(gb);
gp3=gp1-gp2;
if gp3~=0
    gb(end+1)=nan;
else
    gb=gb;
end
gr = [ga;gb];
gr = gr(:);
gr1=gr(~isnan(gr))';
ggg=char(gr1);
```