PENGKODEAN PLAINTEKS MENGGUNAKAN GABUNGAN *SHIFT CHIPER* DAN TRANSPOSISI DIAGONAL

SKRIPSI

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

Oleh

Tri Lindawati
NIM 071810101081

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2015

PERSEMBAHAN

Skripsi ini saya persembahkan untuk:

1. Orangtuaku Bapak Sudarto dan Ibu Aspiyah yang selalu memberikan doa, restu dan motivasi dalam perjalanan hidupku;
2. Suamiku kakanda Nurlatiful Wahid yang selalu mendampingiku, memberikan support dan semangat;
3. Guru-guruku sejak taman kanak-kanak sampai dengan perguruan tinggi yang telah mendidik, membimbing dan mengajariku tentang pentingnya ilmu;
4. Sahabat-sahabatku Umi Kholila, Hilyatul A, Fatihatul Ayatillah, Mike Ardila, Silvia Anggraeni, Janter Sinaga, Wika Anggani, Yulan Isa Puspita, Prastowo Sandi Asmoro, Dimas yang selalu membantu dan memberiku semangat;
5. Almamater Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

MOTTO

Sesungguhnya bersama kesulitan ada kemudahan. Maka, apabila kamu
telah selesai (dari suatu urusan) tetaplah bekerja keras
(untuk urusan yang lain) dan hanya
kepada Tuhanmulah hendaknya
engkau berharap.
(terjemahan Surat *Al-Insyiroh* Ayat 6-8)[*)]

Orang terkuat bukan mereka yang selalu menang, melainkan
mereka yang tetap tegar ketika mereka jatuh.
(Khalil Gibran)

[*)] Departemen Agama Republik Indonesia. 2004. *Al-Qur'an dan Terjemahannya*. Bandung: CV Penerbit Diponegoro.

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Tri Lindawati

NIM : 071810101081

menyatakan dengan sesungguhnya bahwa karya ilmiah yang berjudul "Pengkodean Plainteks Menggunakan Gabungan *Shift Chiper* dan Transposisi Diagonal" adalah benar-benar hasil karya sendiri, kecuali kutipan yang sudah saya sebutkan sumbernya, belum pernah diajukan pada 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, 2015

Yang menyatakan,

Tri Lindawati

NIM. 071810101081

SKRIPSI

PENGKODEAN PLAINTEKS MENGGUNAKAN GABUNGAN *SHIFT CHIPER* DAN TRANSPOSISI DIAGONAL

Oleh

Tri Lindawati
NIM 071810101081

Pembimbing:

Dosen Pembimbing Utama : Kiswara Agung Santoso, S.Si., M.Kom.
Dosen Pembimbing Anggota : Ika Hesti Agustin, S.Si., M.Si.

PENGESAHAN

Skripsi yang berjudul "Pengkodean Plainteks Menggunakan Gabungan *Shift Chiper* dan Transposisi Diagonal" telah diuji dan disahkan pada:

hari :

tanggal :

tempat : Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Tim Penguji:

Ketua,	Sekretaris
Kiswara Agung Santoso, S.Si, M.Kom.	Ika Hesti Agustin, S.Si, M.Si.
NIP.197209071998031003	NIP.198408012008012006
Penguji I,	Penguji II,
Kusbudiono, S.Si., M.Si.	Bagus Julianto, S.Si.
NIP. 197704302005011001	NIP 198007022003121001

Mengesahkan

Dekan,

Prof. Drs. Kusno, DEA., Ph.D.

NIP 196101081986021001

RINGKASAN

Pengkodean Plainteks Menggunakan Gabungan *Shift Chiper* dan Teknik Transposisi Diagonal; Tri Lindawati, 071810101081; 2015; 30 halaman; Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Gabungan *shift chiper* dan transposisi diagonal merupakan metode dalam kriptografi (pengkodean) yang digunakan untuk aspek keamanan pesan atau data yang akan dikirim, dimana pesan ini menjadi aman dan hanya pengirim dan penerima yang bisa mengetahui isi pesan tersebut. Kriptografi dibagi menjadi dua yaitu kriptografi klasik dan kriptografi modern. Perbedaan utama kriptografi klasik dan kriptografi modern adalah enkripsi berbasis karakter pada kriptografi klasik dan enkripsi berbasis bit pada kriptografi modern. Ada dua teknik dasar yang biasa digunakan algoritma kriptografi klasik yaitu chiper transposisi dan chiper substitusi. Chiper transposisi adalah satu jenis teknik pengenkripsian pesan dengan cara mengubah urutan huruf-huruf yang ada di dalam plainteks (pesan yang belum dienkripsi).

Dalam skripsi ini kriptografi yang digunakan adalah kriptografi simetri dan sebagai algoritma enkripsi dan dekripsinya adalah gabungan *shift chiper* dan transposisi diagonal. Kunci yang digunakan pada algoritma ini adalah bilangan bulat dan karakter yang digunakan adalah karakter pada table ASCI namun karakter yang digunakan dibatasi hanya dari karakter 32 sampai 126 saja.

Skripsi ini bertujuan untuk membuat sebuah algoritma pengkodean sederhana menggunakan gabungan *shift chiper* dan teknik transposisi diagonal namun tidak mudah diketahui oleh orang lain. Adapun langkah-langkah algoritmanya adalah mengenkripsi pesan terlebih dahulu. Pertama, geser nilai numerik plainteks sepanjang *K*. Kedua, masukkan hasil pengkodean *shift chiper* dalam matriks berordo $n \times n$ dimana $n = \lceil \sqrt{P} \rceil$. Ketiga, baca secara diagonal kiri bawah ke kanan atas. Setelah didapat chiperteksnya maka untuk mengembalikan menjadi plainteks lagi yaitu dengan mendekripsi chiperteks menggunakan kunci

yang sama. Pertama, geser nilai numerik chiperteks sepanjang *K*. Kedua, masukkan hasil pengkodean *shift chipper* dalam matriks berordo $n \times n$ dimana $n = \lceil \sqrt{P} \rceil$. Ketiga, baca secara diagonal kiri ke kanan

Dari hasil penelitian yang dilakukan, dapat ditarik beberapa kesimpulan. Pertama, dalam proses enkripsi *shift chipper* digunakan pada algoritma pertama lalu yang kedua menggunakan transposisi diagonal dengan pola baca kiri bawah ke kanan atas. Kedua, dalam proses dekripsi algoritma pertama yang digunakan adalah *shift chipper* dan algoritma kedua adalah transposisi diagonal dengan pola baca kiri ke kanan. Ketiga, pengkodean plainteks menggunakan gabungan *shift chipper* dan transposisi diagonal lebih baik dari pada pengkodean plainteks yang menggunakan *shift chipper* saja atau pengkodean yang menggunakan transposisi diagonal saja.

PRAKATA

Alhamdulillah, puji syukur kehadirat Allah SWT atas segala rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "Pengkodean Plainteks Menggunakan Gabungan Shift Chiper dan Teknik Transposisi Diagonal". Skripsi ini disusun untuk memenuhi salah satu syarat menyelesaikan pendidikan strata satu (S1) pada Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Penyusunan skripsi ini tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. Kiswara Agung Santoso, S.Si, M.Kom., selaku Dosen Pembimbing Utama dan Ika Hesti Agustin, S.Si, M.Si., selaku Dosen Pembimbing Anggota yang telah meluangkan waktu, pikiran, dan perhatian dalam penulisan skripsi ini;
2. Kusbudiono, S.Si, M.Si dan Bagus Julianto, S.Si, selaku dosen penguji atas saran-saran yang diberikan;
3. Yuliani Setia Dewi, S.Si, M.Si selaku Dosen Pembimbing Akademik yang telah membimbing selama masa perkuliahan;
4. Sahabat-sahabat tercinta Umi Kholila, Hilya, Fatihatul A, Mike Ardila, Yulan Isa Puspita, Wika Anggani, Silvia Anggraeni, Janter Sinaga, Ratih Kartika, yang selalu memberikan semangat, keceriaan, serta motivasi;
5. Teman-teman angkatan 2007 Jurusan Matematika yang tidak bisa disebutkan satu persatu terima kasih atas keceriaan dan kebersamaannya selama ini;
6. Semua pihak yang tidak dapat disebutkan satu per satu.

Penulis juga menerima segala kritik dan saran dari semua pihak demi kesempurnaan skripsi ini. Akhirnya penulis berharap, semoga skripsi ini dapat bermanfaat.

Jember, 2015

Penulis

DAFTAR ISI

Halaman

DAFTAR TABEL

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR LAMPIRAN

Halaman

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Masalah keamanan dan kerahasiaan merupakan salah satu aspek penting dalam suatu sistem informasi, dalam hal ini sangat terkait dengan betapa pentingnya informasi tersebut dikirim dan diterima oleh orang yang berkepentingan. Informasi tidak akan berguna lagi apabila di tengah jalan informasi tersebut disadap atau dibajak orang yang tidak berhak. Salah satu solusi yang dapat digunakan untuk menyelesaikan masalah tersebut adalah menggunakan kriptografi. Kriptografi adalah suatu ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi, seperti kerahasiaan data, keabsahan data, integritas data, serta autentikasi data (Menezesdkk, 1996).

Kriptografi menggunakan berbagai macam teknik dalam upaya untuk mengamankan data. Pengiriman data dan penyimpanan data dengan media elektronik memerlukan suatu proses yang dapat menjamin keamanan dan keutuhan dari data yang dikirimkan tersebut. Data tersebut harus tetap rahasia selama pengiriman dan harus tetap utuh pada saat penerimaan di tujuan. Untuk memenuhi hal tersebut, dilakukan proses penyandian yaitu enkripsi (proses yang digunakan untuk menyamarkan/menyembunyikan plainteks) dan dekripsi (mengembalikan cipherteks menjadi plainteks) terhadap data yang akan dikirimkan. Kriptografi dibagi menjadi dua yaitu kriptografi klasik dan kriptografi modern, yang menjadi perbedaan utama kriptografi klasik dengan kriptografi modern adalah enkripsi berbasis karakter pada kriptografi klasik dan enkripsi berbasis bit pada kriptografi modern. Salah satu jenis dari kriptografi klasik adalah chipper transposisi. Nama lain untuk metode ini adalah permutasi, pada metode permutasi ini ada banyak pola yang dapat digunakan untuk merubah plainteks menjadi cipherteks seperti zig-zag, segitiga dan spiral.

Mulyono, E (2012) telah mengkaji salah satu teknik dalam kriptografi simetris yaitu tentang pengkodean pesan teks menggunakan *logika XOR* dengan

satu karakter kunci. Asmara, FD (2009) mengkaji tentang salah satu teknik dalam penyandian, yaitu penyandian Hill menggunakan kode plainteks kembar. Pohan, YR (tanpa tahun) mengkaji tentang perbandingan berbagai macam algoritma chiper transposisi semua algoritma cipher transposisi mempunyai kelemahan frekuensi kemunculan karakter cipherteks sama dengan plainteks sehingga bisa diserang menggunakan analisis frekuensi. Mukmin, I (2009) mengkaji tentang modifikasi playfair chiper dengan kombinasi bifid, caesar, dan transpositional chiper yang masih terdapat beberapa kelemahan dalam algoritma modifikasi khususnya belum adanya penanganan otomatis untuk mengolah plainteks hasil dekripsi menjadi plainteks yang persis sama dengan plainteks asli.

Dari penelitian yang sudah ada sebelumnya maka perlu dilakukan penelitian untuk menyempurnakan penelitian sebelunya Ide dasar cipher transposisi adalah cipherteks diperoleh dengan mengubah posisi huruf di dalam plainteks, dengan kata lain algoritma ini melakukan transpose terhadap rangkaian huruf di dalam plainteks. Dari macam-macam pola yang ada di atas maka penulis tertarik untuk mengkaji tentang pengkodean plainteks menggunakan gabungan *shift chiper* dan transposisi diagonal agar menghasilkan chiper yang lebih kuat.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas permasalahan yang akan dibahas adalah bagaimana mengkodekan plainteks menggunakan gabungan *shift chiper* dan transposisi diagonal dan bagaimana program dari pengkodean tersebut jika diterapkan pada program Matlab.

1.3 Tujuan

Adapun tujuan dari skripsi ini adalah untuk membuat pengkodean plainteks menggunakan gabungan *shift chiper* dan transposisi diagonal dan mengetahui program dari pengkodean plainteks menggunakan gabungan *shift chiper* dan transposisi diagonal pada program matlab.

1.4 Manfaat

Manfaat yang diharapkan dalam tulisan ini adalah dapat menjadi tambahan informasi untuk solusi pengkodean yang digunakan bagi semua masyarakat pada umumnya dan matematikawan pada khususnya.

BAB 2. TINJAUAN PUSTAKA

2.1 Kriptografi

Kriptografi secara umum adalah ilmu dan seni untuk menjaga kerahasiaan berita (Schneier, 1996). Selain pengertian tersebut terdapat pula pengertian ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan data, keabsahan data, integritas data, serta autentikasi data (A. Menezes, dkk. 1996).

Munir, R (2006) menjelaskan *Cryptography* (kriptografi) berasal dari bahasa Yunani yaitu dari kata *crypto* yang berarti penulisan *screet* (rahasia), sedangkan *graphein* artinya *writing* (tulisan). Jadi secara sederhana dapat diartikan *screet writing* (tulisan rahasia). Definisi lain dari kriptografi adalah sebuah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan, integritas data serta otentikasi

Menurut Andri (2009), komunikasi yang dilakukan di zaman ini semua membutuhkan kepastian akan keamanan komunikasinya. Terutama komunikasi yang bersifat sensitive, seperti komunikasi militer, transaksi cashless dan lain sebagainya. Terdapat 4 hal yang dibutuhkan keamanannya dalam komunikasi.

a. Kerahasiaan pesan

   Kerahasiaan pesan adalah layanan yang digunakan untuk menjaga isi dari informasi dari siapapun kecuali yang memiliki otoritas atau kunci rahasia untuk membuka atau mengupas informasi yang telah disandi.

b. Integritas pesan

   Integritas data, adalah berhubungan dengan penjagaan dari perubahan data secara tidak sah. Untuk menjaga integritas data, sistem harus memiliki kemampuan untuk mendeteksi manipulasi data oleh pihak-pihak yang tidak berhak, antara lain penyisipan, penghapusan, dan pensubstitusian data lain kedalam data yang sebenarnya.

c. Autentikasi

Autentikasi adalah berhubungan dengan identifikasi atau pengenalan, baik secara kesatuan sistem maupun informasi itu sendiri. Dua pihak yang saling berkomunikasi harus saling memperkenalkan diri. Informasi yang dikirimkan melalui kanal harus diautentikasi keaslian, isi datanya, waktu pengiriman, dan lain-lain.

d. Penyangkalan

Penyangkalan adalah usaha untuk mencegah terjadinya penyangkalan terhadap pengiriman atau terciptanya suatu informasi oleh yang mengirimkan atau membuat.

2.1.1 Istilah dalam Kriptografi

Ada beberapa istilah-istilah yang berhubungan dengan kriptografi yang juga perlu untuk dipahami.

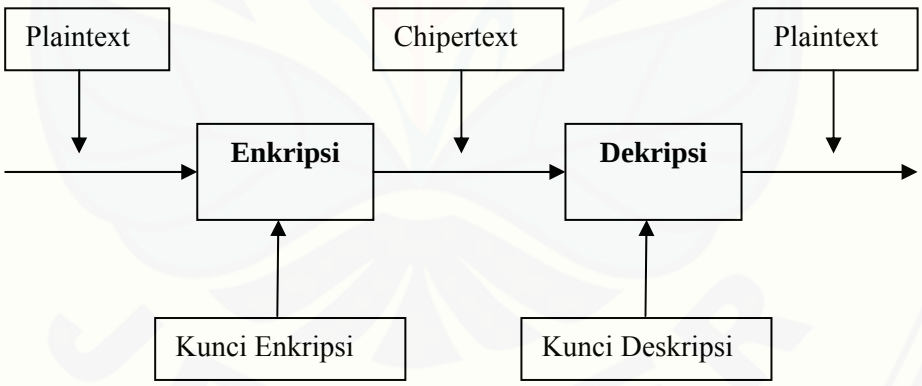
a. Plainteks adalah data asli atau informasi bersifat terbuka yang isinya dapat dibaca dan dipahami secara langsung. Menjadi sumber data untuk proses enkripsi.
b. Cipherteks adalah data sandi hasil proses dekripsi.
c. Cipher adalah algoritma untuk mengubah plainteks menjadi cipherteks menggunakan persamaan matematika. Hasil perubahan dapat berbentuk cipher substitusi, cipher transposisi, atau gabungan dari keduanya.
d. Cipher substitusi adalah algoritma mengubah plainteks menjadi cipherteks dengan cara mengganti menggunakan persamaan matematika tertentu.
e. Cipher transposisi adalah algoritma mengubah plainteks menjadi cipherteks dengan cara menggeser menggunakan persamaan matematika tertentu.
f. Cipher blok adalah algoritma mengubah plainteks menjadi cipherteks untuk setiap blok data. Jumlah data atau besarnya blok adalah tertentu.
g. Kunci (k*ey*) adalah data atau nilai yang sangat spesifik yang diketahui oleh pengirim dan penerima yang berhak. Digunakan bersama sama dengan algoritma kriptografi untuk melakukan proses enkripsi dan dekripsi.

h. Enkripsi adalah proses yang digunakan untuk menyamarkan atau menyembunyikan plainteks. Hasil dari proses enkripsi adalah data sandi (cipherteks).
i. Dekripsi kebalikan dari proses enkripsi yaitu mengembalikan cipherteks menjadi plainteks.
j. Kriptanalis (code breaking) adalah kegiatan untuk mengubah cipherteks menjadi pesan aslinya tanpa mengetahui kunci yang sesuai, dengan coba-coba (trial and error) secara sistematis.

Kriptografi itu sendiri terdiri dari 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. Apabila E dimisalkan dengan sebuah fungsi yang bijektif, maka D adalah inversnya. Dengan demikian dapat dituliskan rumusan umum untuk enkripsi dan deskripsi sebagai berikut:

Enkripsi : $E(P) = C$

Deskripsi : $D(C) = P$ atau dapat juga ditulis $D\big(E(P)\big) = P$



Gambar 2.1 *Diagram proses enkripsi dan dekripsi*

2.1.2 Kriptografi Klasik

Munir, R (2004) menjelaskan bahwa kriptografi klasik merupakan suatu algoritma yang menggunakan satu kunci untuk mengamankan data. Teknik ini sudah digunakan beberapa abad yang lalu. Beberapa teknik dasar yang biasa digunakan pada algoritma jenis ini adalah sebagai berikut:

a. Cipher transposisi

Cipher transposisi adalah salah satu jenis teknik pengenkripsian pesan dengan cara mengubah urutan huruf-huruf yang ada di dalam plainteks (pesan yang belum dienkripsi) menjadi cipherteks (pesan yang telah dienkripsi) dengan cara tertentu agar isi dari pesan tersebut tidak dimengerti kecuali oleh orang-orang tertentu. Cipher transposisi juga memiliki beberapa metode untuk menyelesaikan enkripsi dan dekripsinya. Diantaranya adalah metode zig-zag, segitiga, spiral dan blok.

1) Metode Zig-zag

Metode zig-zag digunakan dengan memasukkan plainteks seperti pola zig-zag.

Plainteks : PERHATIKAN RAKYAT KECIL

			H						N						T					X
		R		A				A		R				A		K			L	
	E				T		K				A		Y				E		I		
P						I						K						C			

Maka chiperteks yang dihasilkan adalah HNTXRAARAKLETKAYEIPIKC

2) Metode Segitiga

Metode segitiga digunakan dengan memasukkan plainteks seperti pola segitiga.

Plainteks : PERHATIKAN RAKYAT KECIL

				P				
			E	R	H			
		A	T	I	K	A		
	N	R	A	K	Y	A	T	
K	E	C	I	L	X	X	X	X

Maka chiperteks yang dihasilkan adalah knearcetaipriklhkyxaaxtxx

3) Metode Spiral

Metode spiral digunakan dengan memasukkan plainteks disusun seperti pola spiral.

Palinteks : PERHATIKAN RAKYAT KECIL

P	E	R	H	A
T	K	E	C	T
A	X	X	I	I
Y	X	X	L	K
K	A	R	N	A

Maka chiperteks yang dihasilkan adalah PTAYKEKXXAREXXRHCILNATIKA

4) Metode Blok

Metode ini melakukan enkripsi menggunakan blok dengan membagi text asli/ plainteks menjadi blok-blok dengan setiap blok mengandung beberapa karakter biasanya 3 atau lebih per blok tergantung perjanjian.

Plaintext : NINJ AASA SSIN MEMA NGKE RENX

K1 K2 K1 K2 K1 K2

Chipertext : GIGB BBIB MMAG RLRB GTCP HLEX

K1 K2 K1 K2 K1 K2

Untuk mengurangi kekurangan huruf digunakan huruf X atau yang lain sesuai perjanjian.

b. Cipher substitusi

Cipher substitusi adalah algoritma kriptografi yang mula-mula digunakan oleh kaisar Romawi, Julius Caesar (sehingga dinamakan juga *caesar cipher*), untuk menyandikan pesan yang ia kirim kepada para gubernurnya. Caranya adalah dengan mengganti (menyulih atau mensubstitusi) setiap karakter dengan karakter lain dalam susunan abjad (alfabet).

Contoh tabel substitusi:

p_i : 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
c_i : **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**

Teknik ini menggunakan permutasi karakter, yang mana dengan menggunakan teknik ini pesan asli tidak dapat dibaca kecuali oleh orang yang memiliki kunci untuk mengembalikan pesan tersebut ke bentuk semula. Adapun Jenis-jenis dari Cipher subsbtitusi :

1) Cipher abjad tunggal (*monoalphabetic cipher* atau *cipher* substitusi sederhana *simple substitution cipher*). Satu karakter di plainteks diganti dengan satu karakter yang bersesuaian. Jadi, fungsi *ciphering*-nya adalah fungsi satu-kesatu.
2) Cipher substitusi homofonik (*Homophonic substitution cipher*). Seperti *cipher* abjad-tunggal, kecuali bahwa setiap karakter didalam plainteks dapat dipetakan ke dalam salah satu dari karakter cipherteks yang mungkin.
3) Cipher abjad-majemuk (*Polyalpabetic substitution cipher*). Merupakan *cipher* substitusi-ganda (*multiple-substitutioncipher*) yang melibatkan penggunaan kunci berbeda.*Cipher* abjad-majemuk dibuat dari sejumlah *cipher* abjad-tunggal,masing-masing dengan kunci yang berbeda. Kebanyakan *cipher* abjad-majemuk adalah *cipher* substitusi periodik yang didasarkan pada periode *m*.
   Misalkan plainteks *P* adalah $P = p1p2 \ldots pmpm+1 \ldots p2m \ldots$
   maka cipherteks hasil enkripsi adalah $Ek(P) = f1(p1)\ f2(p2) \ldots fm(pm)\ fm+1(pm+1) \ldots f2m(p2m) \ldots$
   yang dalam hal ini *pi* adalah huruf-huruf di dalam plainteks. Untuk $m = 1$, *cipher*-nya ekivalen dengan *cipher* abjad tunggal.
5) Cipher substitusi poligram (*Polygram substitution cipher*). Blok karakter disubstitusi dengan blok cipherteks. Misalnya ABA diganti dengan **RTQ**, ABB diganti dengan **SLL**, dan lain-lain. Munir, R (2005)

c. Shift chipper (kode geser)

Teknik sustitusi kode geser (shift) dengan modulus 26 memberikan angka ke setiap alphabet seperti a 0, B 1 ..... Z 25. Agar lebih jelas, perhatikan contoh di bawah ini:

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

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Teks asli: “We Will Meet at Mild Nigth”

Kalimat di atas mendapat angka dari setiap huruf sebagai berikut:

22 4 22 8 11 12 4 4 19 0 19 12 8 3 13 8 6 7 19

Untuk mendapatkan teks kode, kita mempunyai kunci 11. Dengan menambahkan setiap nilai dari teks asli dengan kunci 11 maka didapat:

7 15 7 19 22 22 23 15 15 4 11 4 23 19 14 24 19 17 18 4

Jika leibih 20, setelah ditambah dengan kunci maka akan dikurangi dengan 26. Misalnya 22 + 11= 33 – 26 = 7. Setelah dikonversi menjadi huruf maka akan didapatkan teks kode: “HPHTWWXPPELEXTOYTRSE”

Teks asli : We will meet at mid night

Teks kode : HPHTWWXPPELEXTOYTRSE

Kunci : 11

(Ariyus, D. 2008)

Sandi geser adalah salah satu teknik enkripsi paling sederhana dan paling terkenal. Sandi ini termasuk sandi substitusi dimana setiap huruf pada teks terang (*plaintext*) digantikan oleh huruf lain yang memiliki selisih posisi tertentu dalam alfabet. Sandi geser (shift cipher) merupakan generalisasi dari sandi caesar, yaitu tidak membatasi pergeseran sebanyak tiga huruf. Jadi ada sebanyak 26 kunci pergeseran yang bisa digunakan. Secara umum dapat dituliskan dengan persamaan berikut ini:

$$e_K(x) = (x + k) mod\ 26$$
$$e_K(y) = (x - k) mod\ 26$$

Untuk K dengan $0 \leq K \leq 25$ dan $x, y \in \mathbb{Z}_{26}$

2.2 Matriks

Matriks adalah kumpulan bilangan berbentuk persegi panjang yang disusun menurut baris dan kolom. Bilangan-bilangan yang terdapat di suatu matriks disebut dengan elemen atau anggota matriks. Dengan representasi matriks, perhitungan dapat dilakukan dengan lebih terstruktur. Pemanfaatannya

misalnya dalam menjelaskan persamaan linear, transformasi koordinat, dan lainnya (Yahya, dkk. 2005).

a. Matriks Bujur Sangkar

Matriks bujur sangkar adalah matriks yang memiliki ordo $n \times n$ atau banyaknya baris sama dengan banyaknya kolom yang terdapat dalam mtriks tersebut. Matriks ini disebut juga dengan matriks persegi berordo n. Matriks bujur sangkar termasuk salah satu jenis matriks yang dilihat dari ordonya.

Contoh : $A = \begin{pmatrix} 2 & 4 \\ 5 & 1 \end{pmatrix}$

Jumlah baris = jumlah kolom

b. Matriks Diagonal

Matriks ini termasuk Matriks persegi karena mensyaratkan banyak baris sama dengan banyak kolom. Suatu matriks persegi disebut sebagai matriks diagonal jika semua komponen diagonal utamanya tidak nol dan semua komponen lainnya adalah nol.

Contoh $A = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$

Matriks diagonal ini merupakan salah satu jenis matriks yang dilihat dari elemen-elemen penyusunnya.

c. Matriks Transpose

Matriks transpose adalah jika sebuah matriks, dimensi baris dengan kolomnya ditukar. Dalam pengertian lain, matriks transpose yaitu matriks yang diperoleh dari memindahkan elemen-elemen baris menjadi elemen pada kolom atau sebaliknya. Umumnya transpose matriks di lambangkan tanda petik (A‘) atau huruf T di atasnya (A^T). Secara definisi matriks $m \times n$ akan menjadi $n \times m$.

Beberapa Sifat Matriks Transpose:

1) $(A + B)^T = A^T + B^T$
2) $(A^T)^T = A$
3) $\lambda\,(A^T) = (\lambda A^T)$, bila λ suatu scalar
4) $(AB)^T = B^T\ A^T$

Adapun jenis-jenis matriks transpose adalah sebgai berikut:

a. Matriks simetris

Matriks simetris adalah matriks elemen aij pada baris ke-i dan kolom ke-j sama dengan elemen a_{ji} pada baris ke-j dan kolom ke-i. Hubungan antara elemen tersebut berarti bahwa transpose dari sebuah matriks adalah sama dengan matriks asal, maka matriks simetris adalah:

$A = A^T$ jika A adalah matriks simetri

b. Matriks miring (*skew*)

Matriks miring adalah matriks yang antara elemen-elemen yang tidak terletak pada diagonal utamanya mempunyai hubungan negatif. Artinya $a_{ij} = -a_{ji}$ dan elemen diagonal utamanya boleh terdiri atas sembarang bilangan asalkan tidak nol semuanya $a_{ii} \neq 0$

Contoh : $\begin{bmatrix} 1 & 2 & 3 \\ -2 & 4 & 5 \\ -3 & 5 & 6 \end{bmatrix}$

c. Matriks miring simetris (*skew symetris*)

Matriks miring adalah matriks yang jika semua elemen diagonalnya adalah nol semuanya dan transpose dari matriks ini sama dengan matriks asal dengan tanda negatif. Matriks skew simetris mempunyai syarat:

A $= -A^T$

a_{ij} $= -a_{ij}$ dan $a_{ii} = 0$

Contoh :

$A = \begin{bmatrix} 0 & 2 & 3 \\ -2 & 0 & -5 \\ -3 & 5 & 0 \end{bmatrix}$ $\quad -A^T = \begin{bmatrix} 0 & -2 & -3 \\ 2 & 0 & 5 \\ -3 & -5 & 0 \end{bmatrix}$

2.3 Sistem Bilangan

Sistem bilangan (number system) adalah suatu cara mewakili besaran dari suatu item fisik. Sistem bilangan yang banyak dipergunakan manusia adalah sistem bilangan desimal, yaitu sistem bilangan yang menggunakan 10 macam

simbol untuk mewakili suatu besaran. Sistem ini banyak digunakan karena kemudahannya untuk kita gunakan dalam perhitungan sehari-hari (Wulandari, Tanpa Tahun).

2.3.1 Bilangan Desimal

Sistem ini menggunakan 10 macam simbol yaitu 0,1,2,3,4,5,6,7,8 dan 9. Bentuk nilainya dapat berupa integer desimal atau pecahan. Integer desimal adalah nilai desimal yang bulat, misalnya 8598 dapat diartikan:

$$\begin{aligned} 8 \times 10^3 &= 8000 \\ 5 \times 10^2 &= 500 \\ 9 \times 10^1 &= 90 \\ 8 \times 10^0 &= 8 + \\ \hline & 8598 \end{aligned}$$

Adapun 8598 disebut absolute value yaitu merupakan nilai untuk masing-masing digit bilangan. Sedangkan baris yang dipangkatkan disebut position value. Position value adalah merupakan penimbang atau bobot dari masing-masing digit tergantung dari letak posisinya, yaitu bernilai basis dipangkatkan dengan urutan posisinya.

2.3.1 Bilangan Bulat

Bilangan bulat adalah bilangan yang tidak mempunyai pecahan desimal, misalnya 4, 27, 342, 54, 1, dsb. Berlawanan dengan bilangan bulat adalah bilangan riil yang mempunyai titik desimal, seperti 1,2; 25,45; 0,09, dsb. Sifat pembagian pada bilangan bulat:

Misalkan a dan b adalah dua buah bilangan bulat dengan syarat $a \neq 0$. Dinyatakan bahwa a habis membagi b jika terdapat bilangan bulat c sedemikian sehingga $b = ac$.

Misalkan m dan n adalah dua buah bilangan bulat dengan syarat $n > 0$. Jika m dibagi dengan n maka terdapat dua buah bilangan bulat unik q $(quotient)$ dan r $(remainder)$, sedemikian sehingga

$$m = nq + r \qquad (2.1)$$

dengan $0 \leq r < n$ (Munir, 2012)

2.3.2 Aritmatika Modulo

Dalam hitungan modular diberikan bilangan bulat positif m, disebut modulus dan dua bilangan bulat sebarang yang selisihnya adalah kelipatan bilangan bulat itu dipandang sebagai "sama" atau "setara" terhadap modulus.

Jika m bilangan bulat positif dan a serta b bilangan bulat sebarang, maka dikatakan bahwa a setara b modulo m, ditulis sebagai:

$$a = b(mod\ m)$$

Jika $a - b$ adalah bilangan bulat kelipatan m.

Untuk modulus m sebarang dapat dibuktikan bahwa setiap bilangan bulat a adalah setara modulo m terhadap salah satu bilangan bulat

$$0,1,2, \dots, m-1$$

Bilangan bulat ini disebut sisa *(residue)* dari a modula m, dan dituliskan:

$$Z_m = \{0,1,2, \dots, m-1\}$$

Dalam hitungan biasa setiap bilangan tak-nol a mempunyai balikan atau invers perkalian yang dinyatakan oleh a^{-1}, sedemikian rupa sehingga

$$a\,a^{-1} = a^{-1}\,a = 1$$

Dalam hitungan modularnmempunyai konsep yang berpadanan berikut:

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

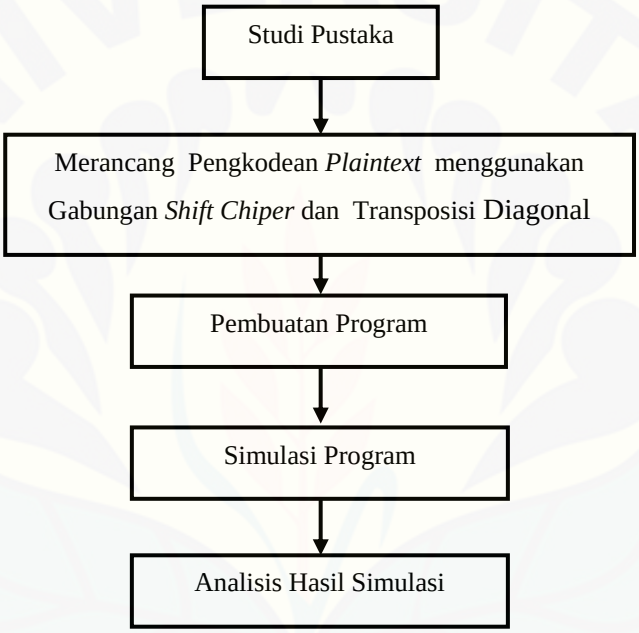
Jika a dan m tidak mempunyai faktor prima bersama, maka a mempunyai balikan modulo m unik; sebaliknya jika a dan m mempunyai faktor prima bersama maka a tidak mempunyai balikan modulo m (Asmara, 2009).

BAB 3. METODE PENELITIAN

3.1 Data Penelitian

Data yang digunakan pada penelitian ini berupa karakter ASCI. Karakter tersebut selanjutnya digunakan sebagai *plaintext* dan *ciphertext.*

3.2 Langkah-langkah Penelitian



Gambar 3.2 Skema langkah-langkah penelitian

Penjelasan langkah-langkah penelitian pada skema diatas adalah sebagai berikut.

a. Studi Pustaka

   Studi pustaka ini dilakukan untuk memperlajari berbagai literatur yang berkaitan dengan kriptografi, khususnya pengkodean menggunakan gabungan *shift chiper* dan transposisi diagonal.

b. Pengkodean pesan menggunakan gabungan *shift chiper* dan transposisi diagonal

   Ada tiga tahap dalam langkah kedua ini, antara lain:

Penentuan Kunci (*key*)

Pada tahap ini terlebih dahulu ditentukan kunci yang digunakan untuk mengkodekan plainteks. Kunci merupakan bilangan bulat.

1) Enkripsi

Pada tahap kedua ini menetukan chiperteks dari plainteks menggunakan persamaan:

$$C = (P + k)\ mod\ 256$$

Selanjutnya chiperteks disusun ke dalam bentuk matriks $n \times n$ dimana $n = \lceil \sqrt{P} \rceil$ dan chiperteks didapat dari aturan baca secara diagonal.

2) Dekripsi

Pada tahap yang ketiga, *ciphertext* yang sudah didapatkan dari proses enkripsi dikembalikan seperti semula menggunakan persamaan:

$$P = (C - k)\ mod\ 256.$$

Selanjutnya plainteks disusun ke dalam bentuk matriks $n \times n$ dimana $n = \lceil \sqrt{P} \rceil$ dan plainteks didapat dari aturan baca diagonal. Hal ini bertujuan untuk mengembalikan *ciphertext* ke *plaintext* awal.

c. Pembuatan Program

Pada langkah yang ketiga, terlebih dahulu dibuat algoritma pembentukan kunci, enkripsi, dan dekripsi dengan gabungan *shift chiper* dan transposisi diagonal. Setelah itu akan dibuat program dengan bantuan *software* MATLAB berdasarkan algoritma yang telah dibuat sebelumnya.

d. Simulasi Program

Setelah program selesai dibuat, langkah selanjutnya adalah simulasi program tersebut. Program diuji dengan menggunakan teks pesan dengan karakter yang telah ditentukan yaitu karakter ASCI namun karakter yang bisa digunakan hanya karakter dari 32 sampai 126 saja.

e. Analisis Hasil Simulasi

Langkah terakhir adalah menganalisis hasil dari simulasi program. Analisis dilakukan dengan membandingkan karakter sebelum dan sesudah proses enkripsi.

BAB 4. HASIL DAN PEMBAHASAN

Dalam bab ini akan dibahas mengenai pengkodean plainteks menggunakan gabungan *shift chiper* dan transposisi diagonal. Pertama akan dijelaskan algoritma kriptosistem tersebut secara umum. Kemudian akan dijelaskan pengerjaan kriptosistem ini secara manual. Pada bagian terakhir dari bab ini akan dibahas hasil dari penggunaan algoritma dan program dalam sistem pengkodean tersebut.

4.1 Hasil

4.1.1 Algoritma Pengkodean Plainteks Menggunakan Gabungan *Shift Chiper* dan Transposisi Diagonal

Merujuk pada tinjauan pustaka, algoritma penyelesaian masalah kriptosistem, yakni pesan/plainteks yang mula-mula dibuat kemudian dienkripsi untuk dijadikan chiperteks lalu setelah itu didekripsi untuk kembali dijadikan pesan/plainteks seperti semula. Kriptosistem dalam skripsi ini merupakan kriptografi simetri. Sebagaimana telah dijelaskan di tinjauan pustaka bahwa pengirim dan penerima pesan telah berbagi kunci sebelumnya. Akan diuraikan lebih lanjut, bagaimana kriptosistem ini bekerja dalam pengkodean pesan menggunakan gabungan *shift chiper* dan transposisi diagonal.

Adapun algoritmanya adalah sebagai berikut:

a. Menentukan plainteks
b. Mengenkripsi pesan

   Dalam mengenkripsi pesan dilakukan dengan tahapan berikut;

   1) Tentukan kunci
   2) Geser plainteks dengan pergeseran karakter sesudahnya sesuai dengan kunci yang telah ditentukan dan menggunakan persamaan:

$$C = (P + k)\ mod\ 256$$

3) Hitung P = banyaknya plainteks

4) Bentuk matriks $n \times n$ dengan $n = \lceil\sqrt{P}\rceil$

5) Tulis chiperteks ke dalam matriks bujur sangkar

$$M_{n\times n} = \begin{bmatrix} C_1 & C_2 & C_3 \\ C_4 & C_5 & C_6 \\ C_7 & C_8 & C_n \end{bmatrix}$$

6) Huruf chiperteks adalah elemen matriks M yang dibaca menurut diagonal kiri bawah ke kanan atas.

$$M_{n\times n} = \begin{bmatrix} C_1 & C_2 & C_3 \\ C_4 & C_5 & C_6 \\ C_7 & C_8 & C_n \end{bmatrix}$$

$C = [C_1, C_4, C_2, C_{7,\ldots,} C_n]$

c. Mendekripsi pesan

1) Tentukan kunci

2) Geser chiperteks dengan pergeseran karakter sebelumnya sesuai dengan kunci yang telah ditentukan dan menggunakan persamaan:

$$P = (C - k)\ mod\ 256$$

3) Hitung C = banyaknya chiperteks

4) Bentuk matriks $n \times n$ dengan $n = \lceil\sqrt{C}\rceil$

5) Tulis plainteks ke dalam matriks

$$M_{n\times n} = \begin{bmatrix} P_1 & P_2 & P_3 \\ P_4 & P_5 & P_6 \\ P_7 & P_8 & P_n \end{bmatrix}$$

6). Huruf plainteks adalah elemen matriks M yang dibaca menurut diagonal kiri ke kanan

$$M_{n\times n} = \begin{bmatrix} P_1 & P_2 & P_3 \\ P_4 & P_5 & P_6 \\ P_7 & P_8 & P_n \end{bmatrix}$$

$P = [P_1, P_2, P_3, P_{4,\ldots,} P_n]$

4.1.2 Penyelesaian Pengkodean Plainteks Menggunakan Gabungan *Shift Chiper* dan Transposisi Diagonal

Dalam subbab ini akan diuraikan penggunaan algoritma yang telah dibuat pada subbab sebelumnya secara manual sebagai contoh penerapan dari algoritma tersebut dalam penyelesaian pengkodean plainteks menggunakan gabungan *shift chiper* dan transposisi diagonal.

Contoh :

1. Menentukan kunci *K*, misal *K* = 4
2. Membuat pesan yang akan dikirim. Misalkan pesan yang akan dikirim adalah "aku juga bisa". Maka $P = 11$
3. Mengenkripsi pesan dengan tahapan-tahapan seperti berikut:

   $C = (P + k)\ mod\ 256$

   Dari kunci yang sudah ditentukan untuk mengenkripsi pesan maka plainteks yang akan dikirim akan digeser 4 langkah sesuai nilai numerik pada tabel ASCI.

a	**k**	**u**	**j**	**u**	**g**	**a**	**b**	**i**	**s**	**a**
97	**107**	**117**	**106**	**117**	**103**	**97**	**98**	**105**	**115**	**97**
e	**o**	**y**	**n**	**y**	**k**	**e**	**f**	**m**	**w**	**e**
101	**111**	**121**	**110**	**121**	**107**	**101**	**102**	**109**	**119**	**101**

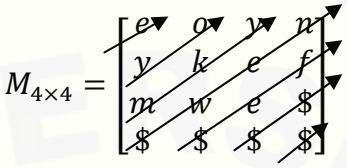
Tabel 4.1. Tabel Perubahan Plainteks yang Dienkripsi

Setelah langkah diatas maka panjang plainteks dihitung untuk menentukan matriks $M_{n\times n}$. Jumlah huruf "aku juga bisa" adalah 11 dan akar kuadrat yang mendekati 11 adalah $\sqrt{9}$ yaitu 3. 11 = $\sqrt{9}$ sisa 2, maka hasilnya adalah 3 sisa 2. Karena masih ada sisa bilangan maka akan dibulatkan keatas dan hasil dari pembulatan dari akar kuadrat adalah 4, jadi $n = 4$.

Selanjutnya pesan/plainteks disusun menjadi matriks $M_{n\times n}$. Jika ada baris atau kolom yang kosong maka diisi dengan karakter yang bernilai 4 atau nilai sesuai kunci yang diketahui, misal kunci nya adalah 4 maka tempat yang kosong pada matriks diisi dengan karakter $. Bentuk matriksnya adalah seperti berukut:

$$M_{4\times4} = \begin{bmatrix} e & o & y & n \\ y & k & e & f \\ m & w & e & \$ \\ \$ & \$ & \$ & \$ \end{bmatrix}$$

Jika pesan/plainteks sudah berbentuk matriks, maka untuk menentukan chiperteksnya matriks tersebut dibaca menurut pola baca diagonal kiri bawah ke kanan atas seperti berikut:



Jadi chiperteks yang diperoleh adalah "*eyomkywenef$$$*".

4. Mendekripsi pesan dengan tahapan berikut

Menentukan kunci yang akan digunakan untuk mendekripsi chiperteks menjadi plainteks, kunci yang digunakan untuk mendekripsi pesan sama dengan kunci yang digunakan pada saat pesan/plainteks dienkripsi, kuncinya yaitu 4. Jika kunci sudah diketahui maka chiperteks digeser dengan karakter ke empat sebelumnya sesuai nilai nuerik plainteks seperti tabel berikut:

e	**o**	**y**	**n**	**y**	**k**	**e**	**f**	**m**	**w**	**e**
101	**111**	**121**	**110**	**121**	**107**	**101**	**102**	**109**	**119**	**101**
a	**k**	**u**	**j**	**u**	**g**	**a**	**b**	**i**	**s**	**a**
97	**107**	**117**	**106**	**117**	**107**	**97**	**98**	**105**	**115**	**97**

Tabel 4.2. Tabel Perubahan Chiperteks yang Didekripsi

Setelah langkah diatas maka banyaknya chiperteks dihitung untuk menentukan matriks $M_{n\times n}$. Banyaknya chiperteks "mrreihevwsdemdbd" adalah 11 dan akar kuadrat yang mendekati 11 adalah $\sqrt{9}$ yaitu 2. 11 = $\sqrt{9}$ sisa 2, maka hasilnya adalah 3 sisa 2. Karena masih ada sisa bilangan maka akan dibulatkan keatas dan hasil dari pembulatan dari akar kuadrat adalah 4, jadi $n = 4$.

Selanjutnya chiperteks disusun menjadi matriks $M_{4\times4}$. Bentuk matriksnya adalah sebagai berkut:

$$M_{4\times4=}\begin{bmatrix} a & k & u & j \\ u & g & a & b \\ i & s & a & \$ \\ \$ & \$ & \$ & \$ \end{bmatrix}$$

Jika chiperteks sudah berbentuk matriks, maka untuk mengembalikan menjadi plainteks matriks tersebut dibaca menurut pola baca diagonal kiri ke kanan seperti berikut:

$$M_{4\times4} = \begin{bmatrix} \cancel{a} & \cancel{k} & \cancel{u} & \cancel{j} \\ \cancel{u} & \cancel{g} & \cancel{a} & \cancel{b} \\ \cancel{i} & \cancel{s} & \cancel{a} & \cancel{\$} \\ \cancel{\$} & \cancel{\$} & \cancel{\$} & \cancel{\$} \end{bmatrix}$$

Dengan menghilangkan karakter $ sebagai tambahan pada kolom matriks maka chiperteks akan kembali menjadi plainteks seperti semula yaitu “aku juga bisa”.

4.2 Pembahasan

Secara umum kebutuhan dari penggunaan kriptografi dalam pengiriman pesan adalah agar pesan yang dikirimkan menjadi aman dan tetap terjaga kerahasiaannya. Pada skripsi ini digunakan kriptografi klasik dan pada algoritma kriptografi klasik ini hanya digunakan dua teknik dasar yaitu *shift chiper* dan chiper transposisi. Hal tersebut bertujuan untuk mendapatkan pengkodean yang lebih kuat dari pada hanya menggunakan satu pengkodean saja, sehingga tidak mudah untuk dipecahkan. Enkripsi dan dekripsi dapat dilakukan dengan urutan *shift chiper* terlebih dahulu kemudian chiper transposisi. Dalam tinjauan pustaka telah disebutkan bahwa shift chiper adalah salah satu teknik enkripsi paling sederhana, Sandi ini termasuk sandi substitusi dimana setiap huruf pada teks terang (*plaintext*) digantikan oleh huruf lain yang memiliki selisih posisi tertentu dalam alfabet. Sandi geser (shift cipher) merupakan generalisasi dari Sandi Caesar, yaitu tidak membatasi pergeseran sebanyak tiga huruf sedangkan chiper transposisi adalah salah satu jenis teknik pengenkripsian pesan dengan cara mengubah urutan huruf-huruf yang ada di dalam plainteks menjadi chiperteks

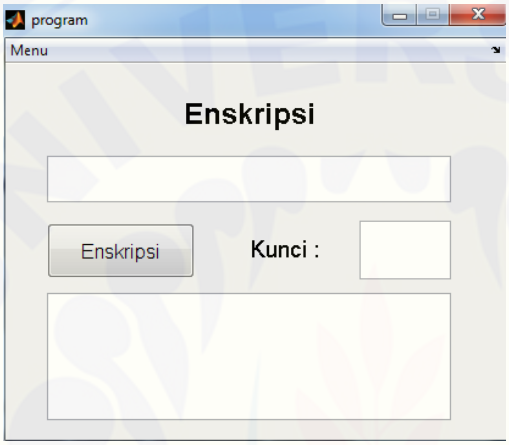
dengan cara tertentu agar isi dari pesan tersebut tidak dimengerti kecuali oleh orang-orang tertentu. Akan tetapi pada penyelesaian skripsi ini ditemukan masalah-masalah yang boleh dikatakan tidak sederhana dalam teori tersebut. Oleh karena itu dibuatlah algoritma penyelesaian pengkodean plainteks menggunakan gabungan *shift chiper* dan transposisi diagonal. Adapun algoritma tersebut berisi poin utama dari enkripsi dan dekripsi menggunakan gabunga *shift chiper* dan transposisi diagonal.

Walau dalam hal ini kriptosistem yang diselesaikan seolah sederhana namun dalam penyelesaiannya diperlukan adanya langkah-langkah pendukung terbentuknya kriptosistem ini. Hal pertama yang akan timbul bagi penerima pesan adalah kerancuan dari pembacaan pesan dan kunci yang digunakan. Dapat dicontohkan sebagai berikut, misalkan dikirimkan sebuah pesan "linda". Sebagaimana telah diketahui dalam penyelesaian kriptosistem dengan *shift chiper* yaitu dengan cara menggeser karakter-karakter yang ada di dalam plainteks menjadi karakter sesudahnya sesuai dengan kunci yang ditentukan, misal kunci yang digunakan adalah 2 maka plainteks "linda" menjadi nfk"cp"""". Setelah karakter plainteks digeser maka langkah selanjutnya adalah menghitung banyaknya pesan/plainteks, banyaknya plainteks nfk"cp"""" adalah 5. Akar kuadrat yang mendekati 5 adalah $\sqrt{4}$ sisa 1. Jika ada sisa bilangan maka dilakukan pembulatan ke atas jadi hasil $\sqrt{4}$ sisa 1 adalah 3, $n = 3$. Setelah n diketahui maka plainteks dibentuk matrik menjadi matriks $M_{n\times n}$ yakni matriks $M_{3\times 3}$. Maka bentuk matriksnya adalah sebagai berikut:

$$\mathrm{M}_{3\times 3} = \begin{bmatrix} \mathrm{n} & \mathrm{k} & \mathrm{p} \\ \mathrm{f} & \mathrm{c} & " \\ " & " & " \end{bmatrix}$$

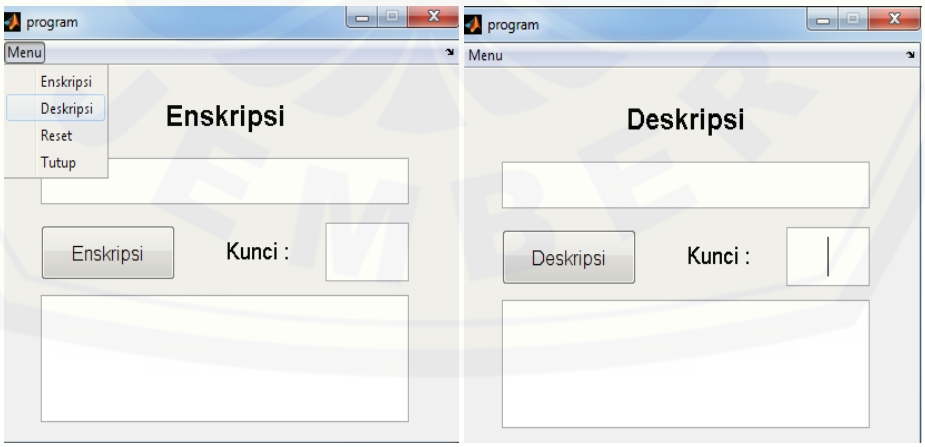
Dari matriks diatas bisa diketahui chiperteks dengan aturan pola baca kiri bawah ke kanan yaitu *nfk"cp""""*. Kerancuan penggunaan kunci dan pola baca yang digunakan disini dimaksudkan akan mempersulit dalam pemecahan pengkodean pesan. Untuk mengembalikan chiperteks menjadi plainteks kembali dilakukan proses yang sama yaitu mengubah chiperteks nfk"cp"""" menjadi linda dengan menggeser karakter-karakter chiperteks menjadi dua karakter sebelumnya

sesuai urutan nilai numeriknya. Dari algoritma yang dibuat telah dibuat juga program untuk pengkodean plainteks menggunakan gabungan *shift chipper* dan transposisi diagonal. Diharapkan dengan program ini semakin memudahkan kriptosistem menggunakan pengkodean plainteks menggunakan gabungan *shift chipper* dan transposisi diagonal. Gambar 4.3 berikut adalah jendela aplikasi untuk program pengkodean plainteks menggunakan gabungan *shift chipper* dan transposisi diagonal.



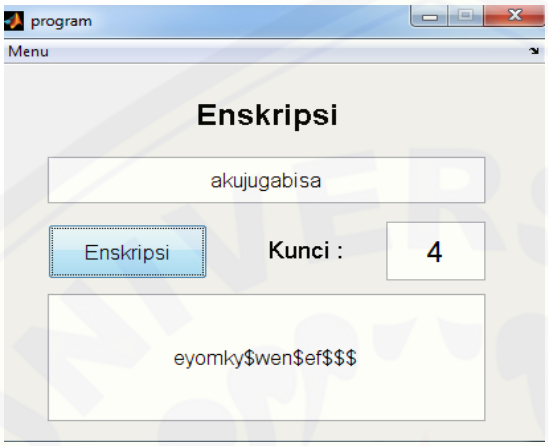
Gambar 4.3 Aplikasi Pengkodean Plainteks Menggunakan Gabungan *Shift Chiper* dan Transposisi Diagonal.

Gambar 4.3 di atas adalah jendela awal dari matlab yang merupakan langkah awal pada pengkodean yaitu enkripsi. Untuk menampilkan menu dekripsi maka tekan menu lalu pilih dekripsi seperti gambar berikut:



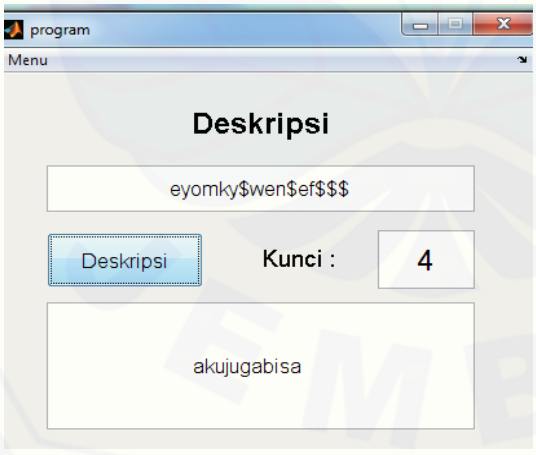
Gambar 4.4 Gambar Tampilan Menu Dekripsi

Setelah tampilan menu sudah keluar maka bisa dilanjutkan proses enkripsi dengan memasukkan plainteks ke dalam kolom enkripsi dan kunci yang digunakan ke dalam kolom kunci. Misal plainteksnya adalah aku juga bisa maka tampilan yang akan muncul sebagai berikut:



Gambar 4.5 Plainteks Menjadi Chiperteks

Setelah proses enkripsi selesai maka buka menu dekripsi untuk mengembalikan chiperteks menjadi plainteks, masukkan chiperteks ke dalam kolom dekripsi dan kunci yang sama pada saat proses enkripsi ke dalam kolom kunci. Tampilan menu dekripsi dapat dilihat pada gambar berikut:



Gambar 4.6 Chiperteks menjadi plainteks

Dalam pengiriman pesan, mula-mula pengirim pesan menuliskan pesan atau plainteks pada kolom yang telah disediakan sebagaimana pada gambar 4.3. Kemudian pesan tersebut dienkripsi oleh pengirim pesan sehingga terbentuk

chiperteks. Selanjutnya penerima pesan kemudian menginput chiperteks tersebut kedalam program untuk kemudian didekripsi sehingga diperoleh pesan yang dikirim oleh pengirim pesan.

BAB 5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian dari pengkodean plainteks menggunakan gabungan *shift chiper* dan transposisi diagonal maka dapat diambil kesimpulan sebagai berikut:

1. Pada pengkodean plainteks menggunakan gabungan *shift chiper* dan transposisi diagonal proses enkripsi dilakukan pertama menggunakan algoritma *shift chipper* terlebih dahulu lalu yang kedua menggunakan transposisi diagonal dengan pola baca kiri bawah ke kanan atas.
2. Pada proses dekripsi pertama juga dilakukan algoritma *shift chipper* terlebih dahulu kemudian algorita kedua transposisi diagonal dengan pola aa kiri ke kanan.
3. pengkodean plainteks menggunakan gabungan *shift chipper* dan transposisi diagonal lebih baik dari pada pengkodean plainteks yang menggunakan *shift chipper* saja atau pengkodean yang menggunakan transposisi diagonal saja.

5.2 Saran

Pengkodean dalam ilmu kriptografi masih luas dan banyak cara untuk mengembangkannya. Untuk penelitian selanjutnya bisa menggunakan teknik gabungan lebih dari dua chiper dan gabungan dari kriptografi klasik dan modern.

DAFTAR PUSTAKA

Ariyus, D. 2008. *Pengantar Ilmu Kriptografi Teori Analisis dan Impleentasi*. Yogyakarta: ANDI. [23 Januari 2015]

Andri, M.Y. 2009. *Implementasi Algoritma Kriptografi DES, RSA dan KompresiLZW pada Berkas Digital*. Medan: Universitas Sumatra Utara. [5 April 2014]

Asmara, F.D. 2009. Penyandian Hill menggunakan kode Plainteks Kembar. Skripsi. Tidak diterbitkan. Jember: Jurusan Matematika. Fakultas Ilmu Pengetahuan Alam. Universitas Jember.

Dian, Novi.M, dkk. *Penerapan Teknik Kriptografi Stream-Cipher untuk Pengaman Basis Data*. Jakarta Selatan: Program Studi Sistem Informasi FTKI. [5 April 2014]

Menezes Alfred J., Paul C. van Oorschot dan Scott A. Vanstone. 1996. *Handbook of Applied Cryptography*. CRC Press, USA.

Mukmin, I. Tanpa Tahun. *Modifikasi Playfair Chiper dengan Kombinasi Bifid, Caesar dan Tranpositional Chiper*. Skripsi. Tidak diterbitkan. Bandung: Jurusan Teknik Inforatika ITB. [26 Januari 2015]

Mulyono, E. 2012. *Sistem Pengkodean Pesan Teks Menggunakan Logika XOR dengan Satu Karakter Kunci*. Tidak Diterbitkan. Skripsi. Jember: FMIPA Universitas Jember

Munir, R. 2004. *Algoritma Kriptografi Klasik*. Bandung: Teknik Informatika ITB.

Pohan, YR. 2009. *Studi dan Perbandingan Berbagai Macam Algorita Chiper Transposisi*. Bandung: Tidak diterbitkan.Skripsi. Bandung: Jurusan Teknik Informatika ITB [26 Januari 2015]

Schneirer, B. 1996. Aplied Cryptography 2nd. New York: John Wiley & Sons

Wulandari, L. Tanpa Tahun. *Pengantar Komputer: Representasi Data*. Jakarta: Universitas Gunadarma. [23 Januari 2015]

Yahya, Yusuf dkk. 2005. *Matematika Dasar Untuk Perguruan Tinggi*. Ghalia Indonesia. [5 April 2014]

LAMPIRAN 1. Tabel Karakter ASCII

Desimal	Simbol	
0	NULL	(Null character)
1	SOH	(Start of Header)
2	STX	(Start of Text)
3	ETX	(End of Text)
4	EOT	(End of Transmission)
5	ENQ	(Enquiry)
6	ACK	(Acknowledgement)
7	BEL	(Bell)
8	BS	(Backspace)
9	HT	(Horizontal Tab)
10	LF	(Line feed)
11	VT	(Vertical Tab)
12	FF	(Form feed)
13	CR	(Carriage return)
14	SO	(Shift Out)
15	SI	(Shift In)
16	DLE	(Data link escape)
17	DC1	(Device control 1)
18	DC2	(Device control 2)
19	DC3	(Device control 3)
20	DC4	(Device control 4)
21	NAK	(Negative acknowledgement)
22	SYN	(Synchronous idle)
23	ETB	(End of transmission block)
24	CAN	(Cancel)
25	EM	(End of medium)
26	SUB	(Substitute)
27	ESC	(Escape)
28	FS	(File separator)
29	GS	(Group separator)

30	RS	(Record separator)
31	US	(Unit separator)

Desimal	Simbol	
32		(Space)
33	!	(Exclamation mark)
34	"	(Quotation mark ; quotes)
35	#	(Number sign)
36	$	(Dollar sign)
37	%	(Percent sign)
38	&	(Ampersand)
39	'	(Apostrophe)
40	(	(round brackets or parentheses)
41	)	(round brackets or parentheses)
42	*	(Asterisk)
43	+	(Plus sign)
44	,	(Comma)
45	-	(Hyphen)
46	.	(Dot , full stop)
47	/	(Slash)
48	0	(number zero)
49	1	(number one)
50	2	(number two)
51	3	(number three)
52	4	(number four)
53	5	(number five)
54	6	(number six)
55	7	(number seven)
56	8	(number eight)
57	9	(number nine)
58	:	(Colon)
59	;	(Semicolon)

60	<	(Less-than sign)
61	=	(Equals sign)
62	>	(Greater-than sign ; Inequality)
63	?	(Question mark)
64	@	(At sign)
65	A	(Capital A)
66	B	(Capital B)
67	C	(Capital C)
68	D	(Capital D)
69	E	(Capital E)
70	F	(Capital F)
71	G	(Capital G)
72	H	(Capital H)
73	I	(Capital I)
74	J	(Capital J)
75	K	(Capital K)
76	L	(Capital L)
77	M	(Capital M)
78	N	(Capital N)
79	O	(Capital O)
80	P	(Capital P)
81	Q	(Capital Q)
82	R	(Capital R)
83	S	(Capital S)
84	T	(Capital T)
85	U	(Capital U)
86	V	(Capital V)
87	W	(Capital W)
88	X	(Capital X)
89	Y	(Capital Y)
90	Z	(Capital Z)
91	[	(square brackets or box brackets)
92	\	(Backslash)

93	]	(square brackets or box brackets)
94	^	(Caret or circumflex accent)
95	_	(underscore , understrike , underbar or low line)
96	`	(Grave accent)
97	a	(Lowercase a)
98	b	(Lowercase b)
99	c	(Lowercase c)
100	d	(Lowercase d)
101	e	(Lowercase e)
102	f	(Lowercase f)
103	g	(Lowercase g)
104	h	(Lowercase h)
105	i	(Lowercase i)
106	j	(Lowercase j)
107	k	(Lowercase k)
108	l	(Lowercase l)
109	m	(Lowercase m)
110	n	(Lowercase n)
111	o	(Lowercase o)
112	p	(Lowercase p)
113	q	(Lowercase q)
114	r	(Lowercase r)
115	s	(Lowercase s)
116	t	(Lowercase t)
117	u	(Lowercase u)
118	v	(Lowercase v)
119	w	(Lowercase w)
120	x	(Lowercase x)
121	y	(Lowercase y)
122	z	(Lowercase z)
123	{	(curly brackets or braces)
124	|	(vertical-bar, vbar, vertical line or vertical slash)
125	}	(curly brackets or braces)

126	~	(Tilde ; swung dash)
127	DEL	(Delete)

Desimal	Simbol	
128	Ç	(Majuscule C-cedilla)
129	ü	(letter "u" with umlaut or diaeresis ; "u-umlaut")
130	é	(letter "e" with acute accent or "e-acute")
131	â	(letter "a" with circumflex accent or "a-circumflex")
132	ä	(letter "a" with umlaut or diaeresis ; "a-umlaut")
133	à	(letter "a" with grave accent)
134	å	(letter "a" with a ring)
135	ç	(Minuscule c-cedilla)
136	ê	(letter "e" with circumflex accent or "e-circumflex")
137	ë	(letter "e" with umlaut or diaeresis ; "e-umlaut")
138	è	(letter "e" with grave accent)
139	ï	(letter "i" with umlaut or diaeresis ; "i-umlaut")
140	î	(letter "i" with circumflex accent or "i-circumflex")
141	ì	(letter "i" with grave accent)
142	Ä	(letter "A" with umlaut or diaeresis ; "A-umlaut")
143	Å	(Capital letter "A" with a ring)
144	É	(Capital letter "E" with acute accent or "E-acute")
145	æ	(Latin diphthong "ae" in lowercase)
146	Æ	(Latin diphthong "AE" in uppercase)
147	ô	(letter "o" with circumflex accent or "o-circumflex")

148	ö	(letter "o" with umlaut or diaeresis ; "o-umlaut")
149	ò	(letter "o" with grave accent)
150	û	(letter "u" with circumflex accent or "u-circumflex")
151	ù	(letter "u" with grave accent)
152	ÿ	(Lowercase letter "y" with diaeresis)
153	Ö	(letter "O" with umlaut or diaeresis ; "O-umlaut")
154	Ü	(letter "U" with umlaut or diaeresis ; "U-umlaut")
155	ø	(slashed zero or empty set)
156	£	(Pound sign ; symbol for the pound sterling)
157	Ø	(slashed zero or empty set)
158	×	(multiplication sign)
159	*ƒ*	(function sign ; f with hook sign ; florin sign)
160	á	(letter "a" with acute accent or "a-acute")
161	í	(letter "i" with acute accent or "i-acute")
162	ó	(letter "o" with acute accent or "o-acute")
163	ú	(letter "u" with acute accent or "u-acute")
164	ñ	(letter "n" with tilde ; enye)
165	Ñ	(letter "N" with tilde ; enye)
166	ª	(feminine ordinal indicator)
167	º	(masculine ordinal indicator)
168	¿	(Inverted question marks)
169	®	(Registered trademark symbol)
170	¬	(Logical negation symbol)
171	½	(One half)
172	¼	(Quarter or one fourth)
173	¡	(Inverted exclamation marks)
174	«	(Angle quotes or guillemets)
175	»	(Guillemets or angle quotes)
176	░	
177	▒	
178	▓	
179	│	(Box drawing character)
180	┤	(Box drawing character)

181	Á	(Capital letter "A" with acute accent or "A-acute")
182	Â	(letter "A" with circumflex accent or "A-circumflex")
183	À	(letter "A" with grave accent)
184	©	(Copyright symbol)
185	╣	(Box drawing character)
186	║	(Box drawing character)
187	╗	(Box drawing character)
188	╝	(Box drawing character)
189	¢	(Cent symbol)
190	¥	(YEN and YUAN sign)
191	┐	(Box drawing character)
192	└	(Box drawing character)
193	┴	(Box drawing character)
194	┬	(Box drawing character)
195	├	(Box drawing character)
196	─	(Box drawing character)
197	┼	(Box drawing character)
198	ã	(Lowercase letter "a" with tilde or "a-tilde")
199	Ã	(Capital letter "A" with tilde or "A-tilde")
200	╚	(Box drawing character)
201	╔	(Box drawing character)
202	╩	(Box drawing character)
203	╦	(Box drawing character)
204	╠	(Box drawing character)
205	═	(Box drawing character)
206	╬	(Box drawing character)
207	¤	(generic currency sign)
208	ð	(Lowercase letter "eth")
209	Ð	(Capital letter "Eth")
210	Ê	(letter "E" with circumflex accent or "E-circumflex")
211	Ë	(letter "E" with umlaut or diaeresis ; "E-umlaut")
212	È	(letter "E" with grave accent)
213	ı	(lowercase dot less i)

214	Í	(Capital letter "I" with acute accent or "I-acute")
215	Î	(letter "I" with circumflex accent or "I-circumflex")
216	Ï	(letter "I" with umlaut or diaeresis ; "I-umlaut")
217	┘	(Box drawing character)
218	┌	(Box drawing character)
219	█	(Block)
220	▄	(Bottom half block)
221	¦	(vertical broken bar)
222	Ì	(letter "I" with grave accent)
223	▀	(Top half block)
224	Ó	(Capital letter "O" with acute accent or "O-acute")
225	ß	(letter "Eszett" ; "scharfes S" or "sharp S")
226	Ô	(letter "O" with circumflex accent or "O-circumflex")
227	Ò	(letter "O" with grave accent)
228	õ	(letter "o" with tilde or "o-tilde")
229	Õ	(letter "O" with tilde or "O-tilde")
230	µ	(Lowercase letter "Mu" ; micro sign or micron)
231	þ	(Lowercase letter "Thorn")
232	Þ	(Capital letter "thorn")
233	Ú	(Capital letter "U" with acute accent or "U-acute")
234	Û	(letter "U" with circumflex accent or "U-circumflex")
235	Ù	(letter "U" with grave accent)
236	ý	(Lowercase letter "y" with acute accent)
237	Ý	(Capital letter "Y" with acute accent)
238	¯	(macron symbol)
239	´	(Acute accent)
240		(Hyphen)
241	±	(Plus-minus sign)
242	_	(underline or underscore)
243	¾	(three quarters)
244	¶	(paragraph sign or pilcrow)
245	§	(Section sign)
246	÷	(The division sign ; Obelus)

247	¸	(cedilla)
248	°	(degree symbol)
249	¨	(Diaeresis)
250	·	(Interpunct or space dot)
251	¹	(superscript one)
252	³	(cube or superscript three)
253	²	(Square or superscript two)
254	■	(black square)
255	nbsp	(non-breaking space or no-break space)

LAMPIRAN 2. Program Pengkodean Plainteks Menggunakan Gabungan Shift Chiper dan Transposisi Diagonal

```
function varargout = program(varargin)
% PROGRAM M-file for program.fig
%      PROGRAM, by itself, creates a new PROGRAM or raises
the existing
%      singleton*.
%
%      H = PROGRAM returns the handle to a new PROGRAM or
the handle to
%      the existing singleton*.
%
%      PROGRAM('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in PROGRAM.M with the given
input arguments.
%
%      PROGRAM('Property','Value',...) creates a new PROGRAM
or raises the
%      existing singleton*.  Starting from the left,
property value pairs are
%      applied to the GUI before program_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes
property application
%      stop.  All inputs are passed to program_OpeningFcn
via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI
allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help program

% Last Modified by GUIDE v2.5 13-Jan-2015 09:40:25

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @program_OpeningFcn,
...
                   'gui_OutputFcn',  @program_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
```

```
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before program is made visible.
function program_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
% varargin   command line arguments to program (see
VARARGIN)

% Choose default command line output for program
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes program wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command
line.
function varargout = program_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function t1_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to t1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of t1 as
text
%        str2double(get(hObject,'String')) returns contents
of t1 as a double


% --- Executes during object creation, after setting all
properties.
function t1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to t1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes during object creation, after setting all
properties.
function t2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to t2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called



function kode_Callback(hObject, eventdata, handles)
% hObject    handle to kode (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of kode as
text
```

```
%        str2double(get(hObject,'String')) returns contents
of kode as a double


% --- Executes during object creation, after setting all
properties.
function kode_CreateFcn(hObject, eventdata, handles)
% hObject    handle to kode (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in e.
function e_Callback(hObject, eventdata, handles)
% hObject    handle to e (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

teks1 = get(handles.t1,'string');       % mengambil data
dari t1
pt = length(teks1);                     % menghitung panjang
teks1
if pt == 0                              % jika teks1 kosong,
maka ...
    set(handles.t2,'string','Tidak ada teks yang akan
dikodekan.');
end


n = ceil(sqrt(pt));                     % mencari akar
kuadrat dari panjang teks1 kemudian dibulatkan keatas (n)
pt2 = pt;
while pt2 ~= n^2                        % ketika panjang
teks1 tidak sama dengan n kuadrat
    teks1 = [teks1 ' '];                % menambahkan
karakter spasi pada akhir teks1
    pt2 = pt2+1;                        % perulangan terus
berlanjut hingga panjang teks 1 sama dengan n kuadrat
end
```

```
teks1_a=double(teks1);
sandi=get(handles.kode,'string');
sandi=str2num(sandi);
teks_sandi=teks1_a+sandi;
n1 = length(teks1);
for i=1:n1
    while (teks_sandi(i) > 126)
        teks_sandi(i)= (teks_sandi(i) - 126) + 32;
    end
end
teks1=char(teks_sandi);

A = [];                                 % membuat variabel
baru A
for i=1:n                               % untuk i=1 sampai n
maka akan dibuat sebuah matrik ukuran n x n dari teks1
    if i == 1
        A = [A;teks1(i:n)];
    else
        A = [A;teks1((i-1)*n+1:i*n)];
    end
end

% encoding
cteks = [];                             % membuat variabel
baru cteks
cteks = num2str(cteks);                 % mengubah format
variabel cteks dari numeric ke string
for i=1:n                               % membaca matriks
secara diagonal pada segitiga atas
    if i == 1
        cteks = [A(i,i)];
    else
        j=i;
        for k=1:i
            cteks = [cteks A(j,k)];
            j=j-1;
        end
    end
end

for i=2:n                               % membaca matriks
secara diagonal pada segitiga bawah
    if i ~= n
        j=i;
        for k=n:-1:i
            cteks = [cteks A(k,j)];
            j=j+1;
        end
    else
        cteks = [cteks A(n,n)];
```

```
    end
end

set(handles.t2,'string',cteks);         % memunculkan
ciperteks

% --- Executes on button press in d.
function d_Callback(hObject, eventdata, handles)
% hObject    handle to d (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
teks1 = get(handles.t1,'string');
pc = length(teks1);
if pc == 0
    set(handles.t2,'string','Tidak ada teks yang akan
dikodekan.');
end

n=ceil(sqrt(pc));
nc = get(handles.kode,'string');
sandi = str2num(nc);
pceksandi = length(sandi);


if pceksandi == 0
    set(handles.t2,'string','Sandi tidak boleh kosong atau
bukan angka.');
    set(handles.kode,'string','');
else

%     if ceksandi ~= n
%         set(handles.t2,'string','Sandi yang digunakan
salah.');
%         set(handles.kode,'string','');
%     else

while pc ~= n^2
            teks1 = [teks1 ' '];
            pc = pc+1;
end

teks_a=double(teks1);
sandi=get(handles.kode,'string');
sandi=str2num(sandi);
teks_sandi=teks_a-sandi;

n2 = length(teks1);
for i=1:n2
    while (teks_sandi(i) < 32)
```

```
            teks_sandi(i) = 126 - (32 - teks_sandi(i));
        end
    end
    teks1=char(teks_sandi);

            pteks2 = '';
            c = 1;
            for i=1:n
                if i == 1
                    pteks2(i,i) = teks1(c);
                    c = c+1;
                else
                    j=i;
                    for k=1:i
                        pteks2(j,k) = teks1(c);
                        j = j-1;
                        c = c+1;
                    end
                end
            end

            for i=2:n
                if i ~= n
                    j=i;
                    for k=n:-1:i
                        pteks2(k,j) = teks1(c);
                        j = j+1;
                        c = c+1;
                    end
                else
                    pteks2(n,n) = teks1(c);
                end
            end
            B = '';
            for i=1:n
                B = [B pteks2(i,:)];
            end

            set(handles.t2,'string',B);

    %     end
    end
    % --------------------------------------------------------------
    ----------
    function Untitled_1_Callback(hObject, eventdata, handles)
    % hObject    handle to Untitled_1 (see GCBO)
    % eventdata  reserved - to be defined in a future version of
    MATLAB
    % handles    structure with handles and user data (see
    GUIDATA)
```

```
% --------------------------------------------------------------------
function enskripsi_Callback(hObject, eventdata, handles)
% hObject    handle to enskripsi (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.d,'visible','off');
set(handles.e,'visible','on');
set(handles.kode,'string','');
set(handles.text2,'string','Enskripsi');

% --------------------------------------------------------------------
function deskripsi_Callback(hObject, eventdata, handles)
% hObject    handle to deskripsi (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.e,'visible','off');
set(handles.d,'visible','on');
set(handles.kode,'enable','on');
set(handles.kode,'string','');
set(handles.text2,'string','Deskripsi');


% --------------------------------------------------------------------
function tutup_Callback(hObject, eventdata, handles)
% hObject    handle to tutup (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close;

% --------------------------------------------------------------------
function resset_Callback(hObject, eventdata, handles)
% hObject    handle to resset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.t1,'string','');
set(handles.t2,'string','');
set(handles.kode,'string','')
```

```
function t2_Callback(hObject, eventdata, handles)
% hObject    handle to t2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of t2 as
text
%        str2double(get(hObject,'String')) returns contents
of t2 as a double
```