SANDI DATA MENGGUNAKAN KRIPTOGRAFI KLASIK *HILL CIPHER* YANG DIPERKUAT DENGAN *VIGENÈRE CIPHER*

Skripsi

Disusun Untuk Melengkapi Syarat-Syarat

Guna Memperoleh Sarjana Sains

KHOIRUZ ZAHRA

3125081768

PROGRAM STUDI MATEMATIKA

JURUSAN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS NEGERI JAKARTA

2015

LEMBAR PERSETUJUAN HASIL SIDANG SKRIPSI

SANDI DATA MENGGUNAKAN KRIPTOGRAFI KLASIK *HILL CIPHER* YANG DIPERKUAT DENGAN *VIGENÈRE CIPHER*

Nama : Khoiruz Zahra

No. Registrasi : 3125081768

		Nama	Tanda Tangan	Tanggal
Penanggung Jawab				
Dekan	:	Prof. Dr. Suyono, M.Si	.....	.....
		NIP. 19671218 199303 1 005		
Wakil Penanggung Jawab				
Pembantu Dekan I	:	Dr. Muktiningsih, M.Si	.....	.....
		NIP. 19640511 198903 2 001		
Ketua	:	Ir. Fariani Hermin, MT	.....	.....
		NIP. 19600211 198703 2 001		
Sekretaris	:	Ratna Widyati, S.Si., M.Kom	.....	.....
		NIP. 19750925 200212 2 002		
Penguji	:	Drs. Sudarwanto, M.Si., DEA	.....	.....
		NIP. 19650325 199303 1 003		
Pembimbing I	:	Drs. Mulyono, M.Kom	.....	.....
		NIP. 19660517 199403 1 003		
Pembimbing II	:	Ria Arafiyah, M.Si	.....	.....
		NIP. 19751121 200501 2 004		

Dinyatakan lulus ujian skripsi tanggal: 10 Juli 2015

ABSTRACT

KHOIRUZ ZAHRA, 3125081768. Code Data Using Classic Cryptography of *Hill Cipher* Reinforced by *Vigenère Cipher*. Thesis. Faculty of Mathematics and Natural Sciences. State University of Jakarta. 2015.

Cryptography is the study of mathematical techniques related to aspects of information security. Both Hill cipher and Vigenère cipher are parts of cryptography methode. In this thesis, discussed the modified method of Hill cipher which is reinforced by Vigenère cipher is the combination of both cipher. the plaintext is encrypted by Hill cipher using equation $C_i = \mathbf{KP_i} \pmod m$. *The encryption using Hill cipher results temporal ciphertext, which is continued then by Vigenère cipher encryption using equation* $C_i = K_i + P_i \pmod m$. *This will result ciphertext. Conversely, to get back the plaintext, the ciphertext is decrypted by vigenère cipher* $P_i = K_i - C_i \pmod m$ *in order to get the temporal plaintext. Subsequently, it is continued by Hill cipher decryption* $P_i = \mathbf{K}^{-1}.C_i \pmod m$ *to get the former plaintext.*

Keywords : *Hill cipher*, *encryption*, *decryption*, *Vigenere cipher*.

ABSTRAK

KHOIRUZ ZAHRA, 3125081768. Sandi Data Menggunakan Kriptografi Klasik *Hill Cipher* yang diperkuat dengan *Vigenère Cipher*. Skripsi. Fakultas Matematika dan Ilmu Pengetahuan Alam. Universitas Negeri Jakarta. 2015.

Kriptografi adalah studi teknik matematika yang berhubungan dengan aspek keamanan informasi. *Hill cipher* dan *Vigenère cipher* merupakan bagian dari metode kriptografi. Pada skripsi ini dibahas metode modifikasi *Hill cipher* yang diperkuat dengan *Vigenère cipher*. Proses enkripsi *plaintext* dengan *Hill cipher* menggunakan persamaan $C_i = \mathbf{K}.P_i \pmod{m}$. Enkripsi pada *Hill cipher* menghasilkan *ciphertext* sementara, kemudian dilanjutkan dengan enkripsi *Vigenère cipher* menggunakan persamaan $C_i = K_i + P_i \pmod{m}$, sehingga menghasilkan *ciphertext*. Proses dekripsi *ciphertext* dengan persamaan *Vigenère cipher* $P_i = K_i - C_i \pmod{m}$yang menghasilkan *plaintext* sementara, kemudian dilanjutkan dengan dekripsi *Hill cipher* $P_i = \mathbf{K}^{-1}.C_i \pmod{m}$ menghasilkan *plaintext* semula.

Kata kunci : *Hill cipher*, enkripsi, deskripsi, *Vigenère cipher*, *plaintext*, *ciphertext*.

PERSEMBAHANKU

"Sesungguhnya orang-orang yang mengatakan: "Tuhan kami ialah Allah" kemudian mereka meneguhkan pendirian mereka, maka malaikat-malaikat akan turun kepada mereka dengan mengatakan:"Janganlah kamu takut dan janganlah kamu bersedih hati; dan gembirakanlah mereka dengan jannah yang telah dijanjikan Allah kepadamu". Kamilah pelindung-pelindungmu dalam kehidupan dunia dan akhirat; di dalamnya kamu memperoleh apa yang kamu inginkan dan memperoleh (pula) di dalamnya apa yang kamu minta. Sebagai penghormatan (bagimu) dari (Allah) Yang Maha Pengampun lagi Maha Penyayang"
(QS. Fussilat : 30-32)

Antara 'Alim dan Faqih. Kita harus menjadi keduanya!

Skripsi ini aku persembahkan untuk kedua orangtuaku
tersayang, keluarga besarku, serta saudari dan sahabatku...

"Terima kasih atas semua dukungan yang menguatkan , do'a yang tak terputus,
serta kasih sayang yang terus mengalir...
Semoga Allah swt selalu Ridho terhadap aku dan kalian...
- Khoiruz Zahra".

KATA PENGANTAR

Alhamdulillahilladzi bini'matihi tatimmushshalihaat, puji syukur kepada Rabb semesta alam, Allah SWT, yang menciptakan kehidupan dan kematian untuk menguji siapa saja yang lebih baik amalnya, yang menggenggam setiap hati dan fikiran, yang maha mengetahui dan menciptakan pengetahuan, juga yang selalu melimpahkan rahmaan dan rahiim-Nya, sehingga penulis bisa menyelesaikan skripsi ini dengan judul "Sandi Data menggunakan Kriptografi Klasik *Hill Cipher* yang Diperkuat dengan *Vigenère Cipher*" yang merupakan salah satu syarat dalam memperoleh gelar Sarjana Sains Jurusan Matematika Universitas Negeri Jakarta.

Skripsi ini berhasil diselesaikan tidak terlepas dari adanya bantuan dari berbagai pihak. Oleh karena itu, dalam kesempatan ini penulis ingin menyampaikan terima kasih terutama kepada:

1. Orang tua tersayang (alm) Bapak Rachmat dan Emak Siti Warti yang selalu memberikan semangat kepada penulis dalam menyusun skripsi. Mohon maaf dan terimakasih atas kesabarannya menunggu kelulusan Zahra.

2. Seluruh keluarga penulis yang selalu mendukung penulis dalam menyusun skripsi. Bang Arip & Mba Lia, Bang Iwan & Mba Nur, Bang Aman & Mba Afifah, Po Lely, Bang Abi, Khoir, Hafidz, Syamsul, Fahmi, Zaki. Juga bocil-bocil keponakan, Kakak Rifdah & Dedek Erdogan, Kakak Saffanah & Abang Hudzaifah, Dedek Nisa. Kalian memberi warna tersendiri buat Zahra.

3. Bapak Drs. Mulyono, M.Kom. dan Ibu Ria Arafiyah, M.Si sebagai dosen

pembimbing yang telah meluangkan waktunya untuk memberikan bimbingan, saran, nasehat serta pengarahannya sehingga skripsi ini dapat menjadi lebih baik.

4. Bapak Drs. Makmuri, M.Si, selaku Ketua Jurusan Matematika FMIPA UNJ dan Ibu Ratna Widyati, S.Si., M.Kom selaku Ketua Prodi Matematika FMIPA UNJ.

5. Seluruh Bapak/Ibu dosen atas pengajarannya yang telah diberikan dan karyawan-karyawati FMIPA UNJ yang telah memberikan informasi yang penulis butuhkan dalam menyelesaikan skripsi.

6. Bapak M. Eka Suryana, M.Kom serta seluruh aslab yang membantu dalam menyelesaikan program.

7. Teman-teman Matematika 2008, Riska, Pipit, Parti, Wieke, Dinar, Indah, Ern, Ana, Nia, Fara, Pewe dan semua sahabat penulis yang telah memberikan dukungan kepada penulis selama menyusun skripsi. 'Tanpa mu apa jadinya aku' :).

8. Mbak Ummi dan seluruh saudari di grup kecil TS dan HaiHai, juga si Bunda. Jazakumullah Khairan Katsiran untuk semangat dan bantuannya.

Penulis menyadari bahwa masih terdapat banyak kekurangan pada skripsi ini dikarenakan pengetahuan penulis yang terbatas. Penulis berharap skripsi ini dapat bermanfaat bagi para pembaca.

Jakarta, Juli 2015

Khoiruz Zahra

DAFTAR ISI

DAFTAR GAMBAR

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Teknologi Informasi dan komunikasi saat ini mengalami perkembangan yang pesat. Perkembangan teknologi informasi dapat ditandai oleh komputer yang canggih, ketersedian jaringan internet yang mudah diakses, hingga banyaknya *gadget* yang berbasis komputer. Perkembangan teknologi informasi mempermudah seseorang dalam pekerjaan, penyampaian pesan serta mengakses informasi.

Kemajuan teknologi informasi dan komunikasi mambantu mempercepat proses pengiriman dan penerimaan pesan dan informasi. Ketersediaan jaringan internet dapat mengefisiensikan kendala jarak dan waktu dalam pengiriman dan penerimaan pesan serta informasi berupa teks, gambar, suara dan video. Akan tetapi, kemajuan teknologi informasi juga memiliki beberapa kelemahan. Banyaknya pertukaran informasi yang terjadi setiap detik, memungkinkan adanya pihak-pihak yang tidak berhak mendapatkan informasi melakukan pencurian informasi atau penyadapan, sehingga maksud dari informasi yang dikirim akan dengan mudah diketahui. Dalam bidang militer, ekonomi, atau kebijakan-kebijakan penting lainnya, hal ini akan berpengaruh terhadap keamanan dan langkah-langkah strategis yang akan diambil berikutnya.

Beberapa cara untuk menangani masalah keamanan sistem komunikasi

mulai ditemukan dan dikembangkan. Salah satu cara yang dikenal sejak dahulu dalam mengamankan pesan atau informasi adalah teknik kriptografi. Menurut Alfred J. Mezes, Paul C Van Oorschot dan Scott A. Vanstone (1996:4), kriptografi adalah studi teknik matematika yang dipercaya untuk menangani masalah keamanan pesan atau informasi, karena disamping menggunakan bahasa pemrograman komputer, kriptografi menggunakan rumus dan fungsi pada matematika, mulai dari rumus yang sederhana sampai dengan rumus yang kompleks.

Secara umum, kriptografi dibedakan menjadi dua, kriptografi klasik dan kriptografi modern. Kriptografi klasik merupakan kriptografi satu kunci, yaitu untuk melakukan enkripsi dan dekripsi menggunakan kunci yang sama. Disebut juga sebagai kriptografi simetri atau kriptografi kunci rahasia. Kriptografi modern disebut sebagai kriptografi dua kunci, kriptografi asimetri atau kriptografi kunci publik, yaitu untuk melakukan enkripsi dan dekripsi menggunakan kunci yang berbeda. Algoritma kriptografi klasik diantaranya adalah *Hill cipher* dan *Vigenère cipher*.

Kriptografi *Hill cipher* ditemukan pertama kali oleh Lester S. Hill (1929) dalam tulisannya *Cryptography in an algebraic alphabet*. Karyanya banyak dikembangkan dan dimodifikasi oleh peneliti diantaranya oleh M. Nordin A. Rahman *et al.* (2013) meneliti kombinasi dari *Affine Cipher* dan *Hill cipher* dan dihasilkan penyandian baru Hill++. Niken (2010) menulis tentang kriptografi *Hill cipher* dengan menggunakan operasi matriks yang membahas penggunakaan matriks pada kriptografi. Pada tulisan tersebut menggunakan matriks ukuran 2x2 dengan determinan matriks bernilai 1 atau -1 serta memanfaatkan kode ASCII dalam pengkonversian simbol. Juliadi *et al.* (2013) meneliti kriptografi klasik dengan metode modifikasi *affine cipher* yang diperkuat dengan *Vigenère cipher*. Dalam penelitiannya dihasilkan algoritma kriptografi menggunakan dua kunci.

Pada penelitian ini akan dibahas mengenai penggabungan metode *Hill cipher* yang diperkuat dengan *Vigenère cipher* yang akan memberikan penyandian baru. Merujuk pada jurnal ilmiah *A New Approach of Classical Hill Cipher* dan Kriptografi Klasik dengan Metode Modifikasi *affine Cipher* yang diperkuat dengan *Vigenère* Cipher. Pada tulisan ini ditiadakan syarat determinan harus 1 dan -1, sehingga memudahkan pengguna metode ini dalam pengambilan kunci acak *Hill cipher* yang berupa matriks. Pada Skripsi ini metode *Hill cipher* yang diperkuat *Vigenère cipher* dilakukan perhitungan secara manual dan menggunakan program MATLAB untuk membantu pengirim dan penerima menjalankan proses enkripsi dan dekripsi teks.

1.2 Perumusan Masalah

Berdasarkan latar belakang masalah, maka pokok permasalahan yang dapat diuraikan adalah sebagai berikut : Bagaimanakah algoritma dari penyandian data teks dengan menggunakan *Hill cipher* yang diperkuat dengan *Vigenère cipher*?

1.3 Pembatasan Masalah

Penggunaan kriptografi *Hill cipher* yang diperkuat dengan *Vigenère cipher* dibatasi oleh kunci *Hill cipher* berupa matriks 3x3.

1.4 Tujuan Penulisan

Tujuan yang ingin dicapai dalam penulisan ini adalah: Menentukan algoritma dari penyandian data teks dengan menggunakan *Hill cipher* yang diperkuat dengan *Vigenère cipher* untuk meningkatkan keamanan pesan dan informasi.

1.5 Manfaat Penulisan

1. Mengaplikasikan serta memperdalam pengetahuan dan teori tentang kriptografi.

2. Sebagai referensi dan informasi tambahan untuk melakukan penelitian dan kajian lebih lanjut mengenai metode dalam kriptografi klasik khususnya *Hill cipher* dan *Vigenère cipher*.

1.6 Metode Penelitian

Metode penelitian yang digunakan dalam penelitian ini adalah studi pustaka, didasarkan pada buku-buku dan jurnal ilmiah yang berhubungan dengan kriptografi dan aturan matematika yang berkaitan dengan *Hill cipher* dan *Vigenère cipher*.

BAB II

LANDASAN TEORI

2.1 Kriptografi

Kriptografi *(cryptography)* berasal dari bahasa Yunani, terdiri dari dua suku kata yaitu kripto dan graphia. Kripto artinya *secret* (menyembunyikan), sedangkan graphia artinya *writing* (tulisan) (Dony Ariyus, 2006:9). Menurut Alfred Menezes et.al(1996:4), *"Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, and data origin authentication."*

Kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi, seperti kerahasiaan data, keabsahan data, integritas data, serta autentikasi data. Akan tetapi tidak semua aspek keamanan informasi dapat diselesaikan dengan kriptografi. Kriptografi dapat pula diartikan sebagai ilmu atau seni untuk menjaga keamanan pesan.

Sentot Kromodimoeljo (2010:1) dalam bukunya Teori dan Aplikasi Kriptografi mengatakan bahwa pada zaman Romawi Kuno, *Julius Caesar* telah menggunakan teknik kriptografi yang dijuluki *Caesar cipher* untuk mengirim pesan secara rahasia. *Casanova* menggunakan pengetahuan mengenai kriptografi untuk mengelabui *Madame d'Urfe*. Ia mengatakan kepada *Madame d'Urfe* bahwa sesosok jin memberi tahu kunci rahasia *Madame d'Urfe* kepadanya, padahal ia berhasil memecahkan kunci rahasia berdasarkan pengetahuannya mengenai krip-

tografi, sehingga ia mampu mengontrol kehidupan *Madame d'Urfe* secara total. Sewaktu perang dunia kedua, pihak sekutu berhasil memecahkan kode mesin kriptografi Jerman, Enigma, keberhasilan yang sangat membantu pihak sekutu dalam memenangkan perang. Sejarah kriptografi penuh dengan intrik dan banyak orang melihat kriptografi sebagai sesuatu yang penuh dengan misteri.

Terdapat empat tujuan mendasar dari ilmu kriptografi yang juga merupakan beberapa aspek keamanan informasi yaitu:

1. Kerahasiaan *(Confidentiality / Privacy)*, adalah aspek yang berhubungan dengan penjagaan isi informasi dari siapapun kecuali yang memiliki otoritas atau kunci rahasia untuk membuka / mengupas informasi yang telah dienkripsi.

2. Integritas data *(Data Integrity)*, adalah aspek yang 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.

3. Autentikasi *(Authentication)*, adalah aspek yang berhubungan dengan identifikasi atau pengenalan, baik secara kesatuan sistem maupun informasi itu sendiri. Dua pihak yang saling berkomunikasi harus saling memperkenalkan diri, waktu pengiriman, dan lain-lain.

4. Menolak Penyangkalan *(Non-repudiasi)*, adalah suatu usaha untuk mencegah terjadinya penyangkalan terhadap pengiriman suatu informasi oleh yang mengirim/membuat, atau harus dapat membuktikan bahwa suatu pesan berasal dari seseorang, apabila ia menyangkal informasi tersebut.

2.1.1 Algoritma Kriptografi

Algoritma adalah urutan langkah-langkah logis untuk menyelesaikan suatu masalah yang disusun secara sistematis (Munir (2001:4)). Langkah-langkah tersebut harus logis, ini berarti nilai kebenarannya harus dapat ditentukan benar atau salah. Algoritma kriptografi adalah bagian dari kriptografi yang berisi kumpulan langkah-langkah logis yang berfungsi untuk melakukan tujuan dari ilmu kriptografi itu sendiri. Biasanya langkah-langkah ini berupa sekumpulan fungsi matematika. Algoritma kriptografi terdiri dari 2 bagian fungsi, yaitu enkripsi dan dekripsi.

Shannon (1949:53) mengatakan bahwa Algoritma kriptografi harus memiliki kekuatan untuk melakukan konfusi dan difusi. Konfusi adalah mengaburkan hubungan antara *plaintext* dan *ciphertext*. Cara paling mudah untuk melakukan konfusi adalah menggunakan substitusi. Konfusi menimbulkan kesulitan dalam usaha musuh untuk mencari keteraturan dan pola statistik antara *plaintext* dan *ciphertext*. Difusi adalah menyebarkan redudansi *plaintext* dengan menyebarkan masukan ke seluruh *ciphertext*. Cara yang paling mudah untuk dapat melakukan difusi adalah dengan menggunakan metode transposisi. Jika menggunakan difusi, akan dibutukan waktu yang lebih lama untuk memecakan sandi rahasia ini. Sehingga dapat digunakan untuk mengamankan informasi.

Pada implementasinya sebuah algoritma sandi harus memperhatikan kualitas layanan dari keseluruhan sistem dimana sandi diimplementasikan. Algoritma sandi yang handal adalah algoritma sandi yang kekuatannya terletak pada kunci, bukan pada kerahasiaan algoritma itu sendiri.

2.1.2 Macam-Macam Kriptografi

William Stallings (2011:35) dalam bukunya *Cryptography and Network Security, Principles and Practice 5th edition* membagi macam-macam kriptografi berdasarkan tiga kriteria yaitu, jenis operasi yang digunakan untuk mengubah *plaintext* ke *ciphertext*, jumlah kunci yang digunakan, dan cara memproses *plaintext*.

1. Jenis operasi yang digunakan untuk mengubah *plaintext* ke *ciphertext*
   (a) Substitusi, setiap elemen *plaintext* (bit, huruf, kelompok bit atau huruf) dipetakan ke elemen lain.
   (b) transposisi, yaitu unsur-unsur dalam *plaintext* disusun kembali secara acak. Persyaratan mendasar adalah bahwa tidak ada informasi yang hilang (semua operasi reversibel).
2. Jumlah kunci yang digunakan
   (a) Jika kedua pengirim dan penerima menggunakan kunci yang sama, Sistem ini disebut sebagai simetris, satu kunci, *secret-key* (kunci rahasia), atau enkripsi konvensional.
   (b) Jika pengirim dan penerima menggunakan kunci yang berbeda, sistem ini disebut sebagai asimetris, dua kunci, atau enkripsi kunci publik.
3. Cara memproses *plaintext*
   (a) Blok cipher yaitu dengan memasukan satu blok elemen pada suatu waktu, menghasilkan sebuah blok output untuk masing-masing masukan blok.

(b) Stream cipher memproses elemen masukan terus menerus, menghasilkan output satu elemen pada suatu waktu.

Sedangkan secara garis besar, algoritma kriptografi dapat dibedakan menjadi dua yaitu kriptografi klasik dan kriptografi modern. Kriptografi klasik yaitu kriptografi yang digunakan pada zaman dahulu, bentuknya sederhana belum berbasis komputer dan tidak begitu rumit. Sedangkan Doni Ariyus (2006:16) menuliskan kriptografi modern yaitu kriptografi yang digunakan pada zaman kekinian (komputer sudah ditemukan), biasanya disertai dengan perhitungan yang rumit dan kompleks, dan dalam pengoperasiannya menggunakan komputer.

2.2 Enkripsi dan Dekripsi

Menurut William Stallings (2011:33) dalam bukunya *Cryptography and Network Security Principles and Practice 5th edition*, Kriptografi simetri memiliki 5 komponen utama yaitu:

1. *Plaintext* (pesan asli), yaitu pesan yang dapat langsung dibaca dan mudah dimengerti.

2. *Ciphertext* (pesan tersandi), yaitu pesan acak yang telah disandikan sehingga tidak bermakna lagi dan tidak dapat dibaca oleh pihak lain yang tidak berkepentingan.

3. *Key* (kunci), yaitu kunci untuk melakukan teknik kriptografi.

4. *Encryption Algorithm* (algoritma enkripsi), yaitu algoritma untuk mengubah *plaintext* menjadi *ciphertext*.

5. *Decryption Algorithm* (algoritma dekripsi), kebalikan dari algoritma enkripsi, yaitu algoritma untuk mengubah *ciphertext* menjadi *plaintext*.

Teknik pengamanan pesan dengan metode kriptografi harus melalui proses enkripsi dan dekripsi. Untuk melakukan proses enkripsi dan dekripsi diperlukan kunci *(key)* dan algoritma *(algorithm)*. Konsep matematis yang mendasari algoritma kriptografi adalah relasi antara dua himpunan yaitu himpunan yang berisi elemen-elemen pesan asli *(plaintext)* dan himpunan yang berisi pesan tersandi (*ciphertext*). Enkripsi dan dekripsi merupakan fungsi yang memetakan elemen-elemen antara kedua himpunan tersebut. Misalkan P menyatakan *plaintext* dan C menyatakan *ciphertext*, maka fungsi enkripsi E memetakan P ke C.

Definisi 2.2.1. (Enkripsi)
Enkripsi (E) adalah proses yang dilakukan untuk mengubah *plaintext* (pesan asli) P kedalam *ciphertext* (pesan tersandi) C.

$$E_k(P) = C \tag{2.1}$$

Definisi 2.2.2. (Dekripsi)
Dekripsi (D) adalah proses yang dilakukan untuk mengubah *ciphertext* (pesan tersandi) C kedalam *plaintext* (pesan asli) P.

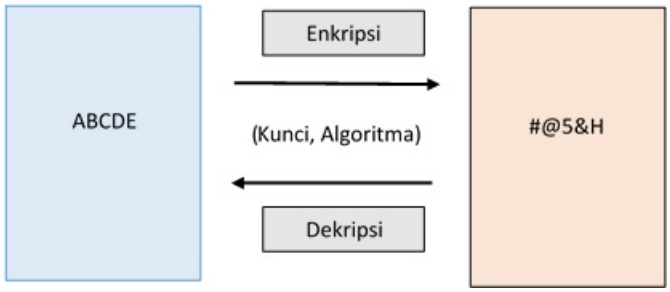
$$D_K(C) = P \tag{2.2}$$

Secara matematika, kriptografi adalah pasangan himpunan *plaintext* P, himpunan *ciphertext* C, himpunan kunci K, fungi enkripsi E, fungsi dekripsi D, dimana $\forall k \in K$ terdapat fungsi $e \in E$, $e_k(x) = y$ merupakan *chipertext* dalam C. Sehingga $d_k(y) = x$ untuk fungsi $d \in D$.

Karena proses enkripsi kemudian dekripsi mengembalikan pesan tersandi ke pesan asal, maka kesamaan berikut harus benar,

$$D(E(P)) = P \tag{2.3}$$

Berikut adalah ilustrasi 5 komponen dan 2 proses yang digunakan dalam teknik kriptografi.



Gambar 2.1: proses enkripsi dan dekripsi

Teknik pengamanan pesan yang baik yaitu dengan menggunakan suatu proses enkripsi yang menghasilkan naskah acak yang memerlukan waktu yang lama untuk didekripsi oleh seseorang atau pihak lain yang tidak mempunyai kunci dekripsi. Serta tentunya naskah acak harus dapat didekripsi oleh seseorang yang mempunyai kunci dekripsi untuk mendapatkan kembali naskah asli. Tingkat keamanan berbanding lurus dengan proses pengamanan. Untuk mendapatkan pesan yang aman maka akan semakin banyak proses yang harus dikerjakan dalam pengamanan pesan. Walaupun memerlukan beberapa proses, pesan yang dienkrip harus dapat didekrip kembali oleh penerima pesan, akan tetapi proses dekripsi harus lebih sulit jika dilakukan oleh kriptanalis.

2.3 Matriks

Sebuah matriks adalah susunan segi empat siku-siku dari bilangan-bilangan. Bilangan-bilangan dalam susunan tersebut dinamankan entri matriks. Matriks merupakan kumpulan koefisien dari sistem persamaan yang dituliskan menjadi entri-entri pada matriks. Disebut matriks yang diperbesar jika entri-entri matriks terdiri atas koefisien-koefisien dan konstanta dari sistem persamaan tersebut.

Definisi 2.3.1. (Matriks)
Matriks **A** berukuran $m \times n$ ialah suatu susunan angka dalam persegi empat ukuran $m \times n$, sebagai berikut:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \tag{2.4}$$

matriks berukuran (ordo) m x n atau $\mathbf{A} = (a_{ij})$.

Matriks ukuran $n \times n$ disebut sebagai matriks persegi. Selain ditulis dengan [ ], bentuk siku empat juga sering ditulis dengan ( ). Matriks yang terdiri dari 1 baris dan n kolom ditulis $1 \times n$ disebut dengan matriks baris atau vektor baris dan yang terdiri dari n baris dan 1 kolom disebut matriks kolom atau vektor kolom.

2.3.1 Eliminasi Gauss dan Eliminasi Gauss-Jordan

Definisi 2.3.2. Operasi berikut disebut sebagai operasi baris elementer, antara lain:

1. Pertukaran dua baris.

2. Perkalian suatu baris dengan skalar tak nol.

3. Penjumlahan baris yang dikalikan dengan skalar tak nol dengan baris lain.

Definisi 2.3.3. Suatu matriks dikatakan dalam bentuk eselon baris (*ref*) bila memenuhi hal-hal berikut:

1. Jika suatu baris tidak terdapat entri nol, maka entri tak nol pertama baris tersebut adalah 1 (entri 1 ini disebut sebagai leading entry)

2. Jika terdapat baris yang semua entrinya nol maka baris tersebut diletakkan di bagian bawah matriks

3. Setiap leading entri 1 terletak disebelah kanan leading entri 1 yang terletak di bagian atas

Definisi 2.3.4. Eselon baris dikatakan sebagai eselon baris tereduksi (*rref*) jika memenuhi kondisi eselon baris dengan setiap kolom terdiri atas leading entri 1 dan nol untuk entri yang lain.

2.3.2 Operasi Matriks

Definisi 2.3.5. (Perkalian Matriks)
Jika $\mathbf{A}$ adalah matriks $m \times r$ dan $\mathbf{B}$ adalah matriks $r \times n$ maka hasilkali $\mathbf{AB}$ adalah matriks $m \times n$ yang entri-entrinya ditentukan sebagai berikut. Untuk mencari entri pada baris i dan kolom j dari $\mathbf{AB}$, pilihlah baris i dari matriks $\mathbf{A}$ dan kolom j dari matriks $\mathbf{B}$. Kalikan entri-entri yang bersesuaian dari baris dan kolom tersebut dan kemudian jumlahkan hasil yang diperoleh.

Definisi perkalian matriks mengharuskan bahwa banyaknya kolom dari faktor pertama **A** harus sama seperti banyaknya baris dari faktor kedua **B** agar membentuk hasil kali **AB**. Jika kondisi ini tidak dipenuhi, maka hasil kali tersebut tidak dapat didefinisikan.

Definisi 2.3.6. (Transpos Matriks)
Jika **A** adalah matriks $m \times n$, maka transpos dari **A** dinyatakan oleh $\mathbf{A}^T$, didefinisikan sebagai matriks $n \times m$ yang didapatkan dengan mempertukarkan entri-entri baris dan entri-entri kolom dari **A**; sehingga entri-entri kolom pertama dari $\mathbf{A}^T$ adalah entri-entri baris pertama dari **A**, entri-entri kolom kedua dari $\mathbf{A}^T$ adalah entri-entri baris kedua dari **A**, dan seterusnya.

Contoh 2.3.1. Transpos matriks $\mathbf{A} = \begin{bmatrix} 3 & -1 \\ -5 & 10 \\ 8 & 7 \end{bmatrix}$, adalah $\mathbf{A}^T = \begin{bmatrix} 3 & -5 & 8 \\ -1 & 10 & 7 \end{bmatrix}$

Perhatikan bahwa tidak hanya entri-entri kolom $\mathbf{A}^T$ yang merupakan entri-entri baris **A**, tetapi juga entri-entri baris dari $\mathbf{A}^T$ adalah entri-entri kolom dari **A**. Jadi, entri pada baris i dan kolom j pada $\mathbf{A}^T$ adalah entri pada baris j dan kolom i pada **A**, sehingga:

$$\mathbf{A}^T = \mathbf{A}_{ji} \tag{2.5}$$

Pada kasus khusus dimana **A** adalah matriks persegi, transpos dari **A** dapat diperoleh dengan saling mempertukarkan entri-entri yang posisinya simetris terhadap diagonal utama.

2.3.3 Determinan Matriks

Definisi 2.3.7. (Hasil Kali Elementer)
Suatu hasil kali elementer dari suatu matriks persegi $\mathbf{A}_{n\times n}$, adalah hasil kali n

entri dari **A**, yang tidak satu pun berasal dari baris atau kolom yang sama.

Definisi 2.3.8. (Determinan)
Misalkan **A** adalah suatu matriks persegi. Fungsi determinan dinyatakan oleh **det**, dan didefinisikan det(A) sebagai jumlah semua hasil kali elementer bertanda dari **A**. Jumlah det(A) dinamakan sebagai determinan A.

Teorema 2.3.1. Jika **A** adalah sebarang matriks persegi yang mengandung sebaris bilangan nol, maka det(A) = 0

Bukti:
Karena hasil kali elementer bertanda dari **A** mengandung satu faktor dari setiap baris **A**, maka tiap-tiap hasil kali elementer bertanda mengandung faktor dari baris bilangan nol dan sebagai konsekuensinya juga akan mempunyai nilai nol. Karena det(A) adalah jumlah semua hasil kali elementer bertanda, maka kita dapatkan det(A) = 0.

Definisi 2.3.9. (Minor - Kofaktor)
Jika **A** adalah suatu matriks persegi, maka minor dari entri a_{ij} dinyatakan sebagai M_{ij} dan didefinisikan sebagai determinan dari submatriks yang tersisa setelah baris ke-i dan kolom ke-j dihilangkan dari **A**. Bilangan $(-1)^{i+j}\mathbf{M}_{ij}$ dinyatakan sebagai $\mathbf{C}_{ij}$ dan disebut sebagai kofaktor dari entri a_{ij}.

Contoh 2.3.2. Menghitung Minor dan Kofaktor matriks $\mathbf{A} = \begin{bmatrix} 3 & 1 & -4 \\ 2 & 5 & 6 \\ 1 & 4 & 8 \end{bmatrix}$

Minor dari entri a_{11} adalah

$$\mathbf{M_{11}} = \begin{bmatrix} 5 & 6 \\ 4 & 8 \end{bmatrix} = 16$$

Kofaktor dari a_{11} adalah

$$C_{11} = (-1)^{1+1} M_{11} = M_{11} = 16$$

Menghitung Determinan Matriks

Untuk menghitung determinan dari suatu matriks, dengan menggunakan perluasan kofaktor, pandanglah suatu unsur a_{ij} dari matriks **A** berukuran $n \times n$ Jika **A** matriks persegi dengan $n > 2$, maka:

1. Pengembangan determinan menurut kolom ke-j.

$$\begin{aligned} det(A) &= a_{1j}C_{1j} + a_{2j}C_{2j} + \ldots + a_{nj}C_{nj} \\ &= \sum_{i=1}^{n} a_{ij}C_{ij} \text{untuk j tetap} 1 \leq j \leq n \end{aligned}$$

2. Pengembangan determinan menurut baris ke-i.

$$\begin{aligned} det(A) &= a_{i1}C_{i1} + a_{i2}C_{i2} + \ldots + a_{in}C_{in} \\ &= \sum_{i=1}^{n} a_{ij}C_{ij} \text{untuk i tetap} 1 \leq j \leq n \end{aligned}$$

Contoh":

Hitunglah determinan dari matriks: $\mathbf{A} = \begin{bmatrix} 2 & 3 & 1 \\ 1 & 2 & 3 \\ 3 & 1 & 2 \end{bmatrix}$

Dengan menggunakan penguraian menurut baris ke 1 diperoleh:

$$
\begin{aligned}
det(A) &= \begin{vmatrix} 2 & 3 & 1 \\ 1 & 2 & 3 \\ 3 & 1 & 2 \end{vmatrix} \\
&= (-1)^{1+1}2\begin{vmatrix} 2 & 3 \\ 1 & 2 \end{vmatrix} + (-1)^{1+2}3\begin{vmatrix} 1 & 3 \\ 3 & 2 \end{vmatrix} + (-1)^{1+3}3\begin{vmatrix} 1 & 2 \\ 3 & 1 \end{vmatrix} \\
&= 2(4-3) - 3(2-9) + (1-6) \\
&= 18
\end{aligned}
$$

2.3.4 Invers Matriks

Definisi 2.3.10. (Invers Matriks)

Jika **A** adalah matriks persegi, dan jika terdapat matriks **B** yang ukurannya sama sedemikian sehingga $\mathbf{AB} = \mathbf{BA} = \mathbf{I}$, maka **A** disebut dapat dibalik (invertible) dan **B** disebut sebagai invers dari **A**.

Teorema 2.3.2. Jika **A** dan **B** adalah matriks persegi yang ukurannya sama, maka det(AB)=det(A)det(B).

Bukti:

Asumsikan satu dari matriks **A** atau **B** mempunyai det=0, berakibat det(A)det(B)=0. Jika det(B)=0 maka **B**x=0, untuk $x \neq 0$. Persamaan ini mempunyai tak hingga banyaknya soluusi. Kalikan **B**X=0 dengan matriks **A** diruas kiri sehingga **AB**x=0 menunjukkan bahwa perkalian matriks **AB** tidak invertible. Oleh karena itu dipenuhi det(AB)=det(A)det(B). Jika $det(B) \neq 0$ dan det(A)=0, maka ada suatu vektor $y \neq 0$ memenuhi persamaan **A**y=0. Ambil $x = \mathbf{B}^{-1}y$ maka $\mathbf{AB}x = \mathbf{A}y$. Karena $\mathbf{A}y = 0$ berarti perkalian matriks **AB** tidak invertible. Asumsikan matriks **A** dan **B** berupa matriks invertible berakibat $\mathbf{C} = \mathbf{AB}$ adalah invertible. Dengan

cara eselon baris tereduksi didapat *rref*(**A**)=*rref*(**B**)=I. Dengan menggunakan matriks elementer I=*rref*(**A**) = $E_1E_2 \dots E_kA^{-1}$ dan I=*rref*(**B**) = $F_1F_2 \dots F_l\mathbf{B}^{-1}$, maka **AB**=$E_1E_2 \dots E_kF_1F_2 \dots F_l$. Karena det(EX)= det(E)det(X) untuk E matriks elementer dan X sebarang matriks persegi. Sehingga diperoleh det(A)=det(E_1)det(E_2)... det(E_k) dan det(B)=det(F_1)det(F_2)... det(F_k). Jadi det(AB)=det(A)det(B).

Teorema 2.3.3. Sebuah matriks **A** persegi dapat dibalik jika dan hanya jika det(A) $\neq 0$

Bukti:

1. Jika **A** dapat dibalik, maka

   $$\mathbf{I} = \mathbf{A}\mathbf{A}^{-\mathbf{1}}$$

   sehingga

   $$\begin{aligned} det(\mathbf{I}) &= det(\mathbf{A}).\mathbf{det}(\mathbf{A}^{-\mathbf{1}}) \\ 1 &= det(\mathbf{A}).det(\mathbf{A}^{-\mathbf{1}}) \end{aligned}$$

   Maka det(A) $\neq 0$

2. Sebaliknya, anggap bahwa det(A) $\neq 0$. Akan ditunjukkan **A** ekuivalen baris pada **I**, maka **A** dapat dibalik.
   Misal **R** adalah bentuk esselon baris tereduksi dari **A**, maka ada $E_1, E_2, \dots, E_k$, maka dengan mengalikan kedua ruas dengan matriks balikan E didapatkan

   $$\begin{aligned} E_k \dots E_2 \dots E_1A &= R \\ (E_1^{-1}E_2^{-1} \dots E_k^{-1})(E_k \dots E_2E_1)A &= (E_1^{-1}, E_2^{-1}, \dots, E_k^{-1})R \\ A &= E_1^{-1}E_2^{-1} \dots E_k^{-1}R \\ det(A) &= det(E_1^{-1})det(E_2^{-1}) \dots det(E_k^{-1})det(R) \end{aligned}$$

Karena det(A) $\neq 0$, maka diperoleh det(R) $\neq 0$. Sehingga $\mathbf{R}$ tidak mempunyai barisan bilangan nol, $\mathbf{R} = \mathbf{I}$.

Definisi 2.3.11. (Kofaktor-Adjoin)

Jika $\mathbf{A}$ adalah matriks persegi dan C_{ij} adalah kofaktor dari a_{ij}, maka matriks

$$\mathbf{C} = \begin{bmatrix} C_{11} & C_{12} & \dots & C_{1n} \\ C_{21} & C_{22} & \dots & C_{2n} \\ \vdots & \vdots & & \vdots \\ C_{n1} & C_{n2} & \dots & C_{nn} \end{bmatrix} \tag{2.6}$$

disebut matriks kofaktor dari $\mathbf{A}$. Transpos dari matriks ini disebut adjoin dari $\mathbf{A}$ dan dinyatakan sebagai adj(A)

Teorema 2.3.4. Jika $\mathbf{A}$ adalah suatu matriks yang dapat dibalik, maka

$$\mathbf{A}^{-1} = \frac{1}{det(\mathbf{A})} adj(\mathbf{A}) \tag{2.7}$$

Bukti:

Akan ditunjukkan bahwa

$$\mathbf{A}adj(\mathbf{A}) = det(\mathbf{A})\boldsymbol{I}$$

Perhatikan hasil kali

$$\mathbf{A}adj(\mathbf{A}) = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{i1} & a_{i2} & \dots & a_{in} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} C_{11} & C_{21} & \dots & C_{j1} & \dots & C_{n1} \\ C_{12} & C_{22} & \dots & C_{j2} & \dots & C_{n2} \\ \vdots & \vdots & & \vdots & & \vdots \\ C_{1n} & C_{2n} & \dots & C_{jn} & & C_{nn} \end{bmatrix}$$

Entri pada baris ke-i dan kolom ke-j dari hasil kali $\mathbf{A}adj(\mathbf{A})$ adalah

$$a_{i1}C_{j1} + a_{i2}C_{j2} + \ldots + a_{in}C_{jn} \tag{2.8}$$

Jika i=j, maka (2.8) adalah ekspansi kofaktor dari det(A) sepanjang baris ke-i dari $\mathbf{A}$ dan jika $i \neq j$, maka semua a dan kofaktor-kofaktornya berasal dari baris-baris yang berbeda dari $\mathbf{A}$, sehingga nilai dari (2.8) adalah nol. Oleh karena itu,

$$\mathbf{A}adj(\mathbf{A}) = \begin{bmatrix} det(A) & 0 & \ldots & 0 \\ 0 & det(A) & \ldots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \ldots & det(A) \end{bmatrix} = det(\mathbf{A})\boldsymbol{I} \tag{2.9}$$

Karena $\mathbf{A}$ dapat dibalik, $det(\mathbf{A}) \neq 0$. Karena itu, persamaan (2.9) dapat ditulis kembali sebagai

$$\frac{1}{det\mathbf{A}}[\mathbf{A}adj(\mathbf{A})] = \boldsymbol{I} \tag{2.10}$$

atau

$$\mathbf{A}\left[\frac{1}{det\mathbf{A}}adj(\mathbf{A})\right] = \boldsymbol{I} \tag{2.11}$$

Dengan mengalikan kedua sisi disebelah kiri dengan $\mathbf{A}^{-1}$ menghasilkan

$$\mathbf{A}^{-1} = \frac{1}{det\mathbf{A}}adj(\mathbf{A}) \tag{2.12}$$

Matriks $\mathbf{A}$ akan memiliki invers jika $\mathbf{A}$ adalah matriks persegi dan $det\mathbf{A} \neq 0$. Jika det(A) = 0, maka $\mathbf{A}$ disebut matriks singular. Invers matriks $\mathbf{A}$ ditulis $\mathbf{A}^{-1}$.

2.4 Teori Bilangan

Definisi 2.4.1. (Greatest Common Divisors (GCD))

Jika a dan b adalah bilangan bulat yang sekurang-kurangnya salah satu dari a dan b tidak sama dengan nol maka faktor persekutuan terbesar dari a dan b, dinotasikan dengan GCD(a, b) adalah suatu bilangan bulat positif k yang memenuhi sifat berikut.

1. $k \mid a$ dan $k \mid b$
2. Jika terdapat suatu bilangan bulat j, $j \mid a$ dan $j \mid b$ maka $j \leq k$

Contoh:

Faktor-faktor bilangan bulat positif dari 12 adalah 1, 2, 3, 4, 6, 12.

Faktor-faktor bilangan bulat positif dari 30 adalah 1, 2, 3, 5, 6, 10, 15, 30.

Faktor-faktor persekutuan dari 12 dan 30 adalah 1, 2, 3, 6.

Maka, Faktor persekutuan terbesar dari 12 dan 30 adalah 6. GCD(12,30)=6.

Definisi 2.4.2. (Relatif Prima)

Misal $a, b \in \mathbb{Z}$ dan b tidak nol dikatakan relatif prima dimana GCD(a,b)=1

Contoh:

5 dan 14 relatif prima, karena GCD(5,14)=1

Definisi 2.4.3. (Keterbagian)

Jika a dan b adalah bilangan bulat dengan a dikatakan membagi (habis) b jika terdapat sebuah bilangan bulat k sedemikian sehingga $b = ak$. Jika a membagi (habis) b dikatakan juga bahwa a adalah pembagi atau faktor dari b. Jika a membagi (habis) b maka ditulis $a \mid b$ dan jika a tidak membagi (habis) b maka ditulis $a \nmid b$.

Contoh:

$4 \mid 124$ karena terdapat bilangan bulat 31 sedemikian sehingga (4)(31)=124

$7 \nmid 51$ karena tidak ada bilangan bulat k sedemikian sehingga 7k=51

Teorema 2.4.1. Jika nilai bilangan bulat a dan b relatif prima dan $a \mid b.m$ maka $a \mid m$

Bukti:

Diketahui a dan b relatif prima, yaitu $GCD(a, b) = 1$ dan $a \mid bm$, terdapat $x, y \in \mathbb{Z}$ sedemikian hingga $ax + by = 1$ dan $bm = ap$ untuk suatu $p \in \mathbb{Z}$ sehingga:

$$\begin{aligned} axm + bym &= m \\ axm + apy &= m \\ a(xm + py) &= m \end{aligned}$$

Terbukti $a \mid m$

Teorema 2.4.2. Jika $a, b \in \mathbb{Z}$ relatif prima, maka terdapat $m, n \in \mathbb{Z}$

$$ma + nb = 1$$

Bukti:

Misal $GCD(a, b) = n$ sedemikian hingga terdapat $m, n \in \mathbb{Z}$ dimana $ma + nb = n$ karena a, b relatif prima sedemikian hingga $GCD(a, b) = 1$ dan $ma + nb = 1$. Terbukti

2.4.1 Kekongruenan

Definisi 2.4.4. Misalkan $m \in \mathbb{N}$, maka dikatakan a kongruen terhadap r modulo m jika $m \mid (a - r)$. dinotasikan oleh

$$a \equiv r \pmod{m}$$

dan jika $m \nmid (a - r)$, maka dinotasikan

$$a \not\equiv r \pmod{m}$$

dikatakan bahwa a dan r tidak kongruen modulo m.

Contoh:

1. $18 \equiv 4 (\text{mod} 7)$ karena 18-4 = 14 dan 7 membagi 14, yaitu $7 \mid 14$.

2. $-22 \equiv 3 (\text{mod} 5)$ karena -22-3=-25 dan 5 membagi -25, yaitu $5 \mid -25$.

Teorema 2.4.3. Misalkan $m \in \mathbb{N}$ dan $a, b, c, d \in \mathbb{Z}$. Jika $a \equiv b \pmod{m}$ dan $c \equiv d \pmod{m}$, maka berlaku:

1. $-a \equiv -b \pmod{m}$

2. $a + c \equiv b + d \pmod{m}$

3. $ac \equiv bd \pmod{m}$

Bukti:

1. Karena m membagi $(a - b)$, maka m juga membagi $(-a + b)$, akibatnya terbukti bahwa $-a \equiv -b \pmod{m}$.

2. Karena m membagi $(a-b)$ dan $(c-d)$, maka m juga membagi $(a-b+c-d) = (a + c) - (b + d)$, dengan demikian terbukti $a + c \equiv b + d \pmod{m}$.

3. Untuk membuktikan $ac \equiv bd \pmod{m}$, misal $k, l \in \mathbb{Z}$ sedemikian sehingga $a = b + km$ dan $c = d + lm$ maka $ac = (b + km)(d + lm) = bd + m(kd + lb + lkm)$ atau dengan kata lain m membagi $(ac - bd)$, terbukti bahwa $ac \equiv db \pmod{m}$

Definisi 2.4.5. (Invers Modulo)

Jika a adalah sebuah bilangan di dalam $\mathbb{Z}_m$, maka sebuah bilangan a^{-1} di dalam $\mathbb{Z}_m$ disebuat sebuah resiprok atau invers perkalian dari a modulo m jika $aa^{-1} = a^{-1}a = 1 \pmod{m}$

Teorema 2.4.4. Sebuah matriks persegi $\mathbf{A}$ dengan entri-entri di dalam $\mathbb{Z}_m$ bersifat modulo m yang dapat dibalik jika dan hanya jika residu dari $det(\mathbf{A})$ modulo m mempunyai sebuah modulo m resiprok.

Bukti:

Karena residu dari $det(\mathbf{A})$ modulo m akan mempunyai sebuah modulo m resiprok jika dan hanya jika residu ini dan m tidak mempunyai faktor prima yang sama, maka berakibat:

Akibat 2.4.1. Sebuah matriks persegi $\mathbf{A}$ dengn entri-entri di dalam $\mathbb{Z}_m$ adalah modulo m yang dapat dibalik jika dan hanya jika m dan residu dari $det(\mathbf{A})$ modulo m tidak mempunyai faktor prima yang sama.

2.4.2 Algoritma Euclide

Algoritma Euclide digunakan untuk menghitung GCD dari dua bilangan. Misal r_0 dan r_1 dua bilangan bulat, dengan $r_0 \geq r_1 > 0$ maka untuk menghitung GCD (r_0, r_1) dengan menggunakan algoritma Euclide adalah sebagai berikut

1. Mencari sisa pembagian r_0 oleh r_1, yaitu r_2 dengan algoritma pembagian. sehingga
$$r_0 = q_1 r_1 + r_2, 0 \leq r_2 < r_1 \tag{2.13}$$
2. Jika $r_2 = 0$, maka $r_1 \mid r_0$ dan $\text{GCD}(r_0, r_1) = r_2$ akan tetapi, jika $r_2 \neq 0$ maka r_2 digunakan sebagai pembagi r_2 sehingga diperoleh bilangan bulat q_2 dan

r_3 yang memenuhi

$$r_1 = q_2 r_2 + r_3, 0 \leq r_3 < r_2 \tag{2.14}$$

Jika $r_3 = 0$ maka berhenti, jika sebaliknya, maka proses dilanjutkan. Sehingga diperoleh

$$r_2 = q_3 r_3 + r_4, 0 \leq r_4 < r_3 \tag{2.15}$$

3. Proses pembagian pada langkah 2) berlanjut hingga didapat $r_j = 0$. Apabila proses pembagian diatas dituliskan dalam sistem persamaan,

$$\begin{aligned} r_0 &= q_1 r_1 + r_2, 0 \leq r_2 < r_1 \\ r_1 &= q_2 r_2 + r_3, 0 \leq r_3 < r_2 \\ r_2 &= q_3 r_3 + r_4, 0 \leq r_4 < r_3 \\ &\vdots \\ r_{j-3} &= q_{j-2} r_{j-2} + r_{j-1}, 0 \leq r_{j-1} < r_{j-2} \\ r_{j-1} &= q_j r_j + 0 \end{aligned} \tag{2.16}$$

sehingga GCD(r_0, r_1)= GCD(r_1, r_2)= $\ldots$ = GCD(r_{j-1}, r_j)= GCD$(r_j, 0)$= r_j. Jadi, GCD$(r_0, r_1) = r_j$

2.4.3 Extended Euclide Algorithm

Perluasan dari algoritma Euclide dikenal dengan algoritma perluasan Euclide. Menurut Stinson(2006:166), Algoritma perluasan Euclide digunakan untuk menghitung nilai $r_1^{-1} \mathrm{mod} r_0$ dengan r_0, r_1 merupakan bilangan bulat positif, dan $r_0 > r_1$. Algoritma perluasan Euclide sebagai berikut:

1. Menghitung GCD (r_0, r_1) menggunakan algoritma Euclide

2. Apabila GCD $(r_0, r_1) = 1$ maka $r_1^{-1} \mathrm{mod} r_0$ ada dan $r_1^{-1} \mathrm{mod} r_0 = t_j \mathrm{mod} r_0$. Hal ini sesuai dengan teorema dan akibat berikut:

Teorema 2.4.5. Jika $0 \leq j \leq m$ maka $r_j = s_j r_0 + t_j r_1$, dengan r_j dan q_j seperti yang didefinisikan pada algoritma Euclide serta s_j dan t_j didefinisikan sebagai berikut: $t_j = 0$, jika j=0 $t_j = 1$, jika j=1 $t_j = t_{j-2} q_{j-1} t_{j-1}$, jika $j \geq 2$

Bukti:
Akan dibuktikan dengan menggunakan induksi matematika. Pernyataan benar untuk j=0 dan j=1.
Untuk j=0,

$$\begin{aligned} r_0 &= s_0 r_0 + t_0 r_1 \\ &= 1 r_0 + 0 r_1 \\ &= r_0 \end{aligned}$$

Untuk j=1

$$\begin{aligned} r_1 &= s_1 r_0 + t_1 r_1 \\ &= 0 r_0 + 1 r_1 \\ &= r_1 \end{aligned}$$

Diasumsikan pernyataan benar untuk j=i-1 dan j=i-2.
Untuk j=i-1,

$$r_{i-1} = s_{i-1} r_0 + t_{i-1} r_1 \tag{2.17}$$

Untuk j=i-2,

$$r_{i-2} = s_{i-2} r_0 + t_{i-2} r_1 \tag{2.18}$$

Akan dibuktikan pernyataan benar untuk j=i'

$$\begin{aligned} r_i &= r_{i-2} - q_{i-1}r_{i-1} \\ &= s_{i-2}r_0 + t_{i-2}r_1 - q_{i-1}(s_{i-1}r_0 + t_{i-1}r_1) \\ &= s_i r_0 + t_i r_1 \end{aligned}$$

Jadi terbukti hasil benar untuk semua bilangan bulat $j \geq 0$ dengan menggunakan induksi matematika.

Akibat dari teorema 2.4.5, dinyatakan sebagai berikut

Akibat 2.4.2. Andaikan GCD$(r_0, r_1) = 1$ maka $r_1^{-1}\text{mod}r_0 = t_j\text{mod}r_0$

Bukti:

$$\begin{aligned} GCD(r_0, r_1) &= s_j r_0 + t_j r_1 \\ s_j r_0 + t_j r_1 &= 1 \\ t_j r_1 &= 1 - s_j r_0 \end{aligned}$$

menurut teorema pada kekongruenan, $t_j r_1 = 1 \bmod r_0$.
Sehingga $t_j\text{mod } r_0 = r_1^{-1} \bmod r_0$

2.5 *Hill Cipher*

Hill Cipher merupakan penerapan aritmatika modulo pada kriptografi. Teknik kriptografi ini menggunakan sebuah matriks persegi sebagai kunci yang digunakan untuk melakukan enkripsi dan dekripsi. *Hill Cipher* diciptakan oleh *Lester S. Hill* pada tahun 1929. Teknik kriptografi ini diciptakan dengan maksud untuk dapat menciptakan *cipher* (kode) yang tidak dapat dipecahkan menggunakan teknik analisis frekuensi. *Hill Cipher* tidak mengganti setiap abjad yang sama

pada *plaintext* dengan abjad lainnya yang sama pada *ciphertext* karena menggunakan perkalian matriks untuk enkripsi dan dekripsinya.

Hill Cipher yang merupakan *polyalphabetic cipher* dapat dikategorikan sebagai *block cipher* karena teks yang akan diproses akan dibagi menjadi blok-blok dengan ukuran tertentu. Setiap karakter dalam satu blok akan saling mempengaruhi karakter lainnya dalam proses enkripsi dan dekripsinya, sehingga karakter yang sama tidak dipetakan menjadi karakter yang sama pula. *Hill Cipher* termasuk algoritma kriptografi klasik yang sangat sulit dipecahkan oleh kriptanalis apabila dilakukan hanya dengan mengetahui berkas *ciphertext* saja. Namun, teknik ini dapat dipecahkan dengan cukup mudah apabila kriptanalis memiliki berkas *ciphertext* dan potongan berkas *plaintext*. Teknik kriptanalisis ini disebut *known-plaintext attack*.

Dasar dari teknik *Hill Cipher* adalah aritmatika modulo terhadap matriks. Dalam penerapannya, *Hill Cipher* menggunakan teknik perkalian matriks dan teknik invers terhadap matriks. Kunci pada *Hill Cipher* adalah matriks $n \times n$ dengan n merupakan ukuran blok. Matriks $\mathbf{K}$ yang menjadi kunci ini harus merupakan matriks yang invertible, yaitu memiliki inverse, sehingga $\mathbf{K}\mathbf{K}^{-1} = \mathbf{I}$. Kunci harus memiliki invers karena matriks tersebut adalah kunci yang digunakan untuk melakukan dekripsi.

2.5.1 Teknik Enkripsi pada *Hill Cipher*

Proses enkripsi pada *Hill cipher* dilakukan per blok *plaintext*. Ukuran blok tersebut sama dengan ukuran matriks kunci. Sebelum membagi teks menjadi deretan blok-blok, *plaintext* terlebih dahulu dikonversi menjadi angka, misal $A = 0, B = 1, \ldots Z = 25$. P adalah himpunan *plaintext* $\{p_1,\ p_2,\ p_3,\ \ldots p_n\}$.

K adalah matriks kunci ukuran 3x3. C adalah himpunan *ciphertext* $\{c_1, c_2, c_3, \ldots c_n\}$. Secara matematis, proses enkripsi pada Hill Cipher adalah:

$$C = \mathbf{K}.P \pmod{m} \tag{2.19}$$

Algoritma enkripsi pada *Hill cipher* sebagai berikut:

1. Pilih entri matriks kunci $\mathbf{K}_{3\times3}$ yang bersesuaian dengan syarat $\mathbf{K}$ invertible dan $det(\mathbf{K} \neq 0)$ relatif prima terhadap bilangan $\mathrm{mod} m$.

2. Pilih *plaintext* kemudian dibagi menjadi blok-blok dengan banyak anggota 3 (apabila kelompok terakhir tidak memiliki panjang 3, tambahkan karakter dummy apa saja (yang disepakati) sehingga setiap blok memiliki 3 anggota). P=$\{p_1, p_2, p_3, \ldots p_n\}$

3. Ubah *plaintext* ke bilangan yang bersesuaian menggunakan tabel konversi.

4. Menggunakan operasi transpos, bentuk *plaintext* ke dalam matriks kolom

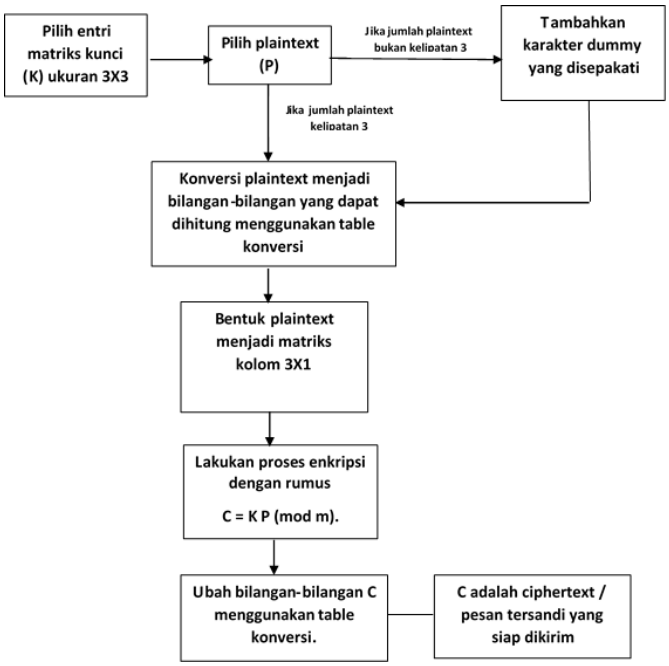
$$\mathbf{P}_{3\times1} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

5. Untuk mendapatkan *ciphertext*, lakukan teknik enkrip dengan rumus $C = \mathbf{K}.P \pmod{m}$.

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} k_1 1 & k_1 2 & k_1 3 \\ k_2 1 & k_2 2 & k_2 3 \\ k_3 1 & k_3 2 & k_3 3 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

6. Untuk setiap blok matriks kolom *plaintext*, lakukan perhitungan yang sama.

7. Ubah angka-angka tersebut menjadi *ciphertext* menggunakan tabel konversi.



Gambar 2.2: diagram alur enkripsi Hill cipher

2.5.2 Teknik Dekripsi pada *Hill Cipher*

Proses dekripsi pada Hill Cipher pada dasarnya sama dengan proses enkripsinya. Namun matriks kunci harus dibalik (invers) terlebih dahulu. Secara matematis, proses dekripsi pada Hill Cipher dapat diturunkan dari persamaan.

$$
\begin{aligned}
C &= \mathbf{K}P \bmod m \\
C &= \{\mathbf{K} \bmod m\}\{P \bmod m\} \\
\{\mathbf{K}^{-1} \bmod m\}C &= \{\mathbf{K}^{-1} \bmod m\}\{\mathbf{K} \bmod m\}\{P \bmod m\} \\
\mathbf{K}^{-1}C \bmod m &= \{\mathbf{K}^{-1}\mathbf{K} \bmod m\}\{P \bmod m\} \\
\mathbf{K}^{-1}C \bmod m &= IP \bmod m \\
P &= \mathbf{K}^{-1}C \bmod m
\end{aligned}
$$

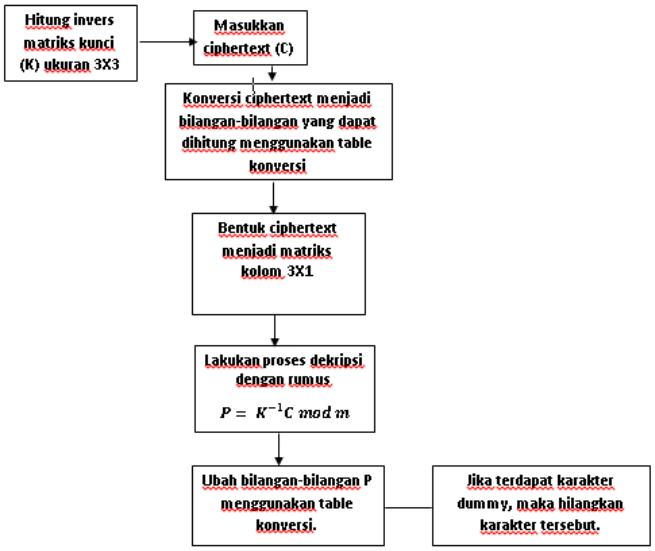
Menjadi persamaan proses dekripsi:

$$P = \mathbf{K}^{-1}C \pmod{m} \tag{2.20}$$

Proses dekripsi diawali dengan mencari invers dari matriks **K**. Mencari invers dilakukan dengan beberapa tahap:

1. Hitung determinan kunci (**K**),
2. Hitung residu r dari determinan kunci (**K**), misal Z.
3. Hitung invers menggunakan algoritma euclide yang diperluas Z^{-1}.
4. Hitung adjoint matrik kunci **K**
5. maka $\mathbf{K^{-1}} = Z^{-1} \times adj(\mathbf{K}) \pmod{m}$

Kemudian dengan tahap yang sama dengan enkripsi dilakukan proses dekripsi. Catatan: apabila kelompok terakhir memiliki karakter dummy, kurangkan karakter dummy (yang disepakati) tersebut sehingga didapatkan *plaintext* semula.



Gambar 2.3: diagram alur dekripsi Hill cipher

Contoh 2.5.1. Diberikan tabel konversi karakter sebagai berikut:

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

Terdapat *plaintext* (P) = MATEMATIKA, dengan menggunakan tabel konversi diatas, *plaintext* tersebut dikonversi menjadi P = 12 0 19 4 12 0 19 8 10 0. *Plaintext* tersebut akan dienkripsi dengan teknik Hill Cipher, dengan kunci K yang merupakan matriks 3x3.

$$\mathbf{K} = \begin{bmatrix} 3 & 1 & -2 \\ 5 & 8 & 7 \\ 1 & 0 & 8 \end{bmatrix}$$

Matriks kunci **K** berukuran 3x3, maka *plaintext* dibagi menjadi blok yang masing-masing bloknya berukuran 3 karakter. Karakter terakhir tidak memiliki pasangan, maka diberi tambahan karakter yang sama yaitu dua karakter Z, sehingga membentuk blok yang terdiri dari 3 karakter. Nilai *plaintext* yang telah dikonversi menggunakan tabel dibagi menjadi blok-blok *plaintext* (matriks kolom) sebagai berikut:

P_1= {12 0 19}

P_2= {4 12 0}

P_3= {19 8 10}

P_4= {0 25 25}

atau dapat disusun menjadi kumpulan matriks kolom sebagai berikut:

$$\begin{bmatrix} 12 & 4 & 19 & 0 \\ 0 & 12 & 8 & 25 \\ 19 & 0 & 10 & 25 \end{bmatrix}$$

Selanjutnya dilakukan perhitungan untuk memperoleh *ciphertext* dengan persamaan (2.19)

$$\begin{aligned} C &= KP(\mathrm{mod} n) \\ &= \begin{bmatrix} 3 & 1 & -2 \\ 5 & 8 & 7 \\ 1 & 0 & 8 \end{bmatrix} \begin{bmatrix} 12 & 4 & 19 & 0 \\ 0 & 12 & 8 & 25 \\ 19 & 0 & 10 & 25 \end{bmatrix} \bmod 26 \\ &= \begin{bmatrix} -2 & 24 & 45 & -25 \\ 193 & 116 & 229 & 375 \\ 164 & 4 & 99 & 200 \end{bmatrix} (\mathrm{mod} 26) \\ &= \begin{bmatrix} 24 & 24 & 19 & 1 \\ 11 & 12 & 21 & 11 \\ 8 & 4 & 21 & 18 \end{bmatrix} \end{aligned}$$

Setelah melakukan enkripsi maka dihasilkan *ciphertext* C sebagai berikut: 24 11 8 24 12 4 19 21 21 1 11 18. Selanjutnya dikonversi menjadi: Y L I Y M E T V V B L S

Dari *ciphertext* yang dihasilkan terlihat bahwa Hill Cipher menghasilkan *ciphertext* yang tidak memiliki pola yang mirip dengan *plaintext*nya.

Untuk melakukan dekripsi, akan digunakan persamaan (2.20) dengan *ciphertext* "YLIYMETVVBLS". *Ciphertext* dikonversi menjadi bilangan-bilangan sesuai tabel konversi sebagai berikut:

24 11 8 24 12 4 19 21 21 1 11 18

Menurut persamaan (2.20), dibutuhkan invers **K**. $\mathbf{K}^{-1}$ dapat diperoleh dengan cara $\mathbf{K}^{-1} = \det\,(\mathrm{K})^{-1}\mathrm{adj}\,(\mathrm{K})$

- Menghitung $\det\,(\mathrm{K})^{-1}$:

$$\mathbf{K} = \begin{bmatrix} 3 & 1 & -2 \\ 5 & 8 & 7 \\ 1 & 0 & 8 \end{bmatrix}$$

Menggunakan perhitungan determinan, det(K)= 175. Kemudian dihitung nilai Z, 175 mod 26 = 19. Terdapat invers dari matriks kunci **K**, dengan menggunakan algoritma euclide yang diperluas akan dihitung invers dari Z=19 sebagai berikut:

$$\begin{aligned} 26 &= 19.1 + 7 \\ 19 &= 7.2 + 5 \\ 7 &= 5.1 + 2 \\ 5 &= 2.2 + 1 \end{aligned}$$

maka

$$\begin{aligned} x_0 &= 0 \\ x_1 &= 1 \\ x_2 &= 0 - 1(1) \bmod 26 = 25 \\ x_3 &= 1 - 25(2) \bmod 26 = 3 \\ x_4 &= 25 - 3(1) \bmod 26 = 22 \\ x_5 &= 3 - 22(2) \bmod 26 = 11 \end{aligned}$$

didapatkan bahwa Z^{-1}=11.

- Berdasarkan definisi Adjoin, maka dapat dihitung adjoint (**K**):

$$Adj(\mathbf{K}) = \begin{bmatrix} 64 & -8 & 23 \\ -33 & 26 & -31 \\ -8 & 1 & 19 \end{bmatrix}$$

- Menghitung invers modulo:

$$\begin{aligned} \mathbf{K}^{-1} &= det(\mathbf{K})^{-1} adj(\mathbf{K}) (\mathrm{mod} n) \\ &= 11 \begin{bmatrix} 64 & -8 & 23 \\ -33 & 26 & -31 \\ -8 & 1 & 19 \end{bmatrix} (\mathrm{mod} 26) \\ &= \begin{bmatrix} 704 & -88 & 253 \\ -363 & 286 & -341 \\ -88 & 11 & 209 \end{bmatrix} (\mathrm{mod} 26) \\ &= \begin{bmatrix} 2 & 16 & 19 \\ 1 & 0 & 23 \\ 16 & 11 & 1 \end{bmatrix} \end{aligned}$$

Melakukan proses dekripsi berdasarkan persamaaan (2.20)

$$= \begin{bmatrix} 2 & 16 & 19 \\ 1 & 0 & 23 \\ 16 & 11 & 1 \end{bmatrix} \begin{bmatrix} 24 & 24 & 19 & 1 \\ 11 & 12 & 21 & 11 \\ 8 & 4 & 21 & 18 \end{bmatrix} (\text{mod}26)$$

$$= \begin{bmatrix} 376 & 316 & 773 & 520 \\ 208 & 116 & 502 & 415 \\ 513 & 520 & 556 & 155 \end{bmatrix} (\text{mod}26)$$

$$= \begin{bmatrix} 12 & 4 & 19 & 0 \\ 0 & 12 & 8 & 25 \\ 19 & 0 & 10 & 25 \end{bmatrix}$$

Setelah semua blok selesai didekripsi, maka didapatkan hasil *plaintext*: 12 0 19 4 12 0 19 8 10 0 25 25. dikonversi menggunakan tabel konversi menjadi: M A T E M A T I K A Z Z

Karena terdapat karakter dummy, maka ZZ dihilangkan. *Plaintext* yang didapatkan adalah MATEMATIKA.

2.6 *Vigenère Cipher*

Vigenère cipher adalah suatu algoritma kriptografi klasik yang ditemukan oleh *Giovan Battista Bellaso*. Beliau menuliskan metodenya pada buku yang berjudul *La Cifra del. Sig. Giovan Battista Bellaso* pada tahun 1553. Nama *Vigenère* sendiri diambil dari seorang yang bernama *Blaise de Vigenère*. Nama *Vigenère* diambil sebagai nama algoritma ini karena beliau menemukan kunci yang lebih kuat untuk algoritma ini dengan metode *autokey cipher* meskipun algoritma dasarnya telah ditemukan oleh *Giovan Battista Bellaso*.

Algoritma *Vigenère cipher* ini adalah suatu algoritma yang dirancang untuk memperbaiki kelemahan dari algoritma subsitusi tunggal, yaitu setiap karakter pada *plaintext* disubsitusikan dengan karakter yang sama. Algoritma subsitusi tunggal ini mudah dipecahkan dengan metode analisis frekuensi, yaitu dengan menghitung frekuensi kemunculan huruf pada bahasa tertentu. Misalnya pada bahasa Inggris, huruf E adalah huruf yang sering muncul, sedangkan pada bahasa Indonesia, huruf A adalah huruf yang sering muncul.

2.6.1 Teknik Enkripsi pada *Vigenère cipher*

Algoritma *Vigenère cipher* dirancang untuk menghilangkan pola frekuensi huruf pada *ciphertext*. Dengan begitu, *ciphertext* yang dihasilkan dengan algoritma *Vigenère cipher* ini tidak akan dapat dikenakan metode analisis frekuensi. Algoritma ini bekerja dengan cara yang hampir mirip dengan algoritma *Caesar cipher*. Namun, pada algoritma *Vigenère cipher* ini, pergeseran yang dilakukan tidak selalu sama seperti pada *Caesar cipher*.

Pada algoritma *Vigenère cipher*, jauh pergeseran huruf plaintext ditentukan oleh nilai huruf kunci yang bersesuaian dengan *plaintext*nya (A=0, B=1, ..., Z=25) *Vigenère cipher* menggunakan suatu kunci yang memiliki panjang tertentu, misal K=$k_0, k_1, k_2, k_3, \ldots k_{l-1}$. Panjang kunci dapat lebih pendek ataupun sama dengan panjang *plaintext* P=$p_0, p_1, p_2, p_3, \ldots p_{n-1}$. Jika panjang kunci kurang dari panjang *plaintext* ($l < n$), maka kunci tersebut akan berulang periodik hingga sama dengan panjang *plaintext*nya. *Ciphertext* C=$c_0, c_1, c_2, c_3, \ldots c_{n-1}$ da-

pat dituliskan sebagai:

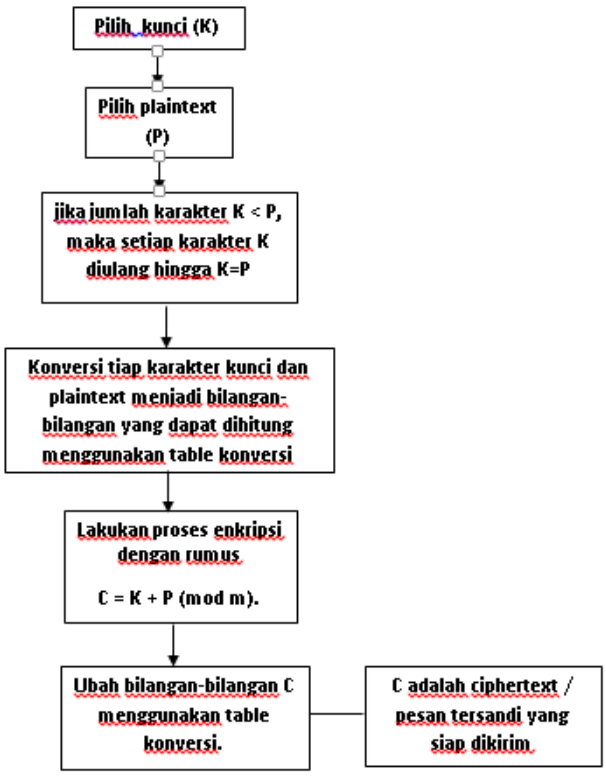
$$\begin{aligned}(c_0, c_1, c_2, c_3, \ldots c_{n-1}) &= (k_0, k_1, k_2, k_3, \ldots k_{l-1}) + (p_0, p_1, p_2, p_3, \ldots p_{n-1}) \bmod m \\ &= (p_0 + k_0) \bmod m, (p_1 + k_1) \bmod m, \ldots (p_{l-1} + k_{l-1}) \bmod m, \\ &\quad (p_l + k_0) \bmod m, (p_{l+1} + k_1) \bmod m, \ldots, \\ &\quad (p_{2l-1} + k_{l-1}) \bmod m, \ldots\end{aligned}$$

Maka persamaan untuk enkripsi pada *Vigenère cipher* adalah:

$$C_i = (K_i + P_{i \bmod l}) \bmod m \tag{2.21}$$

Algoritma enkripsi pada *Vigenère cipher*:

1. Pilih *plaintext* yang akan dienkripsi.
2. Pilih kata kunci, yaitu sebuah kata yang memiliki panjang l.
3. Perubahan kata kunci menjadi angka yang bersesuaian dengan tabel konversi (misal bilangan itu $k_1, k_2, \ldots, k_n$)
4. Jika $l \leq n$ maka kata kunci diulang hingga $l = n$
5. Ubah *plaintext* kedalam bilangan bulat yang bersesuaian dengan tabel konversi (misal $P_1, P_2, \ldots, P_n$)
6. Untuk melakukan enkripsi, gunakan persamaan (2.21), $C_i = P_i + K_i (\bmod m)$
7. Ubah ke huruf yang bersesuaian, hasil akhir huruf-huruf tersebut disebut *ciphertext*

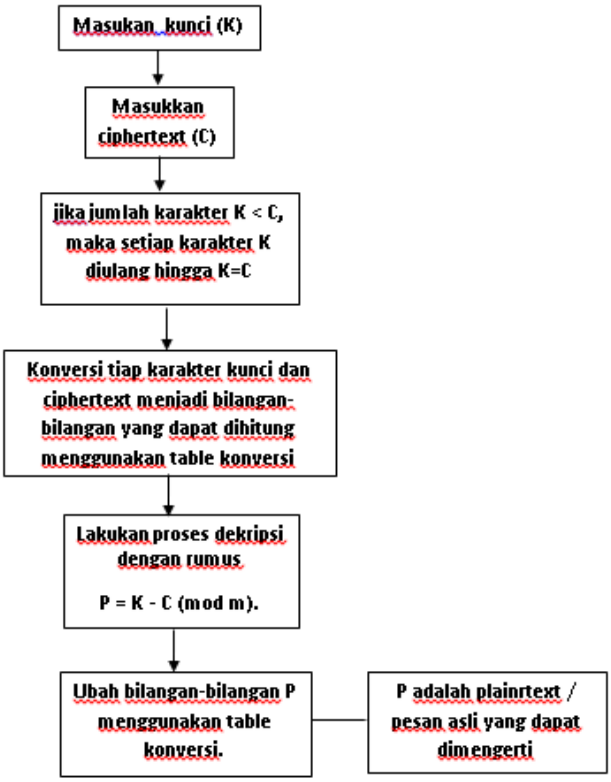


Gambar 2.4: diagram alur enkripsi Vigenere cipher

2.6.2 Teknik Dekripsi pada *Vigenère cipher*

Proses dekripsi pada *Vigenère cipher* adalah invers dari proses enkripsinya. Secara matematis, proses dekripsi pada *Vigenère* Cipher dapat diturunkan sebagai berikut.

$$\begin{aligned} C &= (P + K) \bmod m \\ C &= P \bmod m + K \bmod m \\ P \bmod m &= C - K \bmod m \\ P &= C - K \bmod m \end{aligned} \tag{2.22}$$



Gambar 2.5: diagram alur dekripsi Vigenere cipher

Selain rumus matematika diatas, proses enkripsi dan dekripsi pada *Vigenère cipher* dapat menggunakan tabel *Vigenère* 2.2.

VIGENERE TABEL

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

Gambar 2.6: Tabel *Vigenère*

Contoh 2.6.1. Menggunakan contoh (2.5.1), terdapat *plaintext* MATEMATIKA. *Plaintext* di konversi menggunakan tabel konversi menjadi 12 0 19 4 12 0 19 8 10 0.

Akan di enkrip menggunakan kunci SAYA. Karena panjang plainteks 10 karakter, maka kunci SAYA akan mengalami perulangan menjadi SAYASAYASA dikonversi menjadi 18 0 25 0 18 0 25 0 18 0.

Menggunakan persamaan (2.21) dilakukan proses enkrip:

M: $C = 12 + 18 \bmod 26 = 4$

A: $C = 0 + 0 \bmod 26 = 0$

T: $C = 19 + 25 \bmod 26 = 18$

E: $C = 4 + 0 \bmod 26 = 4$

M: $C = 12 + 18 \bmod 26 = 4$

A: $C = 0 + 0 \bmod 26 = 0$

T: C = 19 + 25 mod 26 = 18

I: C = 8 + 0 mod 26 = 8

K: C = 10 + 18 mod 26 = 2

A: C = 0 + 0 mod 26 = 0

Maka *ciphertext* menjadi 4 0 18 4 4 0 18 8 2 0. Konversi menggunakan tabel menjadi EASEEASICA

Dekripsi:

Untuk mendekripsi *ciphertext* menjadi *plaintext* kembali, akan digunakan persamaan (2.22). Menggunakan *ciphertext* EASEEASICA kemudian dikonversi menggunakan tabel konversi menjadi 4 0 18 4 4 0 18 8 2 0. Dengan kunci yang sama SAYA dan panjang *plaintext* 10 karakter, maka kunci SAYA akan mengalami perulangan menjadi SAYASAYASA dikonversi menjadi 18 0 25 0 18 0 25 0 18 0. Proses dekripsi dengan menggunakan persamaan (2.22)

E: P = 4 - 18 mod 26 = 12

A: P = 0 - 0 mod 26 = 0

S: P = 18 - 25 mod 26 = 19

E: P = 4 - 0 mod 26 = 4

E: P = 4 - 18 mod 26 = 12

A: P = 0 - 0 mod 26 = 0

S: P = 18 - 25 mod 26 = 19

I: P = 8 - 0 mod 26 = 8

C: P = 2 - 18 mod 26 = 10

A: P = 0 - 0 mod 26 = 0

Didapatkan *plaintext* 12 0 19 4 12 0 19 8 10 0, dikonversi menggunakan tabel konversi menjadi M A T E M A T I K A.

BAB III

PEMBAHASAN

3.1 Modifikasi *Hill Cipher* yang diperkuat dengan *Vigenère Cipher*

Algoritma ini adalah penggabungan dua algoritma klasik yang telah lama digunakan yaitu algoritma *Hill cipher* dan algoritma *Vigenère cipher*. Algoritma *Hill cipher* menitik beratkan pada kombinasi linear dari kunci yang digunakan untuk mengenkripsi dan mendekripsi juga berkaitan dengan operasi matriks dan bilangan modulo. Algoritma *Vigenère cipher* merupakan algoritma klasik yang menggunakan konsep modulo dalam mengenkripsi dan mendekripsi suatu *plaintext* menjadi *ciphertext*. Algoritma ini menggunakan operasi penjumlahan dan pengurangan bilangan modulo.

Modifikasi *Hill cipher* yang diperkuat *Vigenère cipher* merupakan penyandian baru, dengan cara menggabungkan dua buah metode. *Hill cipher* memiliki kelemahan dapat dipecahkan jika kriptanalisis memiliki cipher dan potongan *plaintext* (*know plaintext attact*). Penggabungan *Vigenère* cipher ke dalam *Hill cipher* memungkinkan kombinasi linear pada *Hill cipher* menjadi lebih acak sehingga akan lebih sulit untuk dipecahkan oleh kripanalisis.

3.1.1 Enkripsi pada *Hill Cipher* yang diperkuat dengan *Vigenère Cipher*

Proses Enkripsi pada modifikasi *Hill cipher* yang diperkuat *Vigenère cipher* yaitu dengan mengkomposisi fungsi *Hill cipher* kedalam fungsi *Vigenère cipher*.

$$f(V \circ H) = f(V(H)) = C$$

Perhatikan persamaan (2.19) membahas proses enkripsi pada *Hill cipher*, yaitu:

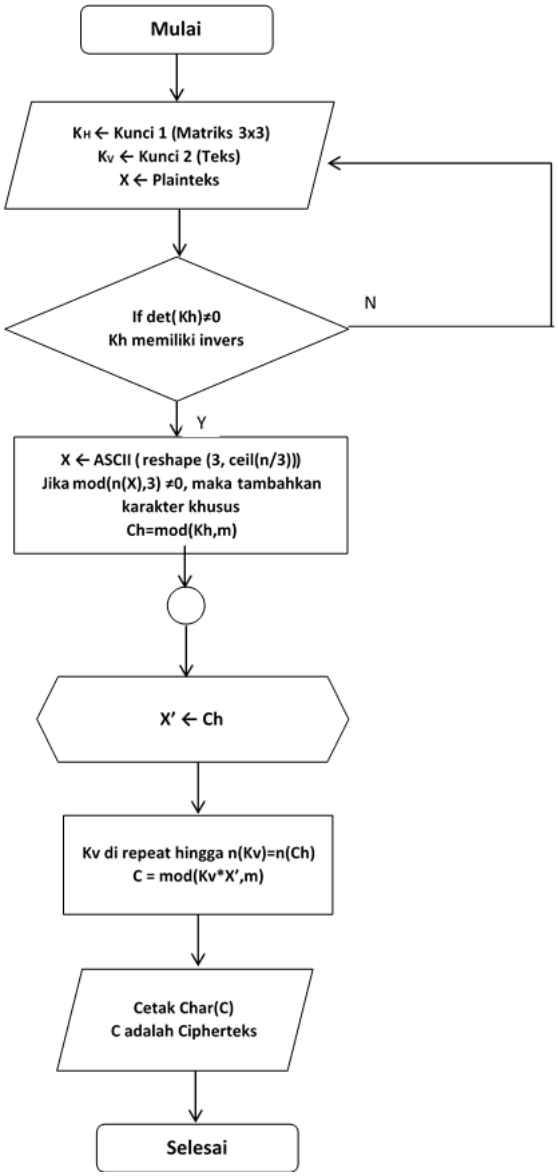
$$C_H = \mathbf{K}_H P (\mathrm{mod} m)$$

Dengan melanjutkan proses enkripsi *Vigenère cipher* yaitu persamaan (2.21), maka C pada *Hill cipher* dipandang sebagai *plaintext* pada *Vigenère cipher*. menjadi:

$$\begin{aligned}
C_V &= P + K_V (\mathrm{mod} m) \\
C_V &= ((\mathbf{K}_H P (\mathrm{mod} m)) + K_V) \bmod m \\
C_V &= (\mathbf{K}_H P)(\mathrm{mod} m) + K_V (\mathrm{mod} m) \\
C_V &= (\mathbf{K}_H \bmod m)(P \bmod m) + K_V \bmod m \\
C_V &= (\mathbf{K}_H P + K_V)(\mathrm{mod} m)
\end{aligned}$$

Dari persamaan diatas, maka didapatkan rumus enkripsi pada modifikasi *Hill cipher* yang diperkuat dengan *Vigenère cipher*, yaitu:

$$C = (\mathbf{K}_H . P + K_V)(\mathrm{mod} m) \tag{3.1}$$



Gambar 3.1: Flow Chart Enkripsi Hill Vigenère

3.1.2 Dekripsi pada *Hill Cipher* yang diperkuat dengan *Vigenère Cipher*

Proses dekripsi pada modifikasi *Hill cipher* yang diperkuat dengan *Vigenère cipher* merupakan komposisi balik dari fungsi *Hill cipher* kedalam fungsi *Vigenère cipher*

$$f(H \circ V) = f(H(V)) = P$$

Perhatikan persamaan (2.22), membahas proses enkripsi pada *Vigenere cipher*yaitu:

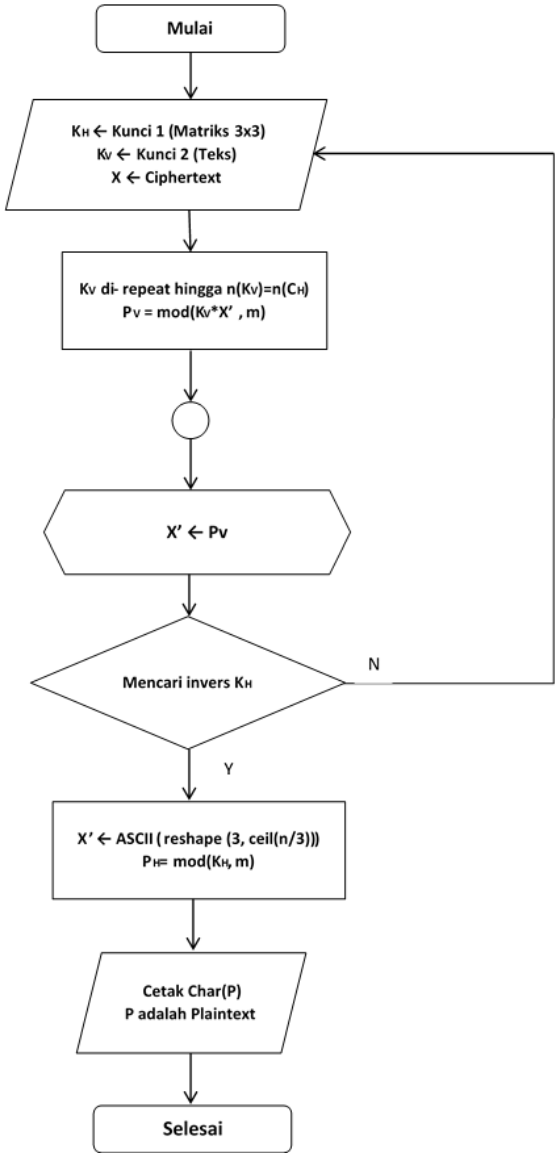
$$P_V = C_V - K_V \pmod{m}$$

Dengan melanjutkan proses dekripsi *Hill cipher* yaitu persamaan (2.20), maka P pada *Vigenère cipher* dipandang sebagai *ciphertext* pada *Hill cipher*. menjadi:

$$\begin{aligned}
P_H &= \mathbf{K_H}^{-1} C \pmod{m} \\
P_H &= (\mathbf{K_H}^{-1}(C_V - K_V \bmod m)) \bmod m \\
P_H &= \mathbf{K_H}^{-1} \bmod m (C_V - K \bmod m) \bmod m \\
P_H &= \mathbf{K_H}^{-1} \bmod m (C_V \bmod m - K_V \bmod m) \\
P_H &= \mathbf{K_H}^{-1} (C_V - K) \bmod m
\end{aligned}$$

Dari persamaan diatas, maka didapatkan rumus dekripsi pada modifikasi *Hill cipher* yang diperkuat dengan *Vigenère cipher*, yaitu:

$$P = \mathbf{K}_H^{-1}.(C - K_V) \bmod m \tag{3.2}$$



Gambar 3.2: Flow Chart Dekripsi Hill Vigenère

3.2 Program MATLAB untuk Sandi Data Menggunakan *Hill cipher* yang Diperkuat dengan *Vigenère cipher*

MATLAB merupakan singkatan dari Matric Laboratory, merupakan bahasa pemrograman *high performance*, bahasa pemrograman level tinggi yang khususnya untuk komputasi teknis. Bahasa ini mengintegrasikan kemampuan komputasi, visualisasi dan pemrograman dalam sebuah lingkungan yang tunggal dan mudah digunakan. Memiliki kemampuan untuk menyelesaikan berbagai kasus yang berhubungan langsung dengan matematika, rekayasa teknik, fisika, statistika, komputasi dan modeling.

MATLAB adalah satu program dengan bahasa pemrograman modern yang dapat menghitung, memvisualisasikan, serta membuat (GUI) yaitu form yang input datanya dapat dipilih secara acak oleh pengguna program.

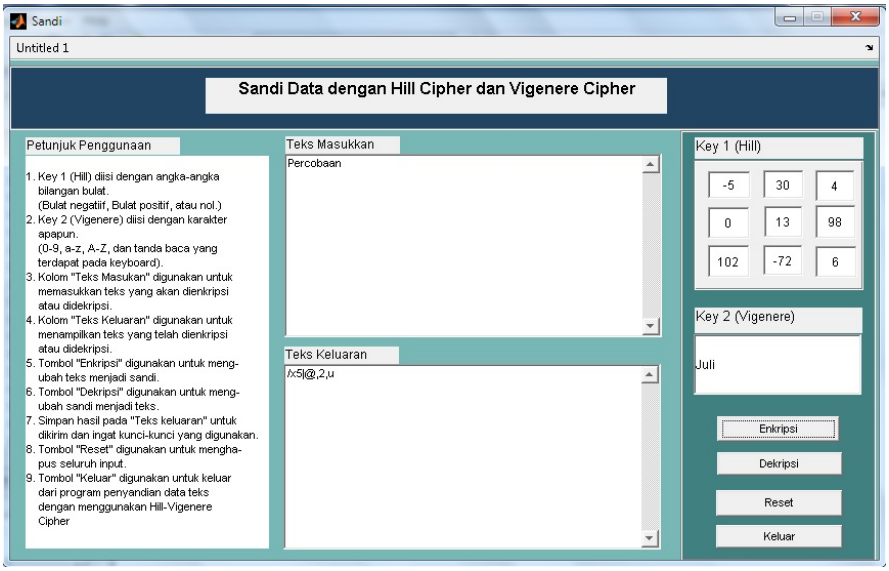
Algoritma *Hill cipher* yang diperkuat dengan *Vigenere cipher* pada MATLAB, menggunakan tabel konversi ASCII Gambar 3.3 nomor 32 hingga 127.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Gambar 3.3: Tabel konversi ASCII

Contoh Proses Enkripsi dan Dekripsi menggunakan tabel konversi ASCII



Gambar 3.4: Contoh Proses Enkripsi

Ditentukan:

1. *Plaintext*: Percobaan. Dikonversi menjadi 80 101 114 99 111 98 97 97 110. Kemudian dibentuk menjadi matriks:

$$\mathbf{P} = \begin{bmatrix} 80 & 99 & 97 \\ 101 & 111 & 97 \\ 114 & 98 & 110 \end{bmatrix}$$

Karena menggunakan tabel konversi ASCII Gambar 3.3 nomor 32 hingga 127. Setiap entri matriks dikurangi 32 menjadi:

$$\mathbf{P} = \begin{bmatrix} 48 & 67 & 65 \\ 69 & 79 & 65 \\ 82 & 66 & 78 \end{bmatrix}$$

2. Kunci *Hill cipher* adalah:

$$\mathbf{K}_H = \begin{bmatrix} -5 & 30 & 4 \\ 0 & 13 & 98 \\ 102 & -72 & 6 \end{bmatrix}$$

3. Kunci *Vigenère cipher* adalah: Juli. Diulang hingga jumlahnya sama dengan jumlah *plaintext*, menjadi JuliJuliJ. Dikonversi menjadi 74 117 108 105 74 117 108 105 74. Kemudian diubah menjadi matriks:

$$\mathbf{K}_V = \begin{bmatrix} 74 & 105 & 108 \\ 117 & 74 & 105 \\ 108 & 117 & 74 \end{bmatrix}$$

Karena menggunakan tabel konversi ASCII Gambar 3.3 nomor 32 hingga 127. Setiap entri matriks dikurangi 32 menjadi:

$$\mathbf{K}_V = \begin{bmatrix} 42 & 73 & 76 \\ 85 & 42 & 73 \\ 76 & 85 & 42 \end{bmatrix}$$

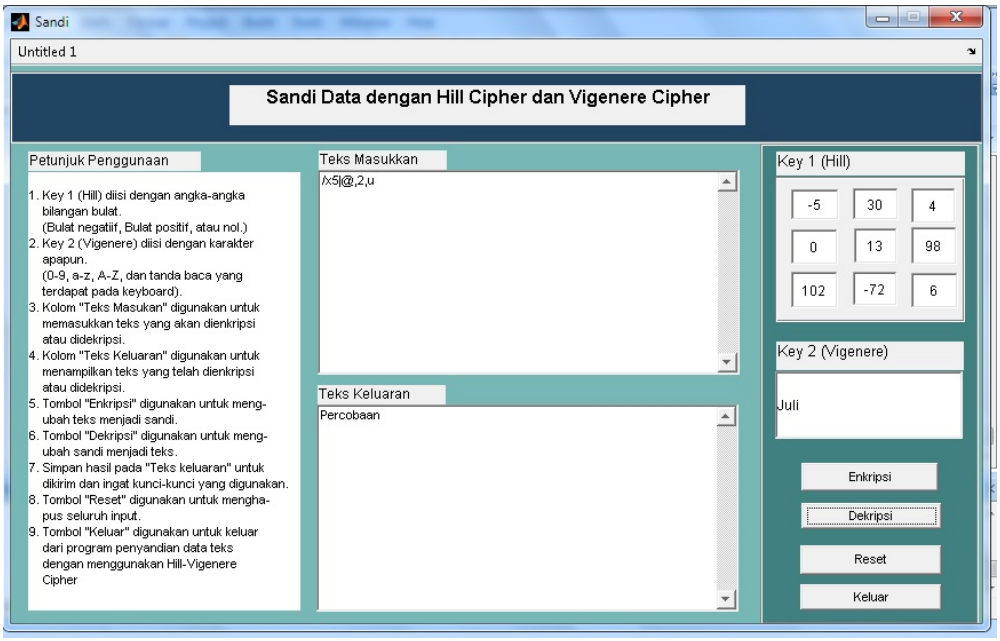
Menggunakan persamaan (3.1) dilakukan enkripsi pada modifikasi *Hill cipher* yang diperkuat dengan *Vigenère cipher*, yaitu

$$
\begin{aligned}
C &= (\mathbf{K}_H P + K_V)(\mathrm{mod} m) \\
C &= \begin{bmatrix} -5 & 30 & 4 \\ 0 & 13 & 98 \\ 102 & -72 & 6 \end{bmatrix} \begin{bmatrix} 48 & 67 & 65 \\ 69 & 79 & 65 \\ 82 & 66 & 78 \end{bmatrix} + \begin{bmatrix} 42 & 73 & 76 \\ 85 & 42 & 73 \\ 76 & 85 & 42 \end{bmatrix} (\mathrm{mod} 95) \\
&= \begin{bmatrix} 2158 & 2299 & 1937 \\ 8933 & 7495 & 8489 \\ 420 & 1542 & 2418 \end{bmatrix} + \begin{bmatrix} 42 & 73 & 76 \\ 85 & 42 & 73 \\ 76 & 85 & 42 \end{bmatrix} (\mathrm{mod} 95) \\
&= \begin{bmatrix} 2200 & 2372 & 2013 \\ 9018 & 7537 & 8562 \\ 496 & 1627 & 2460 \end{bmatrix} (\mathrm{mod} 95) \\
&= \begin{bmatrix} 15 & 92 & 18 \\ 88 & 32 & 12 \\ 21 & 12 & 85 \end{bmatrix}
\end{aligned}
$$

Karena menggunakan tabel konversi ASCII Gambar 3.3 nomor 32 hingga 127. Setiap entri matriks ditambah 32 menjadi:

$$
\mathbf{P} = \begin{bmatrix} 47 & 124 & 50 \\ 120 & 64 & 44 \\ 53 & 44 & 117 \end{bmatrix}
$$

didapatkan *ciphertext* adalah 47 120 53 124 64 44 50 44 117 dan dikonversi kedalam bentuk teks menjadi /x5|@,2,u .



Gambar 3.5: Contoh Proses Dekripsi

Masukkan:

1. *Ciphertext*: /x5|@,2,u . Dikonversi menjadi 47 120 53 124 64 44 50 44 117. Kemudian dibentuk menjadi matriks:

$$\mathbf{C} = \begin{bmatrix} 47 & 124 & 50 \\ 120 & 64 & 44 \\ 53 & 44 & 117 \end{bmatrix}$$

Karena menggunakan tabel konversi ASCII Gambar 3.3 nomor 32 hingga 127. Setiap entri matriks dikurangi 32 menjadi:

$$\mathbf{C} = \begin{bmatrix} 15 & 92 & 18 \\ 88 & 32 & 12 \\ 21 & 12 & 85 \end{bmatrix}$$

2. Kunci *Hill cipher* adalah:

$$\mathbf{K}_H = \begin{bmatrix} -5 & 30 & 4 \\ 0 & 13 & 98 \\ 102 & -72 & 6 \end{bmatrix}$$

3. Kunci *Vigenère cipher* adalah: Juli. Diulang hingga jumlahnya sama dengan jumlah *plaintext*, menjadi JuliJuliJ. Dikonversi menjadi 74 117 108 105 74 117 108 105 74. Kemudian diubah menjadi matriks:

$$\mathbf{K}_V = \begin{bmatrix} 74 & 105 & 108 \\ 117 & 74 & 105 \\ 108 & 117 & 74 \end{bmatrix}$$

Karena menggunakan tabel konversi ASCII Gambar 3.3 nomor 32 hingga 127. Setiap entri matriks dikurangi 32 menjadi:

$$\mathbf{K}_V = \begin{bmatrix} 42 & 73 & 76 \\ 85 & 42 & 73 \\ 76 & 85 & 42 \end{bmatrix}$$

Menggunakan persamaan (3.2) dilakukan dekripsi pada modifikasi *Hill cipher*

yang diperkuat dengan *Vigenère cipher*, yaitu:

$$
\begin{aligned}
P &= \mathbf{K}_H^{-1}(C - K_V) \bmod m \\
P &= \begin{bmatrix} 32816 & -2153 & 13285 \\ 45982 & -2015 & 2254 \\ -6100 & 12420 & 299 \end{bmatrix} \begin{bmatrix} 15 & 92 & 18 \\ 88 & 32 & 12 \\ 21 & 12 & 85 \end{bmatrix} - \begin{bmatrix} 42 & 73 & 76 \\ 85 & 42 & 73 \\ 76 & 85 & 42 \end{bmatrix} (\text{mod}95) \\
&= \begin{bmatrix} 32816 & -2153 & 13285 \\ 45982 & -2015 & 2254 \\ -6100 & 12420 & 299 \end{bmatrix} \begin{bmatrix} -27 & 19 & -58 \\ 3 & -10 & -61 \\ -55 & -73 & 43 \end{bmatrix} (\text{mod}95) \\
&= \begin{bmatrix} -16232 & -3248 & -12008 \\ -13715 & 7293 & -24471 \\ 2184 & -2183 & -4167 \end{bmatrix} (\text{mod}95) \\
&= \begin{bmatrix} 48 & 67 & 65 \\ 69 & 79 & 65 \\ 82 & 66 & 78 \end{bmatrix}
\end{aligned}
$$

Karena menggunakan tabel konversi ASCII Gambar 3.3 nomor 32 hingga 127. Setiap entri matriks ditambah 32 menjadi:

$$
\mathbf{K}_V = \begin{bmatrix} 80 & 99 & 97 \\ 101 & 111 & 97 \\ 114 & 98 & 110 \end{bmatrix}
$$

didapatkan *plaintext* adalah 80 101 114 99 111 98 97 97 110 dan dikonversi kedalam bentuk teks menjadi Percobaan.

BAB IV

PENUTUP

4.1 Kesimpulan

Berdasarkan pembahasan maka kesimpulan yang dapat diambil dari penulisan skripsi ini adalah sebagai berikut: Algoritma untuk menyandi data dengan metode modifikasi *Hill cipher* yang diperkuat dengan *Vigenère cipher* pada GUIDE MATLAB sebagai berikut:
Algoritma Enkripsi:

- Tentukan *plaintext*.
- Tentukan matriks kunci 3x3 untuk *Hill cipher*.
- Tentukan kunci untuk *Vigenère cipher*.
- Gunakan fungsi enkripsi.

$$C = (\mathbf{K}_H.P + K_V)(\mathrm{mod}m)$$

Algoritma Dekripsi:

- Masukkan *ciphertext*.
- Masukkan matriks kunci 3x3 untuk *Hill cipher*.
- Masukkan kunci untuk *Vigenère cipher*.
- Gunakan fungsi dekripsi.

$$P = \mathbf{K}_H^{-1}.(C - K_V) \bmod m$$

4.2 Saran

Berikut beberapa saran yang dapat diuraikan adalah sebagai berikut :

1. Pada Proses *Hill cipher*, matriks kunci yang digunakan dapat diperbesar lebih dari 3x3.
2. Penggunaan syarat kunci *Hill cipher* dengan determinan tidak sama dengan nol (harus invertible), dapat diganti dengan kunci yang noninvertible.
3. Modifikasi *Hill cipher* atau *Vigenere cipher* dengan *Affine cipher* atau *cipher* klasik yang lain.

DAFTAR PUSTAKA

Anton, H. dan Cris Rorres 1987. *Aljabar Linear Elementer Edisi kelima*. Jakarta : Erlangga.

Anton, H. dan Cris Rorres 2005. *Aljabar Linear Elementer Versi Aplikasi Edisi kedelapan*. Jakarta : Erlangga.

Ariyus, Dony. 2006. *Kriptografi: Keamanan Data dan Komunikasi*. Yogyakarta : Penerbit Graha Ilmu.

Away, Guunaidi Adbia. 2014. *The Shortcut of Matlab Programming edisi Revisi*. Bandung : Penerbit Informatika.

Burton, David M. 1980. *Elementary Number Theory*. Boston : Allyn and Bacon, Inc.

Juliadi, Bayu Prihandono dan Nilamsari Kusumastuti. 2013. *Kriptografi Klasik dengan Metode Modifikasi Affine Cipher yang Diperkuat dengan Vigenere Cipher*. Buletin Ilmiah Mat.Stat. dan Terapanya (Bimaster) Vol 02 No. 2 hal 87-92

Kromodimoeljo, Sentot. 2010. *Teori dan aplikasi Kriptografi*. SPK IT Consulting.

Kurniawan, Yusuf. 2005. *Kriptografi: Keamanan Internet dan Jaringan Komunikasi*. Bandung: Penerbit Informatika.

Menezes, Alfred J., Paul C Van Oorscot dan Scott A. Vanstone. 1996. *Handbook of Applied Cryptography*. USA: CRC Press,Inc.

Puspita, Nike Prima dan Nurdin Bahtiar. 2013. *Kriptografi Hill cipher Dengan Menggunakan Operasi Matriks*. Semarang: Universitas Diponegoro

Rahman, M.Noordin, A.F.A Abidin, dan Mohd KAmir Yusuf. 2013. *A New Approach of Classical Hill Cipher*. International Journal of Security and Its Applications Vol 7 No. 2

Rosen, Kenneth H. 2005. *Elemetary Number Theory and Its Applications fifth edition*. New York : ATA&T Laboratories.

Schneier, Bruce 1996. *Applied Cryptography, Second Edition: Protocol, Algorithms and Source Code in C*. John Wiley and Sons,Inc.

Stallings, William. 2006. *Cryptography and Network Security Principles and Practice fifth Edition*. USA :Pearson Education,Inc.

Stinson, Douglas R. 2006. *Cryptography Theory and Practice Third Edition*. Florida :CRC Press,Inc.

Sukirman. 1986. *Ilmu Bilangan*. Jakarta :Karunia.

LAMPIRAN-LAMPIRAN

LAMPIRAN 1

List Program untuk fungsi menggunakan Matlab

1. Fungsi mencari nilai adjoin matriks

```
function adj=adjoin(k)
adj=eye(3)*det(k)*inv(k);
```

2. Fungsi untuk mengubah string menjadi matriks

```
function p=string2matr(x)
% x adalah teks yang diinput oleh user
x=uint8(x);
n=length(x);
p1=ceil(n/3);
%jika banyak x bukan kelipatan dari kunci, maka tambahkan karakter sebarang
%(ditampahkan karakter spasi dengan kode ASCII 32)
if mod(n,3)==0;
    x=[x];
elseif mod(n,3)==1;
    x=[x 32 32];
else mod(n,3)==2;
    x=[x 32];
end
%mengubah teks yang dimasukkan user menjadi matriks ukuran 3x3
p=reshape(x,3,p1);
```

3. Fungsi untuk merepeat kunci Vigenere

```
function k2=repkey(k,x)
n = length(x);
m = length(k);

repk = k(mod(0:n-1,m)+1);
k2=repk;
```

4. Fungsi Algotima Euclide yang diperluas

```
function y = invmodn(z1,m);
%Algoritma Euclide yang diperluas, untuk menghitung invers matriks kunci
n0=m;
b0=z1;
t0=0;
t=1;
q=floor(n0/b0);
r=n0-q*b0;
while r>0,
 temp=t0-q*t;
 temp=mod(temp,m);
 t0=t;
 t=temp;
 n0=b0;
 b0=r;
 q=floor(n0/b0);
 r=n0-q*b0;
end;
 y=mod(t,m);
```

LAMPIRAN 2

List Program Hill Cipher menggunakan Matlab

1. Fungi untuk Enkripsi Hill Cipher

```
function c=enkh(k,x)
if det(k)==0;
    c=disp('Masukkan kunci hill (key 1) yang lain')
    else
    det(k)~=0;
    z1=mod(det(k),95);
    z=gcd(z1,95);
    if z~=1;
        d=disp('masukkan kunci hill (key 1) yang lain');
        else z==1;
    p=string2matr(x);
    j=double(p)-32;
    c1=mod(k*j,95);
    c=reshape(c1,1,numel(p));
    end
end
c=c+32;
c=char(c);
```

2. Fungi untuk deksripsi pada Hill Cpher

```
function d=dekh(k,x)
if det(k)==0;
    d=disp('masukkan kunci hill (key 1) yang lain');
    else
    det(k)~=0;
    z1=mod(det(k),95);
    z=gcd(z1,95);
    if z~=1;
        d=disp('masukkan kunci hill (key 1) yang lain');
    else z==1;
        w=invmodn(z1,95)
        k1=mod(w*adjoin(k),95)
    p=string2matr(x)
    g=double(p)-32
    d1=mod(k1*g,95)
    end
end
d=reshape(d1,1,numel(p))
i=d+32
d=char(int32(i))
```

LAMPIRAN 3

List Program Vigenere Cipher dengan menggunakan Matlab

1. Fungsi untuk Enkripsi Vigenere Cipher

```
function ev=enkv(kv,c)
m=95;
k2=repkey(kv,c);
p=string2matr(c);
pv=p-32;
kv=string2matr(k2);
k3=kv-32;
c1=double(pv)+double(k3);
c2=mod(c1,m);
c3=c2+32;
c4=reshape(c3,1,numel(p));
ev=char(c4);
```

2. Fungsi untuk Dekripsi Pada Vigenere Cipher

```
function ev=enkv(kv,c)
m=95;
k2=repkey(kv,c);
p=string2matr(c);
pv=p-32;
kv=string2matr(k2);
k3=kv-32;
c1=double(pv)+double(k3);
c2=mod(c1,m);
c3=c2+32;
c4=reshape(c3,1,numel(p));
ev=char(c4);
```

LAMPIRAN 4

List Program Hill - Vigenere Cipher dengan menggunakan Matlab

1. Fungsi untuk Enkripsi HHill - Vigenere Cipher

```
function hv=enk(kh,kv,x)
a=enkh(kh,x);
a=uint8(a);
b=enkv(kv,a);
hv=b;
```

2. Fungsi untuk Dekripsi Hill - Vigenere Cipher

```
function vh=dek(kh,kv,x)
a=dekv(kv,x);
a=uint8(a);
b=dekh(kh,a);
vh=b;
```

LAMPIRAN 5

List Program Hill - Vigenere Cipher dengan menggunakan GUIDE Matlab

```
function varargout = Sandi(varargin)
% SANDI M-file for Sandi.fig
%      SANDI, by itself, creates a new SANDI or raises the existing
%      singleton*.
%
%      H = SANDI returns the handle to a new SANDI or the handle to
%      the existing singleton*.
%
%      SANDI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in SANDI.M with the given input arguments.
%
%      SANDI('Property','Value',...) creates a new SANDI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Sandi_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Sandi_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 Sandi

% Last Modified by GUIDE v2.5 17-Jun-2015 13:43:20

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Sandi_OpeningFcn, ...
                   'gui_OutputFcn',  @Sandi_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 Sandi is made visible.
function Sandi_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 Sandi (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

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


% --- Outputs from this function are returned to the command line.
function varargout = Sandi_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;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

myform = guidata(gcbo);

a11=str2double(get(myform.EdA11,'String'));
a12=str2double(get(myform.EdA12,'String'));
a13=str2double(get(myform.EdA13,'String'));
a21=str2double(get(myform.EdA21,'String'));
a22=str2double(get(myform.EdA22,'String'));
a23=str2double(get(myform.EdA23,'String'));
a31=str2double(get(myform.EdA31,'String'));
a32=str2double(get(myform.EdA32,'String'));
a33=str2double(get(myform.EdA33,'String'));

kh=[a11 a12 a13; a21 a22 a23; a31 a32 a33];

kv=get(myform.EdKv,'String');

x=get(myform.EdX,'String');

if det(kh)==0;
    set(myform.EdY,'String','Masukkan kunci Hill (key 1) yang lain');
    else
    det(kh)~=0;
```

```
    z1=mod(det(kh),95);
    z=gcd(z1,95);
    if z~=1;
       set(myform.EdY,'String','Masukkan kunci Hill (key 1) yang lain');
    else z==1;
       x=uint8(x);
       n=length(x);
       p1=ceil(n/3);
    if mod(n,3)==0;
        x1=[x];
        elseif mod(n,3)==1;
        x1=[x 32 32];
        else mod(n,3)==2;
        x1=[x 32];
    end
    end
    x2=reshape(x1,3,p1);

    x3=double(x2)-32;
    x4=mod(kh*x3,95);
    x5=reshape(x4,1,numel(x1));
end
x6=x5+32;
x7=char(x6);


n1 = length(x7);
m = length(kv);

repkv = kv(mod(0:n1-1,m)+1);
kv1=repkv;

x7=uint8(x7);

x8=reshape(x7,3,p1);

x9=x8-32;
kv2=uint8(kv1);

kv3=reshape(kv2,3,p1);

kv4=kv3-32;
c1=double(x9)+double(kv4);
c2=mod(c1,95);
c3=c2+32;
c4=reshape(c3,1,numel(kv3));
y=char(c4);

set(myform.EdY,'String',y);


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
```

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

myform = guidata(gcbo);

a11=str2double(get(myform.EdA11,'String'));
a12=str2double(get(myform.EdA12,'String'));
a13=str2double(get(myform.EdA13,'String'));
a21=str2double(get(myform.EdA21,'String'));
a22=str2double(get(myform.EdA22,'String'));
a23=str2double(get(myform.EdA23,'String'));
a31=str2double(get(myform.EdA31,'String'));
a32=str2double(get(myform.EdA32,'String'));
a33=str2double(get(myform.EdA33,'String'));

kh=[a11 a12 a13; a21 a22 a23; a31 a32 a33];

kv=get(myform.EdKv,'String');

y=get(myform.EdX,'String');

n = length(y);
m = length(kv);

repk = kv(mod(0:n-1,m)+1);
kv1=repk;


y=uint8(y);

p1=ceil(n/3);

if mod(n,3)==0;
    y=[y];
elseif mod(n,3)==1;
    y=[y 32 32];
else mod(n,3)==2;
    y=[y 32];
end

y1=reshape(y,3,p1);



y2=y1-32;

kv1=uint8(kv1);

kv2=reshape(kv1,3,p1);

kv3=kv2-32;
y3=double(y2)-double(kv3);
y4=mod(y3,95);
y5=y4+32;
```

```
y6=reshape(y5,1,numel(y1));
y7=char(y6);


if det(kh)==0;
set(myform.EdY,'String','Masukkan kunci Hill (key 1) yang lain');
else
    det(kh)~=0;
    z1=mod(det(kh),95);
    z=gcd(z1,95);
    if z~=1;
    set(myform.EdY,'String','Masukkan kunci Hill (key 1) yang lain');
    else z==1;

n0=95;
b0=z1;
t0=0;
t=1;
q=floor(n0/b0);
r=n0-q*b0;
while r>0,
 temp=t0-q*t;
 temp=mod(temp,95);
 t0=t;
 t=temp;
 n0=b0;
 b0=r;
 q=floor(n0/b0);
 r=n0-q*b0;
end;
 z2=mod(t,95);

kh1=eye(3)*det(kh)*inv(kh);

        kh2=mod(z2*kh1,95);

y7=uint8(y7);

y8=reshape(y7,3,p1);


    y9=double(y8)-32;
    d1=mod(kh2*y9,95);
    end
end
d2=reshape(d1,1,numel(y8));
i=d2+32;
d3=char(int32(i));
x=d3;

set(myform.EdY,'String',x);


% --- Executes on button press in pushbutton3.
```

```
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

myform = guidata(gcbo);

selection=questdlg(['Anda yakin ingin mereset semua data ini',' ?'],...
['bertanya' '' '...'],...
'Ya','Batal','Ya');
if strcmp(selection,'Batal')
return
end
set(myform.EdA11, 'String', '');
set(myform.EdA12, 'String', '');
set(myform.EdA13, 'String', '');
set(myform.EdA21, 'String', '');
set(myform.EdA22, 'String', '');
set(myform.EdA23, 'String', '');
set(myform.EdA31, 'String', '');
set(myform.EdA32, 'String', '');
set(myform.EdA33, 'String', '');
set(myform.EdKv, 'String', '');
set(myform.EdX, 'String', '');
set(myform.EdY, 'String', '');
%set(handles.enk, 'Enable', 'off');
%set(handles.dek, 'Enable', 'off');
%set(handles.enk, 'Value', 0);
%set(handles.dek, 'Value', 0);

% --------Kode untuk validasi data (reset)
function initialize_gui(fig_handle, handles, isreset)
if isfield(handles, 'metricdata') && ~isreset
return;
end
guidata(myform.Sandi, handles);



% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

selection=questdlg(['Anda Yakin Ingin Menutup Aplikasi',' ?'],...
['bertanya' '' '...'],...
'Ya','Batal','Ya');
if strcmp(selection,'Batal')
return
end
close;

function EdA11_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to EdA11 (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 EdA11 as text
%        str2double(get(hObject,'String')) returns contents of EdA11 as a double


% --- Executes during object creation, after setting all properties.
function EdA11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to EdA11 (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,'defaultUicontrolBackgrou
    set(hObject,'BackgroundColor','white');
end



function a12_Callback(hObject, eventdata, handles)
% hObject    handle to a12 (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 a12 as text
%        str2double(get(hObject,'String')) returns contents of a12 as a double


% --- Executes during object creation, after setting all properties.
function a12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to a12 (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,'defaultUicontrolBackgrou
    set(hObject,'BackgroundColor','white');
end



function a13_Callback(hObject, eventdata, handles)
% hObject    handle to a13 (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 a13 as text
%        str2double(get(hObject,'String')) returns contents of a13 as a double
```

```
% --- Executes during object creation, after setting all properties.
function a13_CreateFcn(hObject, eventdata, handles)
% hObject    handle to a13 (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,'defaultUicontrolBackgrou
    set(hObject,'BackgroundColor','white');
end



function EdA21_Callback(hObject, eventdata, handles)
% hObject    handle to EdA21 (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 EdA21 as text
%        str2double(get(hObject,'String')) returns contents of EdA21 as a double


% --- Executes during object creation, after setting all properties.
function EdA21_CreateFcn(hObject, eventdata, handles)
% hObject    handle to EdA21 (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,'defaultUicontrolBackgrou
    set(hObject,'BackgroundColor','white');
end



function a22_Callback(hObject, eventdata, handles)
% hObject    handle to a22 (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 a22 as text
%        str2double(get(hObject,'String')) returns contents of a22 as a double


% --- Executes during object creation, after setting all properties.
function a22_CreateFcn(hObject, eventdata, handles)
% hObject    handle to a22 (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,'defaultUicontrolBackgrou
    set(hObject,'BackgroundColor','white');
end



function EdA23_Callback(hObject, eventdata, handles)
% hObject    handle to EdA23 (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 EdA23 as text
%        str2double(get(hObject,'String')) returns contents of EdA23 as a double


% --- Executes during object creation, after setting all properties.
function EdA23_CreateFcn(hObject, eventdata, handles)
% hObject    handle to EdA23 (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,'defaultUicontrolBackgrou
    set(hObject,'BackgroundColor','white');
end



function a31_Callback(hObject, eventdata, handles)
% hObject    handle to a31 (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 a31 as text
%        str2double(get(hObject,'String')) returns contents of a31 as a double


% --- Executes during object creation, after setting all properties.
function a31_CreateFcn(hObject, eventdata, handles)
% hObject    handle to a31 (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,'defaultUicontrolBackgrou
    set(hObject,'BackgroundColor','white');
end



function a32_Callback(hObject, eventdata, handles)
% hObject    handle to a32 (see GCBO)
```

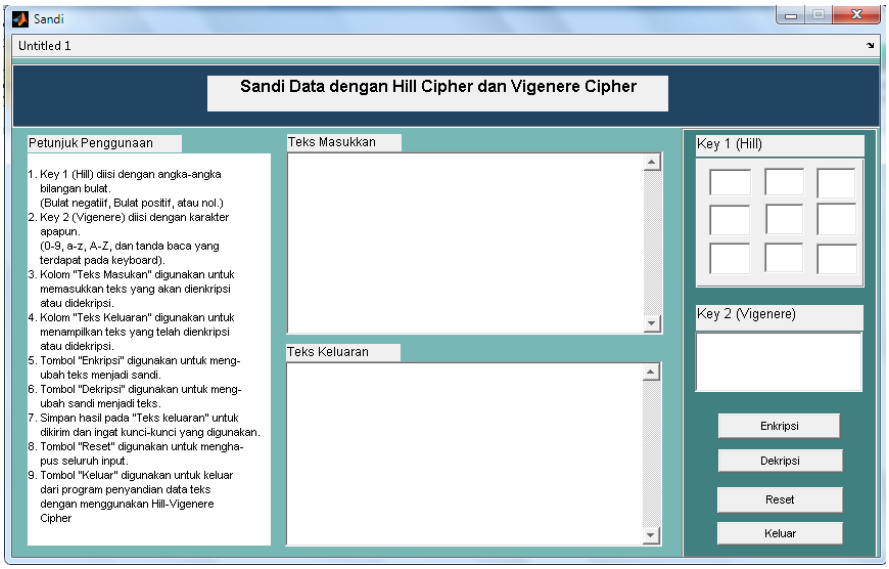
```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of a32 as text
%        str2double(get(hObject,'String')) returns contents of a32 as a double


% --- Executes during object creation, after setting all properties.
function a32_CreateFcn(hObject, eventdata, handles)
% hObject    handle to a32 (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,'defaultUicontrolBackgrou
    set(hObject,'BackgroundColor','white');
end



function a33_Callback(hObject, eventdata, handles)
% hObject    handle to a33 (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 a33 as text
%        str2double(get(hObject,'String')) returns contents of a33 as a double


% --- Executes during object creation, after setting all properties.
function a33_CreateFcn(hObject, eventdata, handles)
% hObject    handle to a33 (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,'defaultUicontrolBackgrou
    set(hObject,'BackgroundColor','white');
end



function EdKv_Callback(hObject, eventdata, handles)
% hObject    handle to EdKv (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 EdKv as text
%        str2double(get(hObject,'String')) returns contents of EdKv as a double


% --- Executes during object creation, after setting all properties.
```

```
function EdKv_CreateFcn(hObject, eventdata, handles)
% hObject    handle to EdKv (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,'defaultUicontrolBackgrou
    set(hObject,'BackgroundColor','white');
end



function x_Callback(hObject, eventdata, handles)
% hObject    handle to x (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 x as text
%        str2double(get(hObject,'String')) returns contents of x as a double


% --- Executes during object creation, after setting all properties.
function x_CreateFcn(hObject, eventdata, handles)
% hObject    handle to x (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,'defaultUicontrolBackgrou
    set(hObject,'BackgroundColor','white');
end



function EdY_Callback(hObject, eventdata, handles)
% hObject    handle to EdY (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 EdY as text
%        str2double(get(hObject,'String')) returns contents of EdY as a double


% --- Executes during object creation, after setting all properties.
function EdY_CreateFcn(hObject, eventdata, handles)
% hObject    handle to EdY (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,'defaultUicontrolBackgrou
```

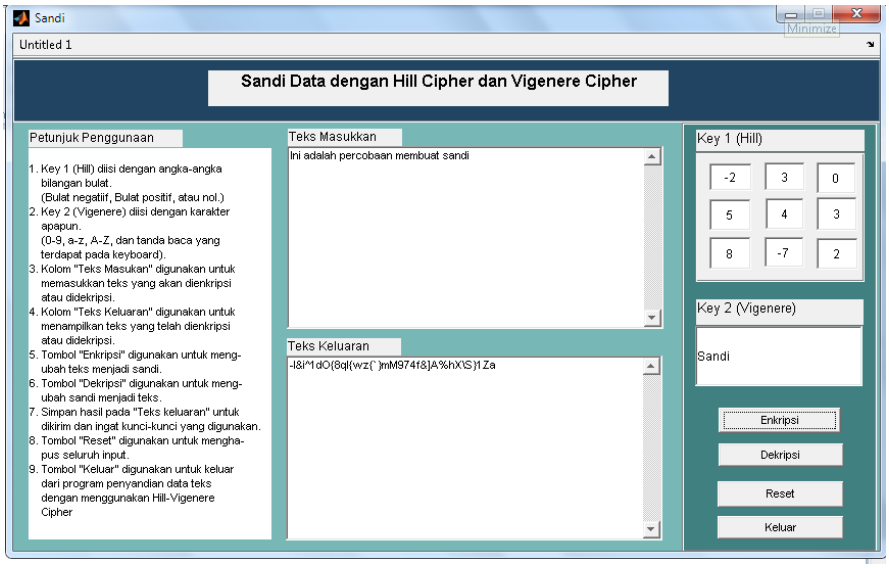
```
    set(hObject,'BackgroundColor','white');
end



% --------------------------------------------------------------------
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 EdX_Callback(hObject, eventdata, handles)
% hObject    handle to EdX (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 EdX as text
%        str2double(get(hObject,'String')) returns contents of EdX as a double
```

LAMPIRAN 6

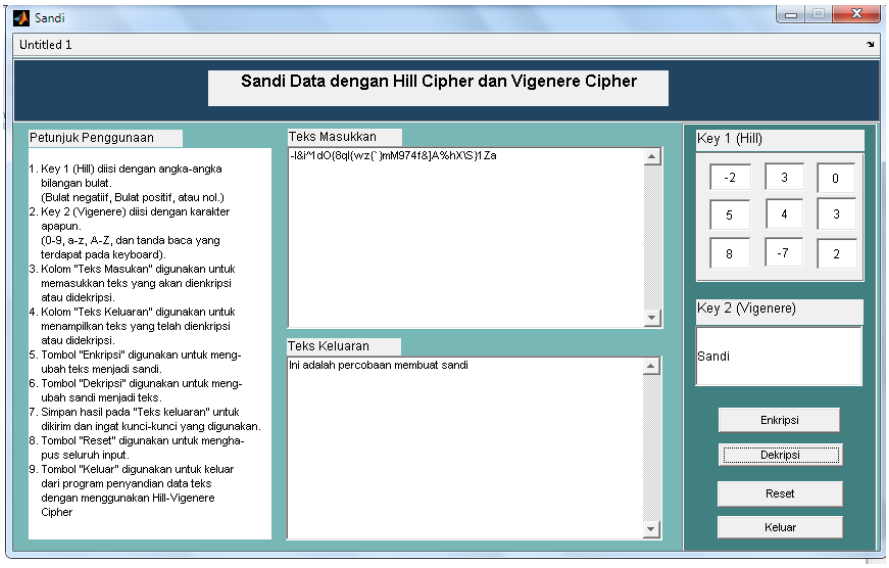
Tampilan implementasi algoritma menggunakan Matlab



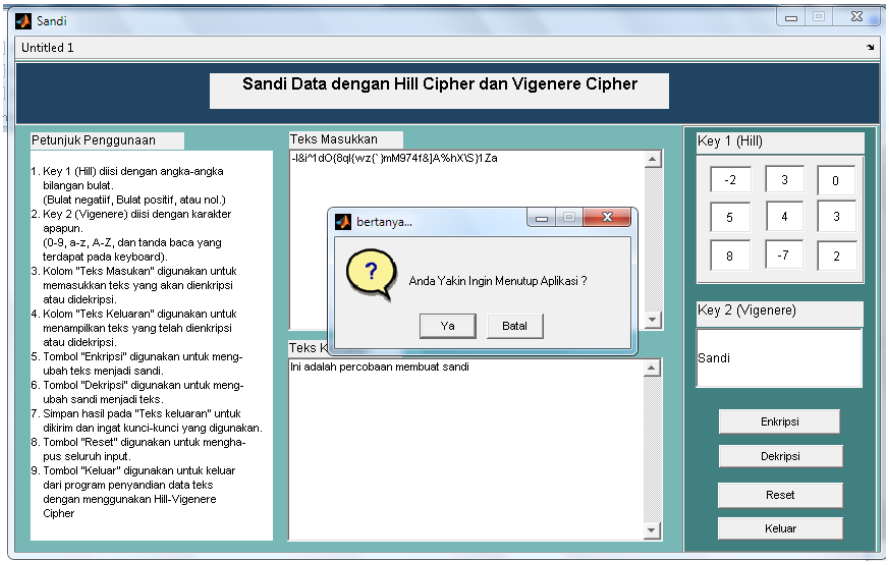
Gambar 4.1: Tampilan awal program "Sandi"



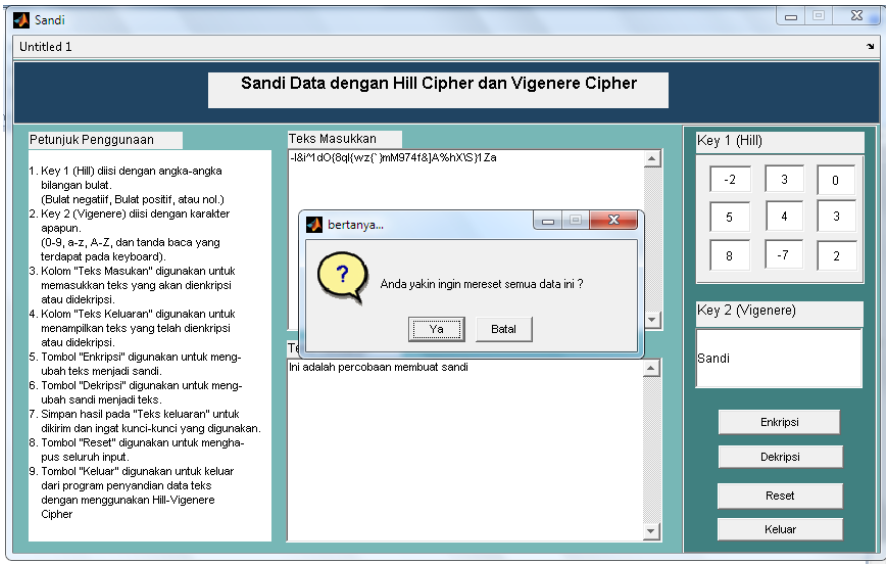
Gambar 4.2: Percobaan Enkripsi Sandi Hill-Vigenere



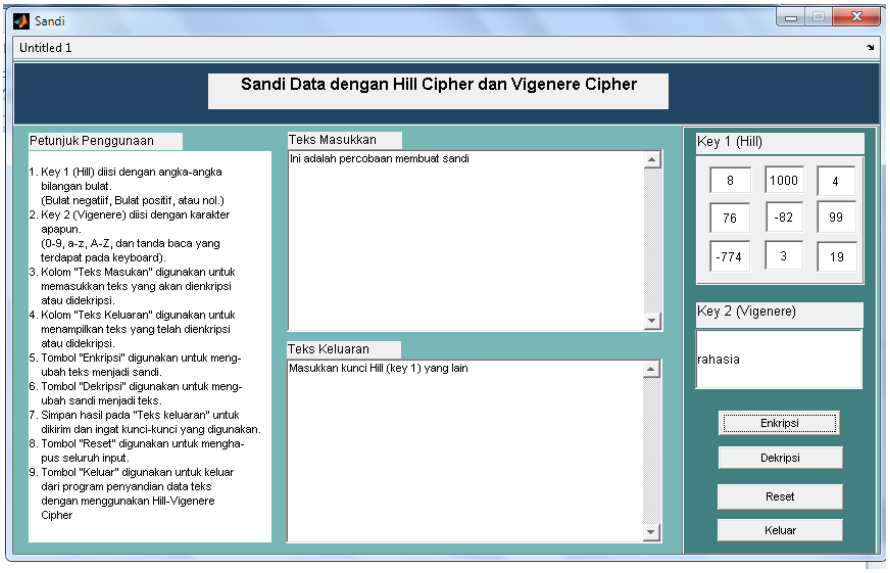
Gambar 4.3: Percobaan Dekripsi Sandi Hill-Vigenere



Gambar 4.4: Percobaan menggunakan tombol "Keluar"



Gambar 4.5: Percobaan menggunakan tombol "Reset"



Gambar 4.6: Tampilan program jika terdapat kesalahan dalam memilih kunci

SURAT PERNYATAAN KEASLIAN SKRIPSI

Dengan ini saya yang bertanda tangan di bawah ini, mahasiswa Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Jakarta:

Nama	:	Khoiruz Zahra
No. Registrasi	:	3125081768
Jurusan	:	Matematika
Program Studi	:	Matematika

Menyatakan bahwa skripsi ini yang saya buat dengan judul "**Sandi Data Menggunakan Kriptografi Klasik *Hill Cipher* yang Diperkuat dengan *Vigenère Cipher***" adalah:

1. Dibuat dan diselesaikan oleh saya sendiri.
2. Bukan merupakan duplikat skripsi yang pernah dibuat oleh orang lain atau jiplakan karya tulis orang lain.

Pernyataan ini dibuat dengan sesungguhnya dan saya bersedia menanggung segala akibat yang timbul jika pernyataan saya tidak benar.

Jakarta, Juli 2015

Yang membuat pernyataan

Khoiruz Zahra

DAFTAR RIWAYAT HIDUP

Khoiruz Zahra dilahirkan di Jakarta pada tanggal 19 April 1990. Anak ke-6 dari 11 bersaudara dari pasangan Bapak Rahmat (alm) dan Ibu Siti Warti.

Riwayat Pendidikan

Pendidikan formal yang pernah ditempuh adalah SD Negeri 04 Petang Kayu Putih Jakarta Timur dan lulus tahun 2002. Pada tahun yang sama masuk SMP Negeri 99 Jakarta Timur dan lulus tahun 2005 kemudian melanjutkan ke SMA Negeri 31 Jakarta Timur dan lulus tahun 2008. Pada tahun yang sama diterima di Universitas Negeri Jakarta, Fakultas Matematika dan Ilmu Pengetahuan Alam, Jurusan Matematika, Program Studi Matematika.

Bagi pembaca yang ingin berkomunikasi dengan penulis dapat mengakses lewat email di `zee_blumme@yahoo.com`.