ALGORITMA: Jurnal Ilmu Komputer dan Informatika
Volume: 01, Number : 01, November 2017 ISSN 2598-6341 (online

Perancangan Perangkat Lunak Pengenkripsian Citra BMP,GIF dan JPG dengan Menggunakan Metode HILL

Abdul Halim Hasugian, M.Kom
Universitas Islam Negeri Sumatera Utara Medan

Abdulhalim0388@gmail.com

Abstrak

Citra digital dapat didefinisikan sebagai fungsi dua variabel, f(x,y), *dimana* x *dan y adalah koordinat spasial dan nilai* f(x,y) *adalah intensitas citra pada koordinat tersebut. Citra digital menyatakan data citra dalam angka yang mewakili aras keabuan (citra hitam putih) atau koordinat warna (citra berwarna).Permasalahan dalam sistem ini adalah bagaimana mengimplementasikan kriptografi citra digital dengan menggunakan metoda Hill Chiper sehingga dapat menjamin keamanan dari citra digital tersebut daripihak-pihak yang tidak berhak untuk mengetahuinya atau melakukan perubahan dalam citra digital tersebut.Kriptografi citra digital ini dirancang menggunakan metode Hill Chiper diimplementasikan dengan bahasa pemograman Visual Basic 6.0. Metode Hill Chiper ini merupakan metode Hill Cipher merupakan salah satu algoritma kriptografi kunci simetris. Algoritma Hill Cipher menggunakan matriks berukuran m x m sebagai kunci untuk melakukan enkripsi dan dekripsi. Dasar teori matriks yang digunakan dalam Hill Cipher antara lain adalah perkalian antar matriks dan melakukan invers pada matriks. Kriptografi citra digital ini menghasilakn sebuah citra digital yang kabur dan tidak dapat dilihat dengan kasat mata tanpa bantuan perangkat lunak yang penulis rancang, serta mengubah besar ukuran file citra digital tersebut dengan cara memetakan nilai RGB pada masing-masing pixel citra digital tersebut menjadi bentuk sebuah geometri. Namun, sistem ini hanya dapat mengenkripsi citra digital dengan format JPG, BMP, dan GIF sehingga masih membutuhkan pengembangan lebih lanjut.*

Kata Kunci : Kriptografi, Citra Digital dan Metode Hill Chiper

Abstract

Digital image can be defined as a function of two variables, f (x, y), where x and y are spatial coordinates and the value of f (x, y) is the image intensity at the coordinates. Digital images represent image data in numbers representing the level of gray (color black and white) or color coordinates (color image). The problem in this system is how to implement digital image cryptography using Hill Chiper method so as to ensure the security of the digital image from the parties which is not entitled to know it or make changes in the digital image. This digital image cryptography is designed using Hill Chiper method implemented with Visual Basic 6.0 programming language. Hill Chip method is a Hill Cipher method is one of the symmetric key cryptography algorithms. Hill Cipher algorithm uses an m x m matrix as the key for encryption and decryption. The basic matrix theory used in Hill Cipher is inter-matrix multiplication and inverse in matrix. This digital image cryptography produces an obscure and invisible digital image without the help of the software that the author designs, and changes the size of the digital image file by mapping the RGB value of each pixel of the digital image into a geometry . However, this system can only encrypt digital imagery with JPG, BMP, and GIF formats so it still needs further development.

Keywords*: Cryptography, Digital Imagery and Hill Chiper Method*

1. Pendahuluan

File citra sebagai salah satu bentuk data digital saat ini banyak dipakai untuk menyimpan photo, gambar, ataupun hasil karya dalam format digital. Bila *file-file* tersebut tidak diamankan sewaktu dikirimkan ke jaringan terbuka seperti Internet, *file* tersebut dapat jatuh ke pihak yang tidak diinginkan, yang kemudian disalah gunakan untuk hal-hal bersifat negatif. Perkembangan teknologi komputer yang semakin canggih serta terbuka jaringan yang menghubungkan satu komputer dengan komputer lain memungkinkan pertukaran data tidak mengenal waktu dan tempat lagi. Sisi buruk dari ini adalah data yang penting dapat dengan mudah jatuh ke tangan orang yang tidak bertanggung jawab. Si A ingin mengirimkan suatu *file* citra rahasia kepada temannya si B dan agar *file* tersebut tidak jatuh ke tangan si C maka baik A dan B harus memikirkan suatu cara untuk menyandikan *file* tersebut dan hanya A dan B yang dapat mengembalikan citra yang disandikan tersebut.

Salah satu cara untuk mengatasi hal tersebut adalah menyandikan citra tersebut sehingga bentuk

citra menjadi teracak. Dan apabila jatuh ke tangan yang tidak diinginkan, citra tersebut juga tidak dapat digunakan. Salah satu metode penyandian untuk tujuan di atas adalah menggunakan teknik penyandian dengan metode *Hill Cipher*. *Hill Cipher* sebenarnya merupakan salah satu teknik penyandian teks, tetapi dengan melakukan perubahan perhitungan pada nilai RGB (*Red Green Blue*) citra maka *Hill Cipher* juga dapat dipakai untuk menyandikan citra. *Hill Cipher* menggunakan matriks persegi sebagai kunci dalam proses penyandiannya. Dengan pemilihan matriks kunci yang baik, *Hill Cipher* dapat dipakai untuk penyandian karena hanya melibatkan operasi matriks biasa sehingga prosesnya relatif cepat.

2. Tinjauan *Hill*

2.1 Pengertian *Hill*

Perancangan dan pembuatan suatu perangkat lunak yang dapat melakukan penyandian citra dengan *Hill Cipher*. Matriks kunci yang dapat dipakai mencakup matriks persegi dengan ordo 2 × 2 atau 3 × 3. *File* citra hasil penyandian ini dapat dicetak dan disimpan kembali dalam format *bitmap*. Pada tulisan ini juga dilakukan pengujian pada *file* citra jenis photo dan jenis kartun agar diketahui jenis citra apa yang mampu disandikan dengan *Hill Cipher*.banyak aplikasi yang memungkinkan kita untuk menerapkan prinsip metode *Hill Cipher.*

Hill Cipher berdasarkan pada aljabar linier dan seperti sandi Vigenére, Hill Cipher merupakan *block cipher*. Sandi ini dapat dipecahkan dengan *known-plaintext attacks* tetapi tahan melawan *ciphertext-only attack*. Cara kerja sandi ini berdasarkan atas perkalian matriks dengan menggnakan sebuah kunci K. Penjelasan mengenai Hill Cipher ini dapat diuraikan sebagai berikut:

Misalkan m adalah bilangan bulat positif dan P = C = $(Z_{26})^m$ dan minsalkan K={m x m merupakan matriks yang nilai elemennya terdir dari Z_{26}} maka untuk suatu kunci K, dapat di defenisikan sebagai $e_K(x)$ = K x Mod 26 dan $d_x(y) = K^{-1}$ y Mod 26 dimana semua operasi dilakukan dalam matrik Z_{26} . Karena K^{-1} dengan mudah dapat dihitung dari K, maka Hill Cipher merupakan suatu kriptosistem asimetrik. Hill Cipher juga merupakan blok *cipher* linier umum. Suatu blok *cipher* linier dapat dengan mudah dipecahkan yang dikenal cara *known-plaintext attacks*. Maka bagi penyerang yang mengetahui beberapa contoh *plaintext* dengan enkripsi yang berhubungan, tidaklah sulit baginya untuk mencari kunci yang dipakai untuk mengenkripsi *plaintext* tersebut. Bahkan nkripsi dengan teks yang tidak diketahui dapat didekripsi tanpa harus memerlukan usaha yang sulit. Metode dari perhitungan frekuensi sering dipakai untuk usaha ini. Metode ini mengeksplorasi perulangan (*redundancy*) dari bahasa alami yang dipakai sebagai *plaintext* pada pesan.

Sebagai contoh, pada banyak bahasa huruf “E” sering muncul dengan persentasi dalam bahasa Inggris mencapai 12,31%, 15,87% untuk bahasa Perancis, dan bahkan 18,46% dalam bahasa Jerman. Penjelasan cara kerja dari Hill Cipher dapat disederhanakan dengan cara seperti ini. Misalkan K merupakan sebuah matriks kunci *m – m* yang merupakan representasi dari suatu persamaan linier. *Ciphertext* (C) dan *plaintext* p merupakan matriks *m* □ 1. Maka didapat persamaan untuk menghasilkan *ciphertext* sebagai berikut:

C=K.P (mod 26)

$$\begin{pmatrix} C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \end{pmatrix} \bmod 26$$

Dekripsi memerlukan kunci K yang bersifat *invertible*. Contohnya K . K^{-1} (mod 26) = I di mana I merupakan matriks identitas.

Karena C = K . P Mod 26 maka K = C . P^{-1} Mod 26 Tidak semua *plaintext* bersifat *invertible* (dapat dibalik kembali). Sandi Caesar, Hill Cipher, dan sandi Playfair semua bekerja dengan sebuah alphabet tunggal saat disubst itusi.

2.2 Pendekripsian Hill Cipher

Algoritma proses pendekripsian Hill Cipher dapat diuraikan dalam bentuk langkah-langkah sebagai berikut:

1. Hitung matriks K^{-1} (mod 256) sebagai kunci pembuka. K^{-1} ada jika FPB ((det(K), 256)=1.
2. Lakukan langkah-langkah 2–5 pada enkripsi dengan mengganti:
   (i) Matriks K dengan matriks K^{-1}
   (ii) Blok vektor piksel asli P dengan blok vektor sandi C dan sebaliknya.

2.3 Dasar Matematika Penyandian Citra pada *Hill Cipher*

Penyandian dengan Hill Cipher pada citra seperti halnya dengan teks juga menggunakan matriks dimana ketentuannya menggunakan matriks bujur sangkar. Matriks bujur sangkar adalah matriks dengan jumlah baris = jumlah ko lom. Matriks bujur sangkar disebut matriks identitas (I) jika semua elemen diagonal utamanya sama dengan satu dan elemen lainnya sama dengan nol. Invers suatu matriks A adalah matriks B sedemikian hingga A . B = I. Invers matriks A ada jika determinan (A) ≠ 0.

Misal a dan b adalah bilangan-bilangan bulat. Bilangan bulat c disebut faktor persekutuan a dan b jika c|a dan c|b. Bilangan bulat tak negatif d disebut faktor persekutuan terbesar (FPB) a dan b jika d adalah faktor persekutuan a dan b dan untuk setiap c, jika c|a dan c|b maka d|c.

2.4 Algoritma Enkripsi dan Dekripsi Citra Digital dengan *Hill Cipher*

Algoritma proses enkripsi citra digital dengan metode Hill Cipher dapat diuraikan dalam bentuk langkah-langkah sebagai berikut:

1. Tentukan ukuran dan nilai RGB tiap piksel citra digital.
2. Pilih suatu matriks kunci K yang berupa matriks bujur sangkar yang dipakai sebagai kunci.
3. Transformasikan tiap piksel dalam citra digital ke bilangan bulat yang sesuai dengan nilai RGB (0-255)
4. Kelompokkan barisan angka yang didapat ke dalam beberapa blok vektor P yang panjangnya sama dengan ukuran matriks K.
5. Hitung C = K . P (mod 256) untuk tiap vektor P.
1. Kembalikan tiap angka dalam vektor sandi C ke piksel yang sesuai untuk mendapatkan piksel sandi.

Sedangkan algoritma proses pendekripsian citra digital dengan metode Hill Cipher dapat diuraikan dalam bentuk langkah-langkah sebagai berikut:

1. Hitung matriks K^{-1} (mod 256) sebagai kunci pembuka. K^{-1} ada jika FPB ((det(K), 256)=1.
2. Lakukan langkah-langkah 2–5 pada enkripsi dengan mengganti:
   i) Matriks K dengan matriks K^{-1}
   ii) Blok vektor piksel asli P dengan blok vektor sandi C dan sebaliknya.

3.1 Cara Kerja *Hill Cipher* Pada Penyandian Citra

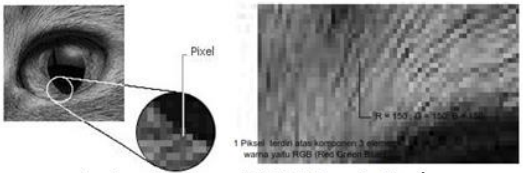
Pada Bagian ini penggunaan *Hill Cipher* diperluas pemakaiannya pada citra baik untuk citra berwarna atau monokrom (hitam-putih). Karena tiap-tiap komponen RGB piksel memiliki panjang 8 bit (0-255), maka sistem modulo yang dipakai dalam penyandian adalah Zn = Z_{256}. Ini mengingat nilai rentang untuk tiap komponen RGB adalah sebesar 0 hingga 255 atau 256. Program ini nantinya dibuat dengan bahasa pemrograman Visual Basic.

Untuk mengenkripsi citra dengan *Hill Cipher*, mula-mula nilai RGB tiap piksel diambil. Adapun algoritma untuk pengambilan nilai RGB yang dilakukan untuk tiap piksel dengan menggunakan fungsi GetPixel adalah sebagai berikut:

```
For i = 0 To PictureWidth - 1
      For j = 0 To PictureHeight - 1
        'GetPixel i, j
        PixelColor = GetPixel(Picture, i, j)
        r = PixelColor Mod 256
        g = (PixelColor \ 256) Mod 256
        b = PixelColor \ 256 \ 256
 Next j
Next i
```

Dimana:

1. PictureWidth = Lebar Citra dalam satuan piksel.
2. PictureHeight = Tinggi Citra dalam satuan piksel.
3. PixelColor = Merupakan hasil pengembalian warna dari fungsi GetPixel dalam nilai *long integer*.
4. r = Merupakan nilai merah (*red*) pada piksel dalam rentang 0 hingga 255.
5. g = Merupakan nilai warna hijau (*green*) pada piksel dalam rentang 0 hingga 255.
6. b = Merupakan nilai warna biru (*blue*) pada piksel dalam rentang 0 hingga 255.
7. i dan j = Merupakan posisi piksel yang dinyatakan dalam sumbu koordinat.



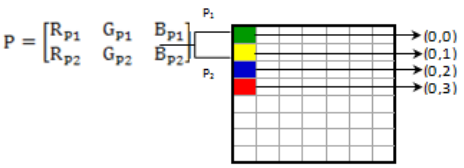
Gambar 3.1 Ilustrasi Nilai RGB pada Piksel

Jika cara di atas diterapkan maka suatu citra dapat dienkripsi dengan *Hill Cipher* seperti halnya dengan teks (ditransformasi dalam nilai ASCII) berdasarkan atas nilai komponen RGB-nya.

Masalah yang dihadapi adalah untuk citra satu piksel warna terdapat tiga buah nilai yang harus dienkripsi. Maka nilai-nilai komponen warna tersebut harus disusun sedemikian rupa agar dapat dikalikan dengan matriks kunci karena perkalian matriks bersifat unik. Matriks tidak dapat dikalikan jika jumlah kolom pada matriks pertama tidak sama dengan jumlah baris pada matriks kedua.

Berdasarkan algoritma untuk mengambil nilai piksel dan mengubahnya menjadi komponen RGB maka dapat diketahui pembacaan dimulai secara per kolom. Artinya dimulai dari baris pertama hingga baris terakhir pada kolom pertama baru

dilanjutkan pada kolom berikutnya. Jika matriks kunci yang dipakai berordo dua maka penyusunan matriks P (matriks yang berisi nilai RGB citra yang akan disandikan) adalah sebagai berikut:



Gambar 3.2 Representasi Nilai Piksel Untuk Matriks 2 × 2

Dimana:
R_{P1} adalah nilai komponen *red* pada *pixel* pertama
G_{P1} adalah nilai komponen *green* pada *pixel* pertama
B_{P1} adalah nilai komponen *blue* pada *pixel* pertama
R_{P2} adalah nilai komponen *red* pada *pixel* kedua
G_{P2} adalah nilai komponen g*reen* pada *pixel* kedua
Jika matriks kunci yang dipakai berordo tiga maka penyusunan matriks P adalah sebagai berikut:

3.2 Contoh Kasus

Agar perhitungan cara kerja *Hill Cipher* pada citra dapat dimengerti dan jelas maka pada bagian ini akan dibahas dengan contoh perhitungan baik dengan matriks 2 × 2 dan matriks 3 × 3.

Berikutnya algoritma enkripsi *Hill Cipher* pada citra diterapkan pada barisan nilai RGB piksel dengan mengambil *n* = 256. Dekripsi juga dilakukan dengan cara yang sama.

Misalkan matriks kunci berordo 2 × 2 atau 3 × 3 dipakai untuk mengenkripsi potongan citra yang mempunyai 6 buah piksel dimana komponennya adalah:

$R_{p1} = 200$ $G_{p1} = 150$ $B_{p1} = 200$	$R_{p2} = 200$ $G_{p2} = 150$ $B_{p2} = 150$	$R_{p3} = 200$ $G_{p3} = 120$ $B_{p3} = 10$
$R_{p4} = 100$ $G_{p4} = 100$ $B_{p4} = 70$	$R_{p5} = 40$ $G_{p5} = 20$ $B_{p5} = 30$	$R_{p6} = 0$ $G_{p6} = 100$ $B_{p6} = 20$

Nilai-nilai RGB tersebut akan diambil dan diproses per tiap baris dengan matriks kunci. Untuk proses dengan matriks kunci 2 × 2 proses akan dilakukan tiap dua piksel per baris citra sedangkan proses dengan matriks kunci 3 × 3 akan dilakukan tiap tiga piksel per baris citra.

Sebagai contoh di atas bila proses dilakukan dengan matriks kunci 2 × 2 maka diperlukan tiga kali proses untuk menghasilkan *output.* Nilai tiap-tiap piksel akan disusun sebagai berikut:

$$P_1 = \begin{pmatrix} R_{p1} & G_{p1} & B_{p1} \\ R_{p2} & G_{p2} & B_{p3} \end{pmatrix};\ P_2 = \begin{pmatrix} R_{p3} & G_{p3} & B_{p3} \\ R_{p4} & G_{p4} & B_{p4} \end{pmatrix};\ P_3 = \begin{pmatrix} R_{p5} & G_{p5} \\ R_{p6} & G_{p6} \end{pmatrix}$$

Sedangkan bila proses dilakukan dengan matriks kunci 3 × 3 maka diperlukan dua kali proses saja untuk menghasilkan *output*. Nilai tiap-tiap piksel akan disusun sebagai berikut:

$$P_1 = \begin{pmatrix} R_{p1} & G_{p1} & B_{p1} \\ R_{p2} & G_{p2} & B_{p2} \\ R_{p3} & G_{p3} & B_{p3} \end{pmatrix};\ P_2 = \begin{pmatrix} R_{p4} & G_{p4} & B \\ R_{p5} & G_{p5} & B \\ R_{p6} & G_{p6} & B \end{pmatrix}$$

Agar lebih jelasnya bagaimana cara perhitungan penyandian dengan *Hill Cipher* maka berikut ini piksel-piksel di atas akan disandikan atau dienkripsikan dengan menggunakan tiga buah matriks berikut yaitu matriks

$2 \times 2 = \begin{pmatrix} 2 & 1 \\ 5 & 3 \end{pmatrix}$ dan matriks $3 \times 3 \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$

3.3Penyandian dengan Matriks 2 × 2

a. Enkripsi Citra

Enkripsi data dilakukan untuk mengubah data asli menjadi data yang tidak bisa dibuka oleh orang yang tidak berhak. adapun citra yang mau di enkripsikan adalah sebagai berikut :

Gambar 3.3 Citra photo yang akan dienkripsi dengan Matriks 2×2

Citra Asli Dengan Variasi Nilai RGB dengan ukuran piksel 477 × 357

Matriks Kunci $K = \begin{pmatrix} 2 & 1 \\ 5 & 3 \end{pmatrix}$ dan piksel yang akan diproses

$$P_1 = \begin{pmatrix} 200 & 150 & 200 \\ 200 & 150 & 150 \end{pmatrix};\ P_2 = \begin{pmatrix} 100 & 120 & 10 \\ 100 & 10 & 70 \end{pmatrix};\ P_3 = \begin{pmatrix} 40 & 20 & 30 \\ 0 & 100 & 20 \end{pmatrix}$$

Adapun persamaan yang digunakan untuk menghasilkan piksel *output* adalah sebagai berikut : $Cn = K \cdot P_n \cdot Mod\ 256$

Untuk proses pertama ($n = 1$) → $C_1 = K \cdot P1 \cdot Mod\ 256$

$$C_1 = \begin{pmatrix} 2 & 1 \\ 5 & 3 \end{pmatrix} \cdot \begin{pmatrix} 200 & 150 & 200 \\ 200 & 150 & 150 \end{pmatrix} Mod\ 256$$

$$C_1 = \begin{pmatrix} 600 & 460 & 550 \\ 1600 & 1200 & 1450 \end{pmatrix} Mod\ 256$$

$$C_1 = \begin{pmatrix} 88 & 194 & 38 \\ 64 & 176 & 170 \end{pmatrix}$$

Untuk proses pertama ($n = 2$) → $C_2 = K \cdot P_2 \cdot Mod\ 256$

$$C_2 = \begin{pmatrix} 2 & 1 \\ 5 & 3 \end{pmatrix} \cdot \begin{pmatrix} 100 & 120 & 10 \\ 100 & 10 & 70 \end{pmatrix} Mod\ 256$$

$$C_2 = \begin{pmatrix} 300 & 250 & 90 \\ 800 & 630 & 260 \end{pmatrix} Mod\ 256$$

$$C_2 = \begin{pmatrix} 44 & 250 & 90 \\ 32 & 118 & 4 \end{pmatrix}$$

Untuk proses pertama ($n = 3$) → $C_3 = K \cdot P_3 \cdot Mod\ 256$

$$C_2 = \begin{pmatrix} 2 & 1 \\ 5 & 3 \end{pmatrix} \cdot \begin{pmatrix} 40 & 20 & 30 \\ 0 & 100 & 20 \end{pmatrix} Mod\ 256$$

$$C_2 = \begin{pmatrix} 80 & 140 & 80 \\ 200 & 400 & 210 \end{pmatrix} Mod\ 256$$

$$C_2 = \begin{pmatrix} 80 & 140 & 80 \\ 200 & 144 & 210 \end{pmatrix}$$

Dari perhitungan di atas untuk tiap pikselnya dalam komponen RGB didapat hasil sebagai berikut dan hasil citra:

$R_{p1} = 88$ $G_{p1} = 194$ $B_{p1} = 38$	$R_{p2} = 64$ $G_{p2} = 176$ $B_{p2} = 170$	$R_{p3} = 44$ $G_{p3} = 250$ $B_{p3} = 90$
$R_{p4} = 32$ $G_{p4} = 118$ $B_{p4} = 4$	$R_{p5} = 80$ $G_{p5} = 140$ $B_{p5} = 80$	$R_{p6} = 200$ $G_{p6} = 144$ $B_{p6} = 210$

Nilai-nilai komponen RGB hasil perhitungan ini dengan menggunakan fungsi SetPixel akan dikembalikan menjadi sebuah piksel dengan warna tertentu sesuai dengan komponen warna Red, Green, dan Blue-nya.

3.4 Dekripsi Citra

Untuk proses sebaliknya yaitu dekripsi langkah yang dipakai hampir sama dengan proses enkripsi. Perbedaannya adalah pada proses dekripsi matriks kunci harus diinvers dahulu dan hasilnya perkalian matriks tidak dimodulokan tetapi ditambahkan dengan nilai 256 hingga tidak dihasilkan bilangan negatif jika hasil yang didapat dalam bilangan negatif. Sedangkan untuk nilai yang didapat merupakan bilangan positif maka baru dimodulokan dengan bilangan 256.

Adapun persamaan yang digunakan untuk menghasilkan piksel *output* adalah sebagai berikut : $Pn = K^{-1} . C_n . \text{Mod } 256$ Sebagai contoh diambil dari proses enkripsi di atas:

Matriks Kunci

$K = \begin{pmatrix} 2 & 1 \\ 5 & 3 \end{pmatrix}$ Maka $K^{-1} = \frac{1}{\det K}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix} = \frac{1}{1}\begin{pmatrix} 3 & -1 \\ -5 & 2 \end{pmatrix} K^{-1} = \begin{pmatrix} 3 & \\ -5 & \end{pmatrix}$

Piksel yang akan dip roses:

$C_1 = \begin{pmatrix} 88 & 194 & 38 \\ 64 & 176 & 170 \end{pmatrix}; C_2 = \begin{pmatrix} 44 & 250 & 90 \\ 32 & 118 & 4 \end{pmatrix} dan\ C_3 = \begin{pmatrix} 80 & 1 \\ 200 & 1 \end{pmatrix}$

Untuk proses pertama ($n = 1$) →

$$P_1 = K^{-1}.C_1$$

$$P_1 = \begin{pmatrix} 3 & -1 \\ -5 & 2 \end{pmatrix}\begin{pmatrix} 88 & 194 & 38 \\ 64 & 176 & 170 \end{pmatrix}$$

$$P_1 = \begin{pmatrix} 200 & 406 & -56 \\ -312 & -618 & 150 \end{pmatrix}$$

$$P_1 = \begin{pmatrix} 200\ Mod\ 256 & 406\ Mod\ 256 & -56 + (1 \\ -312 + (2 \times 256) & -618 + (3 \times 256) & 150\ Mo \end{pmatrix}$$

$$P_1 = \begin{pmatrix} 200 & 150 & 200 \\ 200 & 250 & 150 \end{pmatrix}$$

Untuk proses pertama ($n = 2$) →

$$P_2 = K^{-1}.C_2$$

$$P_2 = \begin{pmatrix} 3 & -1 \\ -5 & 2 \end{pmatrix}\begin{pmatrix} 44 & 250 & 90 \\ 32 & 118 & 4 \end{pmatrix}$$

$$P_2 = \begin{pmatrix} 100 & 632 & 266 \\ -156 & -1014 & -442 \end{pmatrix}$$

$$P_2 = \begin{pmatrix} 100 & 120 & 10 \\ 100 & 10 & 70 \end{pmatrix}$$

Untuk proses pertama ($n = 3$) →

$$P_3 = K^{-1}.C_3$$

$$P_3 = \begin{pmatrix} 3 & -1 \\ -5 & 2 \end{pmatrix}\begin{pmatrix} 80 & 140 & 80 \\ 200 & 114 & 210 \end{pmatrix}$$

$$P_3 = \begin{pmatrix} 40 & 276 & 30 \\ 0 & -412 & 20 \end{pmatrix}$$

$$P_3 = \begin{pmatrix} 40 & 20 & 30 \\ 0 & 100 & 20 \end{pmatrix}$$

Dari perhitungan di atas untuk tiap pikselnya dalam komponen RGB didapat hasil sebagai berikut:

$R_{p1} = 200$ $G_{p1} = 150$ $B_{p1} = 200$	$R_{p2} = 200$ $G_{p2} = 150$ $B_{p2} = 150$	$R_{p3} = 100$ $G_{p3} = 120$ $B_{p3} = 10$
$R_{p4} = 100$ $G_{p4} = 10$ $B_{p4} = 70$	$R_{p5} = 40$ $G_{p5} = 20$ $B_{p5} = 30$	$R_{p6} = 0$ $G_{p6} = 100$ $B_{p6} = 20$

Nilai-nilai komponen RGB hasil perhitungan ini dengan menggunakan fungsi SetPixel akan dikembalikan menjadi sebuah piksel dengan warna tertentu sesuai dengan komponen warna Red, Green, dan Blue-nya.

3.5 Penyandian dengan Matriks 3 × 3

a. Enkripsi Citra

Gambar 3.4 Citra photo yang akan dienkripsi dengan Matriks 3×3

Matriks Kunci $K = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$ dan piksel yang akan di proses adalah sebagai berikut :

$$P_1 = \begin{pmatrix} 200 & 150 & 200 \\ 200 & 150 & 150 \\ 100 & 120 & 10 \end{pmatrix}; P_2 = \begin{pmatrix} 100 & 10 \\ 40 & 20 \\ 0 & 100 \end{pmatrix}$$

Adapun persamaan yang digunakan untuk menghasilkan piksel output sama seperti penyandian dengan menggunakan matriks kunci 2 × 2 yaitu sebagai berikut : Cn = K . P_n . Mod 256

Untuk proses pertama (*n* = 1)□→

$$C_1 = K . P_1 . Mod\ 256$$

$$C_1 = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} . \begin{pmatrix} 200 & 150 & 200 \\ 200 & 150 & 150 \\ 100 & 120 & 10 \end{pmatrix} Mod\ 2$$

$$C_1 = \begin{pmatrix} 500 & 420 & 360 \\ 300 & 270 & 160 \\ 300 & 270 & 210 \end{pmatrix} Mod\ 256$$

$$C_1 = \begin{pmatrix} 244 & 164 & 104 \\ 44 & 14 & 160 \\ 44 & 14 & 210 \end{pmatrix}$$

Untuk proses pertama (*n* = 2)□→

$$C_2 = K . P_2 . Mod\ 256$$

$$C_2 = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} . \begin{pmatrix} 100 & 10 & 70 \\ 40 & 20 & 30 \\ 0 & 100 & 20 \end{pmatrix} Mod\ 25$$

$$C_2 = \begin{pmatrix} 140 & 130 & 120 \\ 40 & 120 & 50 \\ 100 & 110 & 90 \end{pmatrix} Mod\ 256$$

$$C_2 = \begin{pmatrix} 140 & 130 & 120 \\ 40 & 120 & 50 \\ 100 & 110 & 90 \end{pmatrix}$$

Dari perhitungan di atas untuk tiap pikselnya dalam komponen RGB didapat hasil sebagai berikut:

$R_{p1} = 244$ $G_{p1} = 164$ $B_{p1} = 104$	$R_{p2} = 44$ $G_{p2} = 14$ $B_{p2} = 160$	$R_{p3} = 44$ $G_{p3} = 14$ $B_{p3} = 210$
$R_{p4} = 140$ $G_{p4} = 130$ $B_{p4} = 120$	$R_{p5} = 40$ $G_{p5} = 120$ $B_{p5} = 50$	$R_{p6} = 100$ $G_{p6} = 110$ $B_{p6} = 90$

Nilai-nilai komponen RGB hasil perhitungan ini dengan menggunakan fungsi SetPixel akan dikembalikan menjadi sebuah piksel dengan warna tertentu sesuai dengan komponen warna Red, Green, dan Blue-nya.

3.6 Dekripsi Citra

Untuk proses sebaliknya yaitu dekripsi langkah yang dipakai hampir sama dengan proses enkripsi dan aturan yang berlaku untuk perhitungan dengan matriks kunci 2 × 2 juga berlaku untuk perhitungan dengan matriks kunci 3 × 3.

Adapun persamaan yang digunakan untuk menghasilkan piksel *output* adalah sebagai berikut : Pn = K^{-1} . C_n . Mod 256

Sebagai contoh diambil dari proses enkripsi di atas :

Matrik Kunci

$$K = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$ Untuk mencari invers dari matriks ini adalah digunakan persamaan berikut $K^{-1} = \frac{1}{\det K} adj\ K.$

Untuk itu maka langkah pertama adalah mencari nilai determinan untuk matriks 3 × 3 di atas terlebih dahulu dengan menggunakan aturan Sarrus.

$$\text{Det K} = \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix}$$

$$= (K_{11} * K_{22} * K_{33}) + (K_{12} * K_{23} * K_{31}) + (K_{13} * K_{21} * K_{32}) - (K_{31} * K_{22} * K_{13}) - (K_{32} * K_{23} * K_{11}) - (K_{33} * K_{21} * K_{12})$$

$$\text{Det K} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

$$= (1 * 1 * 1) + (1 * 1 * 1) + (1 * 0 * 0) - (1 * 1 * 1) - (0 * 1 * 1) - (1 * 1 * 0) = 1 + 1 + 0 - 1 - 0 - 0 = 1$$

Sedangkan untuk mencari nilai **adj** K (dibaca *adjoin* matriks K) ditentukan dengan persamaan:

$$\text{Adj K} = \begin{pmatrix} \alpha_{11} & \alpha_{21} & \alpha_{31} \\ \alpha_{12} & \alpha_{22} & \alpha_{32} \\ \alpha_{13} & \alpha_{13} & \alpha_{33} \end{pmatrix}$$

dengan □11 adalah kofaktor dari $|K_{ij}|$ yang dapat dicari dengan menggunakan rumus: □ij = (□1)$^{i+j}$ |Mij| dimana |M| adalah minordari K dari matriks K. Minor dicari dengan menghapus elemen-elemen pada baris ke-i dan kolom ke-j dari matriks K maka akan diperoleh matriks persegi berordo 2. Determinan dari matriks persegi berordo 2 itu yang disebut dengan minor. Maka untuk enentukan minor dan kofaktor dari matriks K di atas adalah sebagai berikut:

Maka untuk menentukan minor dan kofaktor dari matriks K di atas adalah sebagai berikut:

kofaktor dari K_{11} adalah

$$\alpha_{11} = (-1)^{1+1}|M_{11}| = + \begin{vmatrix} 1 & 1 \\ 0 & 1 \end{vmatrix} = +(1.1 - 1.0) = 1$$

kofaktor dari K_{12} adalah

$$\alpha_{12} = (-1)^{1+2}|M_{12}| = + \begin{vmatrix} 0 & 1 \\ 1 & 1 \end{vmatrix} = -(0.1 - 1.1) = 1$$

kofaktor dari K_{13} adalah

$$\alpha_{13} = (-1)^{1+3}|M_{13}| = +\begin{vmatrix}0 & 1\\ 1 & 0\end{vmatrix} = +(0.0 - 1.1) =$$

kofaktor dari K_{21} adalah

$$\alpha_{21} = (-1)^{2+1}|M_{21}| = +\begin{vmatrix}1 & 1\\ 0 & 1\end{vmatrix} = -(1.1 - 1.0) =$$

kofaktor dari K_{22} adalah

$$\alpha_{22} = (-1)^{2+2}|M_{21}| = +\begin{vmatrix}1 & 1\\ 1 & 1\end{vmatrix} = +(1.1 - 1.1) =$$

kofaktor dari K_{23} adalah

$$\alpha_{23} = (-1)^{2+3}|M_{23}| = +\begin{vmatrix}1 & 1\\ 1 & 0\end{vmatrix} = -(1.0 - 1.1)$$

kofaktor dari K_{31} adalah

$$\alpha_{31} = (-1)^{3+1}|M_{31}| = +\begin{vmatrix}1 & 1\\ 1 & 1\end{vmatrix} = +(1.1 - 1.1) =$$

kofaktor dari K_{32} adalah

$$\alpha_{32} = (-1)^{3+2}|M_{32}| = +\begin{vmatrix}1 & 1\\ 0 & 1\end{vmatrix} = -(1.1 - 1.0) = -1$$

kofaktor dari K_{33} adalah

$$\alpha_{33} = (-1)^{3+3}|M_{33}| = +\begin{vmatrix}1 & 1\\ 0 & 1\end{vmatrix} = +(1.1 - 1.0) =$$

Dari hasil kofaktor yang didapat maka hasil disusun kembali untuk membentuk adj K yaitu:

$$\text{Adj K} = \begin{pmatrix}1 & -1 & 0\\ 1 & 0 & -1\\ -1 & 1 & 1\end{pmatrix};$$

maka untuk mencari invers dari K atau K^{-1} adalah sebagai berikut :

$$K^{-1} = \frac{1}{1}\begin{pmatrix}1 & -1 & 0\\ 1 & 0 & -1\\ -1 & 1 & 1\end{pmatrix}$$ didapat hasil :

$$K^{-1} = \begin{pmatrix}1 & -1 & 0\\ 1 & 0 & -1\\ -1 & 1 & 1\end{pmatrix}$$

Selanjutnya untuk mendapatkan piksel citra hasil maka dilakukan perkalian matriks dengan menggunakan persamaan: Pn = K^{-1} . Cn . Mod 256

$$K^{-1} = \begin{pmatrix}1 & -1 & 0\\ 1 & 0 & -1\\ -1 & 1 & 1\end{pmatrix}$$ dan piksel yang akan diproses adalah sebagai berikut :

$$C_1 = \begin{pmatrix}244 & 164 & 104\\ 44 & 14 & 160\\ 44 & 14 & 210\end{pmatrix}; C_2 = \begin{pmatrix}140 & 130 & 120\\ 40 & 120 & 50\\ 100 & 110 & 90\end{pmatrix}$$

Untuk proses pertama (n = 1) →

$$P_1 = K^{-1}.C_1$$

$$P_1 = \begin{pmatrix}1 & -1 & 0\\ 1 & 0 & -1\\ -1 & 1 & 1\end{pmatrix}.\begin{pmatrix}244 & 164 & 104\\ 44 & 14 & 160\\ 44 & 14 & 210\end{pmatrix}$$

$$P_1 = \begin{pmatrix}200 & 150 & -56\\ 200 & 150 & -106\\ -156 & -136 & 266\end{pmatrix}$$

$$P_1 = \begin{pmatrix}200\ \text{Mod}\ 256 & 150\ \text{Mod}\ 256 & -56 + 1.256\\ 200\ \text{Mod}\ 256 & 150\ \text{Mod}\ 256 & -106 + 1.256\\ -156 + 1.256 & -136 + 1.256 & 266\ \text{Mod}\ 256\end{pmatrix}$$

$$P_1 = \begin{pmatrix}200 & 150 & 200\\ 200 & 150 & 150\\ 100 & 120 & 10\end{pmatrix}$$

Untuk proses pertama (n =2) →

$$P_1 = K^{-1}.C_1$$

$$P_1 = \begin{pmatrix}1 & -1 & 0\\ 1 & 0 & -1\\ -1 & 1 & 1\end{pmatrix}.\begin{pmatrix}140 & 130 & 120\\ 40 & 120 & 50\\ 100 & 110 & 90\end{pmatrix}$$

$$P_1 = \begin{pmatrix}100 & 10 & 70\\ 40 & 20 & 30\\ 0 & 100 & 20\end{pmatrix}$$

$$P_1 = \begin{pmatrix}100\ \text{Mod}\ 256 & 10\ \text{Mod}\ 256 & 70\ \text{Mod}\ 256\\ 40\ \text{Mod}\ 256 & 20\ \text{Mod}\ 256 & 30\ \text{Mod}\ 256\\ 0\ \text{Mod}\ 256 & 100\ \text{Mod}\ 256 & 20\ \text{Mod}\ 256\end{pmatrix}$$

$$P_1 = \begin{pmatrix}100 & 10 & 70\\ 40 & 20 & 30\\ 0 & 100 & 20\end{pmatrix}$$

Dari perhitungan di atas untuk tiap pikselnya dalam komponen RGB didapat hasil sebagai berikut:

$R_{p1} = 200$ $G_{p1} = 150$ $B_{p1} = 200$	$R_{p2} = 200$ $G_{p2} = 150$ $B_{p2} = 150$	$R_{p3} = 100$ $G_{p3} = 120$ $B_{p3} = 10$
$R_{p4} = 100$ $G_{p4} = 10$ $B_{p4} = 70$	$R_{p5} = 40$ $G_{p5} = 20$ $B_{p5} = 30$	$R_{p6} = 0$ $G_{p6} = 100$ $B_{p6} = 20$

Nilai-nilai komponen RGB hasil perhitungan ini dengan menggunakan fungsi SetPixel akan dikembalikan menjadi sebuah piksel dengan warna tertentu sesuai dengan komponen warna Red, Green, dan Blue-nya.

4.1 Perancangan

Antarmuka merupakan suatu media interaksi (interaktif) antara komputer dengan pemakai (*user*). Pada sistem operasi yang berbasis grafis (*graphic user interface* atau GUI) seperti Windows, antarmuka dari suatu perangkat lunak biasanya berupa jendela (*window*). Melalui jendela inilah pemakai dapat berinteraksi dengan perangkat lunak yang digunakannya.

Perancangan program ini meliputi penempatan dan penyusunan objek-objek yang terdapat dalam

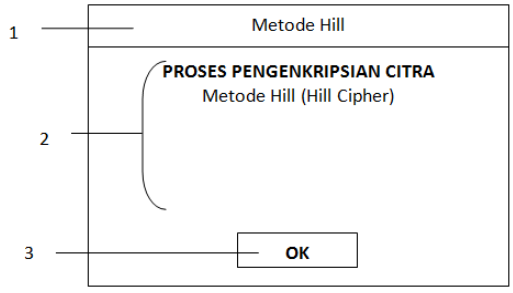
form Visual Basic. *Form* yang dirancang terlebih dahulu dibuat rancangan dasarnya agar memudahkan sewaktu membuat rancangan di dalam Visual Basic.

4.2.1 Perancangan *Form*

Form pada program ini adalah terdiri atas tiga buah *form* yaitu *form splash*, *form* utama, dan *form* konfirmasi. Adapun fungsi dari ketiga *form-form* ini dapat dirinci
sebagai berikut:

1. *Form Splash* untuk memunculkan keterangan program beserta nama penulis pada saat pertama kali program di-*load*. Dengan kata lain *form* ini berfungsi sebagai *splash screen*.
2. Form Utama merupakan bentuk tampilan utama program dimana *user* dapat melakukan hal-hal sepert i me-*load* citra yang akan diproses, menyimpan, memprosesnya dengan *Hill Cipher*.
3. Form Konfirmasi mempunyai fungsi untuk meminta konfirmasi *user* jumlah iterasi yang akan dilakukan sekaligus meminta *user* apakah akan melakukan proses enkripsi atau sebaliknya dekripsi pada citra yang diinput.

Bentuk rancangan dasar dari *form splash* dapat dilihat pada Gambar 4.1 di bawah ini.

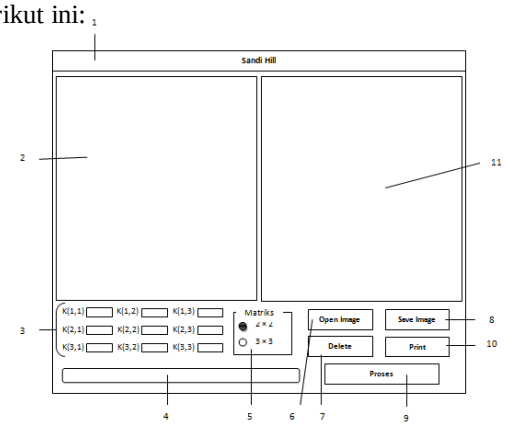


Gambar 4.1 Perancangan *Form Splash*

Pada gambar di atas terdapat beberapa komponen-komponen visual yang mempunyai fungsi-fungsi sebagai berikut:

1. Bagian ini biasanya disebut dengan *title bar* yang berfungsi sebagai judul program.
2. Bagian ini merupakan objek *label* yang berfungsi untuk mencetak tulisan keterangan.
3. Bagian ini adalah objek *command button* yang berfungsi untuk menutup *form* ini dan memunculkan *form* utama.

Bentuk rancangan dasar atau biasanya disebut dengan layout dari *form* utama dapat dilihat pada Gambar 4.2 berikut ini:

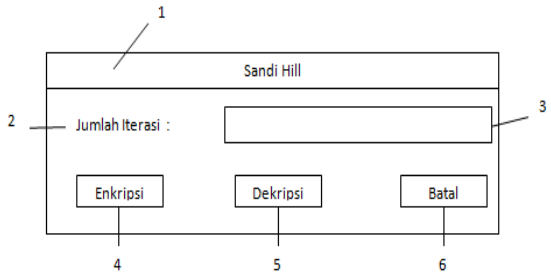


Gambar 4.2 Perancangan *Form* Utama

Pada gambar di atas terdapat beberapa komponen-komponen visual yang mempunyai fungsi-fungsi sebagai berikut:

1. Bagian ini biasanya disebut dengan *title bar* yang berfungsi sebagai judul program.
2. Bagain ini adalah objek *picture box* yang digunakan sebagai tempat untuk memunculkan citra input oleh *user*.
3. Bagian ketiga ini dibuat dari objek label untuk label keterangannya sedangkan kotak pengisiannya dibuat dari objek *text box*. Bagian ini berfungsi untuk meminta input matriks kunci (K).
4. Bagian ini merupakan progress bar yang berfungsi untuk menampilkan status kemajuan proses pada saat *enciphering* dan *deciphering* citra.
5. Bagian ini dibuat dengan dua jenis objek yaitu objek *frame* untuk mengelompokkan tiga buah *option button* yang ada di dalamnya. Berfungsi untuk menentukan apakah ordo matriks adalah dua, tiga atau empat. Bila pilihannya hanya dua yang ditandai maka bagian keempat hanya tersedia empat buah *text box* untuk diisi nilainya.
6. Bagian ini merupakan *command button* yang berfungsi untuk memilih *file*citra yang akan dijadikan citra input. Bila *user* mengklik pada tombol ini akan dimunculkan sebuah kotak dialog *open*.
7. Bagian ini merupakan objek *command button* yang bertujuan untuk membersihkan *picture box* pada bagian ketiga dan kesebelas.
8. Bagian ini merupakan objek *command button* yang berfungsi untukmenyimpan citra *output* ke dalam format *bitmap* dengan memunculkan sebuah kotak dialog *save*.
9. Bagian ini merupakan objek *command button* yang berfungsi untuk memunculkan *form* konfirmasi.
10. Bagian ini merupakan objek *command button* yang berfungsi untuk mencetak citra hasil ke *printer*.

Bentuk rancangan dasar dari *form* konfirmasi dapat dilihat pada Gambar 4.3 di bawah ini.



Gambar 4.3 Perancangan *Form* Konfirmasi

Pada gambar di atas terdapat beberapa komponen-komponen visual yang mempunyai fungsi-fungsi sebagai berikut:

1. Bagian ini biasanya disebut dengan *title bar* yang berfungsi sebagai judul program.
2. BBagian ini merupakan objek label yang berfungsi untuk mencetak tulisan "Jumlah Iterasi:"
3. Bagian ini merupakan objek *combo box* untuk mengisi dan memilih jumlah iterasi.
4. Bagian ini merupakan objek *command button* untuk melakukan proses *ciphering* citra.
5. Bagian ini merupakan objek *command button* untuk melakukan proses *deciphering* citra.
6. Bagian ini merupakan objek *command button* untuk membatalkan proses, menutup *form* konfirmasi dan tampilan kembali ke *form* utama.

4.3 Perancangan *Class Module*

Pada perancangan dan pembuatan program ini dirancang sebuah *Class Module* yang diberi nama clsHill.cls (singkatan dari Class Hill). Pada *class module* ini dikodekan rut in-rutin untuk proses *ciphering* dan *deciphering* dengan algoritma *Hill Cipher*. Fungsi pembacaan *piksel* juga disertakan dalam *class module* ini. Adapun rutin-rutin yang terdapat pada *class module* ini adalah:

1. Function Enhill adalah fungsi untuk melalukan proses *ciphering*. Fungsi ini akan menghasilkan nilai *boolean* berupa *true* jika proses *ciphering* berhasil dan *false* jika proses *ciphering* gagal. Bentuk deklarasinya adalah sebagai berikut:
   Function EnHill (picSource As PictureBox, picDest As PictureBox, MatrixKey(), MatrixSize As Byte) As Boolean Fungsi di atas memiliki empat buah parameter yaitu sebagai berikut:
   a. PicSource yang merupakan parameter input citra yang akan disandikan dengan *Hill Cipher*.
   b. PicDest merupakan parameter citra output hasil penyandian dengan *Hill Cipher*.
   c. MatrixKey() merupakan *array* matriks kunci yang berisikan nilai-nilai dalam bentuk bilangan bulat.
   d. MatrixSize merupakan parameter untuk menyatakan ukuran ordo matriks kunci.
2. Fungsi DeHill adalah fungsi untuk melalukan proses *deciphering*. Fungsi ini kan menghasilkan nilai *boolean* berupa *true* jika proses *deciphering* berhasil dan *false* jika proses *deciphering* gagal. Bentuk deklarasinya adalah sebagai berikut: DeHill(picSource As PictureBox, picDest As PictureBox,MatrixKey(), atrixSize As Byte) As Boolean Fungsi di atas memiliki empat buah parameter yaitu sebagai berikut:
   a. picSource merupakan parameter *input* citra yang akan disandikan balik dengan *Hill Cipher*.
   b. picDest merupakan parameter citra *output* hasil penyandian balik dengan *Hill Cipher*.
   c. MatrixKey() merupakan *array* matriks kunci yang berisikan nilai-nilai dalam bentuk bilangan bulat.
   d. MatrixSize merupakan parameter untuk menyatakan ukuran ordo matriks kunci.

4.3.1 Perancangan *Module*

Berbeda dengan *class module* maka fungsi-fungsi yang terdapat pada *module* tidak perlu dideklarasikan sebagai objek sewaktu akan digunakan. Perancangan module ini dibuat dalam sebuah *file* yang disimpan dengan nama modMain.bas. Adapun fungsifungsi yang terdapat pada module antara lain:

1. GetPixel berfungsi untuk membentuk piksel berwarna pada koordinat tertentu.
2. SetPixel berfungsi untuk mengambil nilai RGB piksel dari koordinat tertentu pada citra.
3. Det22 berfungsi untuk mengecek apakah input kunci matriks 2 × 2 bernilai 0 (nol) atau tidak jika ya maka dikembalikan nilai *true* dan sebaliknya *false* jika tidak bernilai nol.
4. Det33 berfungsi untuk mengecek apakah *input* kunci matriks 3 × 3 bernilai 0 (nol) atau tidak jika ya maka dikembalikan nilai *true* dan sebaliknya *false* jika tidak bernilai nol.
5. VDet22 berfungsi untuk mengembalikan nilai determinan matriks 2 × 2.
6. VDet33 berfungsi untuk mengembalikan nilai determinan matriks 3 × 3.

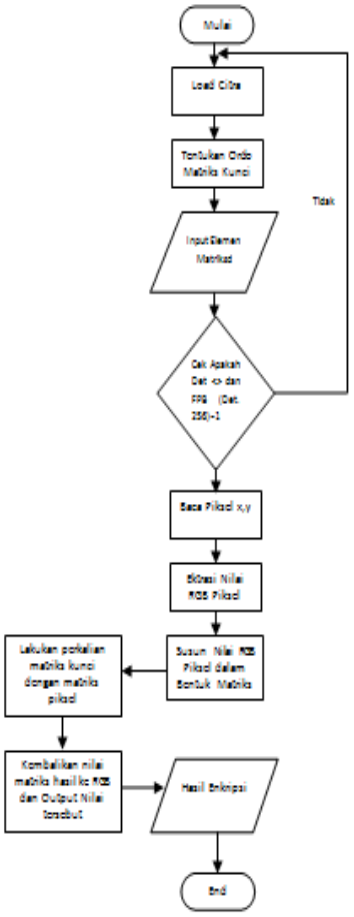
4.3.1 Program *Flowchart*

Dalam pengolahan data dengan mempergunakan komputer telah diketahui bahwa kedudukan komputer hanyalah sebagai alat. Agar suatu masalah dapat diselesaikan dengan memakai komputer maka harus ada serangkaian instruksi yang diberikan kepada komputer untuk memecahkan masalah tersebut sehingga komputer mampu bekerja memecahkan masalah menurut instruksi-instruksi yang sudah ditentukan. Bekerjanya komputer sangat bergantung kepada instruksi-instruksi yang diberikan dan komputer akan bekerja menurut urutan intruksi itu.

Program adalah serangkaian instruksi yang disusun untuk menyelesaikan suatu pekerjaan dengan

menggunakan komputer. Sedangkan program *flowchart* adalah suatu bagan dengan simbol-simbol tertentu yang menggmbarkan urutan-urutan proses secara mendetail dan hubungan antara suatu proses dengan proses lainnya dalam suatu program.

Dalam perancangan logika program ini, penulis membutuhkan dua program *flowchart* yaitu program *flowchart* untuk proses enkrip file dan program *flowchart* untuk proses dekrip file yang masing-masing ditunjukkan oleh gambar dan gambar 4.4 di bawah ini.

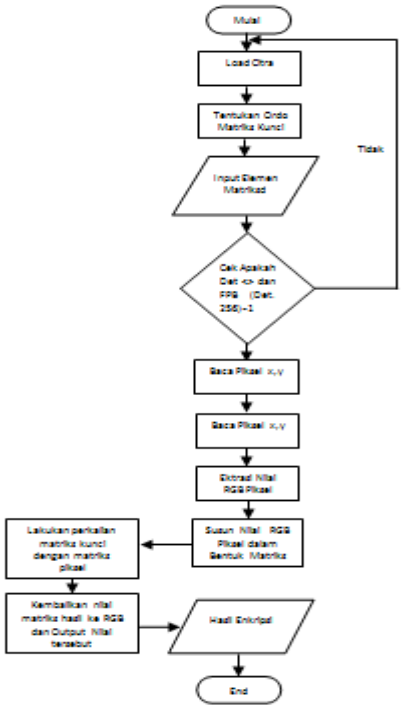


Gambar 4.4 Diagram Alir Enkripsi atau Penyandian Citra

Keterangan:

1. Bagian ini merupakan inisialisasi variabel.
2. Bagian ini merupakan rutin untuk me-*load* citra ke *picture box* berdasarkan nama *file* citra yang dipilih.
3. Menentukan apakah matriks kunci berordo 2, 3 atau 4.
4. Input bilangan bulat untuk tiap elemen matriks kunci.
5. Lakukan pengecekan apakah matriks kunci yang diinput mempunyai determinan dan FPB (det, 256) = 1.
6. Membaca nilai piksel dari citra sesuai algoritma pembacaan piksel.
7. Ekstraksi Nilai RGB seperti pada algoritma pembacaan piksel.
8. Susun Nilai RGB piksel dalam bentuk matriks sesuai dengan ordo matriksseperti cara di atas.
9. Lakukan perkalian matriks kunci dengan matriks piksel untuk menghasilkan matriks cipher. C = K . P Mod 256
10. Baca nilai RGB dari matriks *cipher* dan kembalikan dengan fungsi SetPixel.
11. Jika merupakan piksel akhir maka End

Sedangkan untuk proses sebaliknya yaitu dekripsi citra balik untuk mengembalikan citra asli dengan menggunakan algoritma dekripsi *Hill Cipher* dapat dinyatakan dengan diagram alir (Gambar 4.5) di bawah ini.



Gambar 4.5 Diagram Alir Dekripsi atau Penyandian Balik Citra

Keterangan:

1. Bagian ini merupakan inisialisasi variabel.
2. Bagian ini merupakan rutin untuk me-*load* citra ke *picture box* berdasarkan nama *file* citra yang dipilih.
3. Menentukan apakah matriks kunci berordo 2, 3 atau 4.
4. Input bilangan bulat untuk tiap elemen matriks kunci.
5. Lakukan pengecekan apakah matriks kunci yang diinput mempunyai determinan dan FPB (det, 256) = 1.
6. Menghitung nilai invers dari matriks kunci.
7. Membaca piksel citra input sesuai dengan algoritma membaca piksel.

8. Mengekstraksi komponen RGB dengan algoritma membaca piksel.
9. Menyusun nilai RGB piksel sesuai dengan ordo matriks yang dipakai.
10. Lakukan perkalian invers matriks kunci dengan matriks piksel untuk memperoleh piksel citra asli kembali. $P = K^{-1} . C \text{ Mod } 256$
11. Mengembalikan nilai matriks hasil ke nilai RGB dengan fungsi SetPixel.
12. End

5.1 Kesimpulan

Berdasarkan hasil penelitian ini, maka penulis mengambil kesimpulan sebagai berikut :

1. Algoritma Hill Chiper mampu mendeteksi nilai piksel dari suatu citra digital.
2. Proses enkripsi citra digital menggunakan algoritma Hill Chiper ini dilakukan dengan cara mengambil nilai RGB, lalu di XOR kan antara nilai RGB dengan nilai ASCII kata sandi, lalu disusun kembali menjadi sebuah citra digital yang terenkripsi
3. Algoritma Hil Chiper dapat mendeteksi suatu citra digital walaupun memiliki tingkat piksel yang tinggi.
4. Hasil enkripsi citra digital dengan menggunakan algoritma Hill Chiper dapat digunakan sebagai *output* menjadi teknik pengolahan kriptograpi citra digital yang baik.
5. Hasil enkripsi citra digital ini dapat digunakan menjadi sistem pengamanan agar bentuk citra digital tersebut tidak dapat diketahui dan disalah gunakan oleh orang lain sehingga bisa merugikan pemiliknya baik secara materil maupun immateril.

DAFTAR PUSTAKA

[1] Agustinus, N., 2006, ***Membuat Program Profesional Secara Cepat Dengan Visual Basic 6.0***, Penerbit T. Elex Media Komputindo, Jakarta.

[2] Dony Ariyus, 2008, ***Pengantar Ilmu Kriptografi Teori, Analisis, dan Implementasi***, Penerbit Andi, Yogyakarta**.**

[3] Gehani, Ashish; Thomas LaBean dan John Reif., 1999, ***DNA-based Cryptography***,. Duke University.

[4] Jain, A.K., 1989, ***Fundamentals of Digital Image Processing,*** Englewood Cliffs : Prectice Hall

[5] Kang, Nin., 2004, ***A Pseudo DNA Cryptography Method***, National University of Singapore.

[6] Ramadhan, A., 2007, ***Visual Basic 6.0,*** Penerbit PT. Elex Media Komputindo, Jakarta.

[7] Schneiner, B., 1994, ***Applied Cryptography : Protocols, Algorithm, adn Sorce Code in C,*** New York : Wiley.

[8] Wahana Komputer, 2003, ***Memahami Model Enkripsi dan Security Data,*** Penerbit Andi, Yogyakarta.