BAB III

METODOLOGI PERANCANGAN

A. Analisis Perancangan

Metode pengembangan analisa perancangan yang penulis gunakan, maka tahap analisis merupakan tahap pertama kali yang penulis lakukan. Pada tahap ini penulis melakukan beberapa kegiatan diantaranya berupa analisa sistem, pengumpulan data, identifikasi masalah, analisa teknologi yang digunakan, analisa kebutuhan *sofware* dan *hardware*. Kegiatan tersebut dilakukan penulis melalui wawancara dan studi pustaka dalam hal yang berkaitan dengan Perancangan Penyembunyian Pesan Pada *Image* Berformat *JPEG* Dengan Metode *LSB* dan *Viginer Chiper*.

Dari hasil wawancara yang dilakukan diperoleh data-data yang dibutuhkan untuk membuat perancangan. Data yang diperoleh berupa gambar dengan format *JPEG* dan Tulisan pesan yang akan di enkripsi. Dari penelitian ini akan dibuat perancangan yang dapat memberikan informasi kepada *user* untuk mengetahui kelemahan dan kelebihan program. Adapun kelemahan dari rancangan ini adalah pesan tidak akan terbaca jika user lupa atau kehilangan kunci kode.

B. Pengumpulan Data

Pengumpulan data merupakan tahapan penting dalam proses riset atau perancangan, karena dengan mendapatkan data yang tepat maka perancangan akan berlangsung sesuai dengan perumusan masalah yang sudah ditentukan.

Metodologi pengumpulan data yang digunakan adalah studi pustaka dan wawancara

1. Studi pustaka

Metodologi ini dilakukan dengan mempelajari teori teori yang terkait dengan penelitian yang dapat mendukung pemecahan masalah perancangan. Pencarian referensi dilakukan diperpustakaan, toko buku, maupun secara *online* melalui internet. Pustaka-pustaka yang dijadikan acuan dapat dilihat di Daftar pustaka.

2. Wawancara

Penulis melakukan pertemuan dan wawancara kepada pihak-pihak yang nantinya akan berhubungan dengan perancangan ini. Pihak-pihak yang dimaksud adalah seseorang pakar komputer yang akan berfungsi sebagai admin dari sistem ini.

C. Analisis dan Spesifikasi Perancangan

Analisis kebutuhan digunakan untuk mengidentifikasi terhadap kebutuhan sistem perancangan yang akan dibuat. Kebutuhan sistem meliputi dari kebutuhaan *hardware* dan kebutuhan software.

1. Kebutuhan perangkat keras (*hardware*) yang digunakan meliputi :
    a) Perangkat komputer

        Dibawah ini merupakan spesifikasi Perangkat komputer yang digunakan untuk membuat *program.*

        - Prosesor : intel pentium 4/1,8 GHz
        - Memory : 2GB

- Hardisk : 80GB
- CDROM : 52x
- Monitor : SVGA 15”
- Keyboard dan mouse : serial/pas2
- Soundcard : onboard
- Stabilizer : denkyu 500 VA
- Instalasi software : windows 7, sofware standard

2. Kebutuhan perangkat lunak (sofware) yang digunakan meliputi :

   a) *JDK (Java Development Kit)*

      Perangkat lunak yang digunakan untuk melakukan proses kompilasi dari kode *Java* ke *byte*code yang dapat dimengerti dan dapat dijalankan oleh JRE (*Java* Runtime Envirotment).

   b) Net Bean

      sebuah aplikasi *Integrated Development Environment* (*IDE*) yang berbasiskan *Java* dari *Sun Microsystems* yang berjalan di atas *swing*. *Swing* merupakan sebuah teknologi *Java* untuk pengembangan aplikasi dekstop yang dapat berjalan pada berbagai macam platform seperti windows, linux, Mac OS X dan Solaris.

   c) Microsoft Windows 7 profesional

      Sebagai sistem operasi.

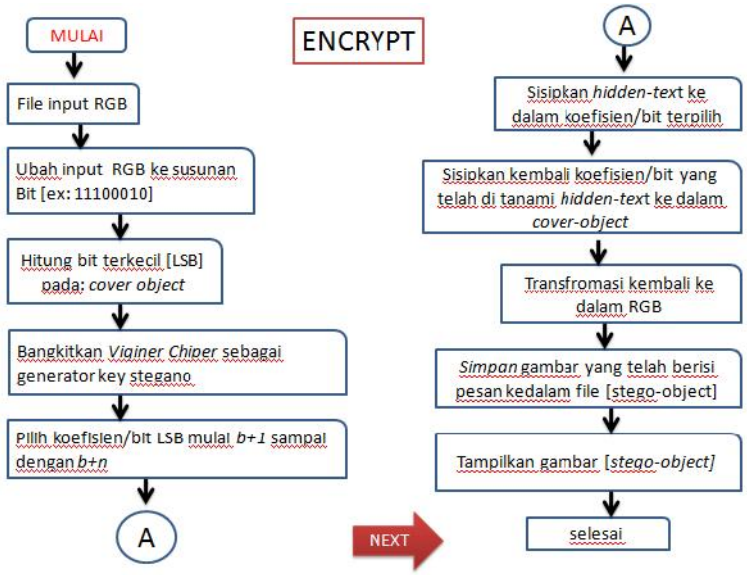
D. Perancangan sistem

Perancangan ini dibuat untuk memberikan informsi atau gambaran dari tugas skripsi ini. perancangan divisualisasikan dalam bentuk gambar yang terdiri dari beberapa screenshot.

1. Penentuan Arsitektur Sistem

   a. Enkripsi

Simulasi enkripsi yang dibangun dalam penelitian ini memiliki arsitektur terlihat seperti **Gambar 3.1** Pengguna simulasi hanya perlu memasukan pesan dan kunci dan simulasi akan memproses sandi menggunakan algoritma modifikasi *vigenere cipher* dan *LSB*

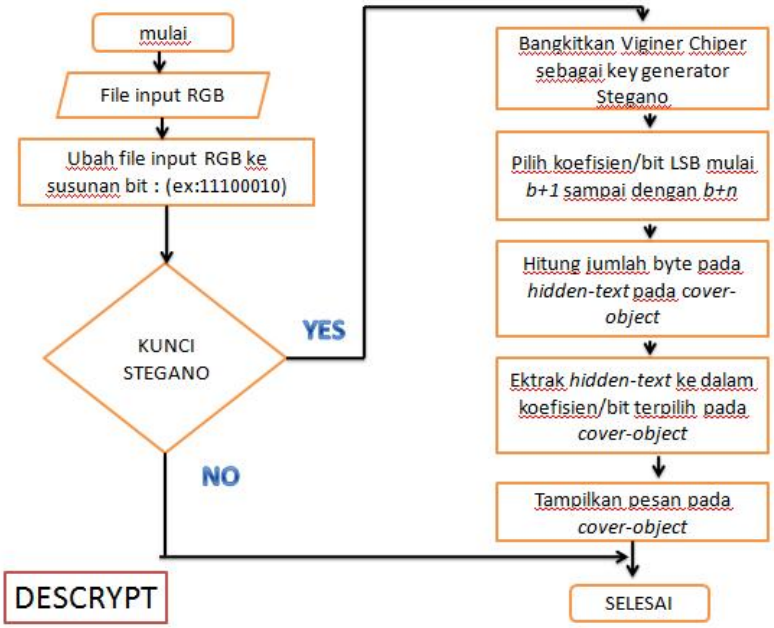


Gambar 3.1 *Encrypt*

Setelah pengguna memasukkan gambar (file input RGB) program akan meminta kita memasukkan kunci viginer chiper. Kemudian pesan dimasukkan dan dienkripsi menjadi file gambar dengan pesan. Gambar->Kunci->Pesan->*LSB*=Gambar (dengan pesan tersembunyi)

b. Deskripsi

Simulasi deskripsi yang dibangun dalam penelitian ini memiliki arsitektur terlihat seperti **Gambar 3.2** Pengguna simulasi hanya perlu memasukan kunci dan simulasi akan memproses kunci tersebut menggunakan algoritma modifikasi *vigenere cipher* dan *LSB.*

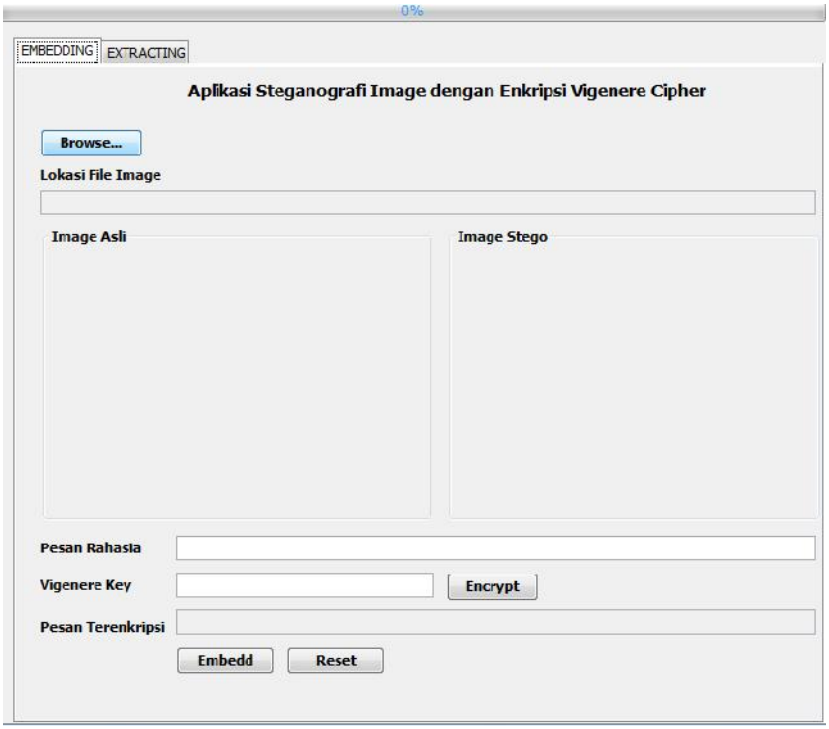


Gambar 3.2 *Decrypt*

Setelah pengguna memasukkan gambar yang berisi pesan yang tersembunyi kemudian program akan meminta kunci *viginere chiper* , jika kunci yang dimasukan tidak sesuai maka pesan tidak akan terurai. jika kata kunci yang di masukan benar maka pesan yang terkunci akan terurai. Gambar (dengan pesan tersembunyi)+Kunci->*LSB*=Gambar+Pesan.

2. Perancangan Antarmuka

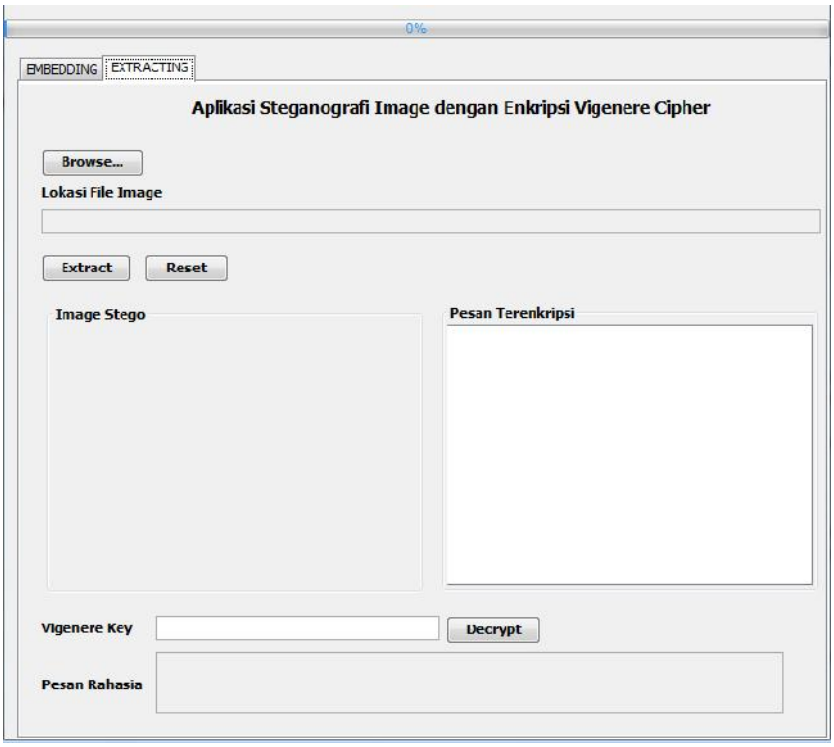
Perancangan antarmuka digunakan untuk menciptakan tampilan program yang mudah digunakan dan dipahami oleh pengguna. Pada program yang digunakan terapat dua tampilan yaitu tampilan untuk enkripsi teks (*EMBEDDING*) dan tampilan untuk dekripsi teks (*EXTRACT*ING). Rancangan antarmuka enkripsi teks yang akan digunakan pada penelitian ini seperti dibawah.



Gambar 3.3 Embedding

Terlihat pada Gambar 3.3 pada sebelah kiri merupakan panel menu *Embedding* dimana terdapat empat tombol yaitu *Browse*, *Encrypt*,*embed* dan *Reset*. Pada bagian browse merupakan tombol untuk mencari gambar yang akan di isi dengan pesan. Kolom Pesan rahasia berfungsi untuk memasukkan pesan. Kolom *Viginere key* berfungsi untuk memasukkan

kunci keamanan, kemudian tombol *Encrypt* untuk mengkonversi gambar,pesan dan kunci menjadi kode yang masuk ke kolom Pesan Terenkripsi, setelah itu pesan terenkripsi akan di *Embed* menjadi gambar kembali dengan metode *LSB*.



Gambar 3.4 *Extract*ion

Terlihat pada Gambar 3.4 pada sebelah kanan merupakan panel menu *Extract*ing dimana terdapat tiga tombol yaitu *Browse*, *Extract* dan *Decrypt*. Pada bagian browse merupakan tombol untuk mencari gambar yang berisi pesan rahasia. kemudian tombol *extract* untuk mengkonversi gambar menjadi pesan terenkripsi. setelah itu masukkan kode kunci ke

dalam kolom *Viginere key* kemudian tekan tombol *Decrypt* agar pesan yang terenkripsi terlihat di kolom *Pesan Rahasia.*

3. Pembuatan Algoritma *Vigenere Cipher*

Pembuatan algoritma enkripsi dan deskripsi dilakukan dalam bahasa program. Simulasi yang digunakan pada penelitian ini menggunakan bahasa pemrograman *Java* dan dibangun menggunakan perangkat lunak NetBeans IDE versi 8.1 dan *Java Development Kit* (JDK) versi 8. Proses pembuatan dengan bahasa program dapat dilihat di bawah

a. *Encrypt*

```
/**
    * Method ini mengenkripsi pesan asli.
    * @param katakunci adalah katakunci.
    * @param pesan adalah pesan asli.
    * @return chipertext dengan enkripsi algoritma VigenereChiper.
    */

    public static String encodeVigenereChiper(String katakunci,String pesan) {
```

- trim all --> hilangkan spasi di depan dan belakang kalimat

```
        pesan=pesan.trim();
        katakunci=katakunci.trim();
```

- jika panjang kata kunci < pesan, maka ulang katakunci hingga=pesan.

```
        if (katakunci.length()<pesan.length()){
            int pjg_kunci=katakunci.length();
            int pjg_pesan=pesan.length();

            int idx=0;
            for (int i = pjg_kunci; i < pjg_pesan; i++) {
                katakunci=katakunci+ katakunci.charAt(idx);
                idx++;
            }
```

- jika panjang kata kunci > pesan, maka katakunci dipotong hingga=pesan.

```
        }else if(katakunci.length()>pesan.length()){
            katakunci=katakunci.substring(0, pesan.length());
        }
```

- ***inisialisasi variable menampung chipertext***

```
        String chipertext="";
        StringBuilder SBchipertext=new StringBuilder();
```

- ***lakukan proses enkripsi vigenere ciphere***

```
        for (int i = 0; i < pesan.length(); i++) {
            int idxpesan=(pesan.charAt(i))-startASCIIchar;
            int idxkatakunci=(katakunci.charAt(i))-startASCIIchar;
            int idxchiper= ((idxpesan+idxkatakunci) %
numberOfChar)+startASCIIchar;
            SBchipertext.append((char)idxchiper);
        }
```

- konversi string builder ke string

```
        chipertext=SBchipertext.toString();
        return chipertext;
    }
```

b. *Decrypt*

```
/**
    * Method ini mendekripsi chiper text.
    * @param katakunci adalah katakunci.
    * @param chipertext adalah chipertext.
    * @return pesan asli.
    */

   public static String decodeVigenereChiper(String katakunci,String
chipertext) {
```

- trim all --> hilangkan spasi di depan dan belakang kalimat

```
        chipertext=chipertext.trim();
        katakunci=katakunci.trim();
```

- jika panjang kata kunci < pesan, maka ulang katakunci hingga=pesan

```
        if (katakunci.length()<chipertext.length()){
            int pjg_kunci=katakunci.length();
```

```
                int pjg_pesan=chipertext.length();
                int idx=0;
                for (int i = pjg_kunci; i < pjg_pesan; i++) {
                    katakunci=katakunci+ katakunci.charAt(idx);
                    idx++;
                }
```

- jika panjang kata kunci > pesan, maka katakunci dipotong hingga=pesan.

```
            }else if(katakunci.length()>chipertext.length()){
                katakunci=katakunci.substring(0, chipertext.length());
            }
```

- inisialisasi variable pesan asli

```
            String pesan="";
            StringBuilder SBpesan= new StringBuilder();
```

- proses dekripsi vigenere cipher

```
            for (int i = 0; i < chipertext.length(); i++) {

                int idxchiper=(chipertext.charAt(i))-start ASCII char;
                int idxkatakunci=(katakunci.charAt(i))-start ASCII char;
                int idxpesan=  (idxchiper-idxkatakunci);
```

- jika hasil index negatif, maka ditambahkan sebanyak numberOfChar

```
                if (idxpesan<0) {
                    idxpesan=idxpesan+numberOfChar;
                }
```

- idxpesan ditambah startASCIIchar

```
                idxpesan=idxpesan+start ASCII char;
```

- tambahkan pesan ke stringbuilder

```
                SBpesan.append((char)idxpesan);
            }
```

- konversi stringbuilder ke string

```
            pesan=SBpesan.toString();
            return pesan;
        }
```

4. Pembuatan Algoritma *List Significant Bit (LSB)*

Pembuatan algoritma Embedding dan *Extract*ing dilakukan dalam bahasa program. Penelitian ini menggunakan bahasa pemrograman *Java* dan dibangun menggunakan perangkat lunak NetBeans IDE versi 8.1 dan *Java* Development Kit (JDK) versi 8. Proses pembuatan dengan bahasa program dapat dilihat di bawah.

a. Embedding

```
public static byte[] embeddMessage(int[] oneDPix, int imgCols, int
imgRows,
                    String str) {

        //pesan asli diberi tanda khusus pada awal dan akhir kalimat
        str += END_MESSAGE_COSTANT;
        str = START_MESSAGE_COSTANT +str;
        //mengambil byte pesan
        byte[] msg = str.getBytes();
        //mendefenisikan 3 channel yaitu R G V
        int channels = 3;
        // variable yang berguna untuk shifting index
        int shiftIndex = 8;
        //variable untuk menampung hasil pixel yang sudah di embedd
pesan
        byte[] result = new byte[imgRows * imgCols * channels];
        //untuk index pesan
        int msgIndex = 0;
        //untuk index citra
        int resultIndex = 0;
        //penanda pesan yang disisipkan sudah berkahir atau belum
        boolean msgEnded = false;

    //looping pada setiap Pixel R G B untuk disisipkan pesan
for (int row = 0; row < imgRows; row++) {
            for (int col = 0; col < imgCols; col++) {
               int element = row * imgCols + col;
               byte tmp = 0;
               for (int channelIndex = 0; channelIndex < channels;
channelIndex++) {
```

```
                if (!msgEnded) {
tmp = (byte) ((((oneDPix[element] >> binary[channelIndex]) & 0xFF) &
0xFE) | ((msg[msgIndex] >> toShift[(shiftIndex++)
                                % toShift.length]) & 0x1));
                    if (shiftIndex % toShift.length == 0) {
                        msgIndex++;
                    }
                    if (msgIndex == msg.length) {
                        msgEnded = true;
                    }
                } else {
                    tmp = (byte) ((((oneDPix[element] >>
binary[channelIndex]) & 0xFF)));
                }

                result[resultIndex++] = tmp;
            }
        }
    }
    return result;
}
```

Input parameter dari methode di atas adalah kumpulan *pixel* suatu citra yang direpresentasikan dalam bentuk *array* integer dan pesan rahasia dalam bentuk String. Yang dilakukan program di atas pada intinya adalah melakukan looping pada tiap *pixel*, dan pada tiap *pixel* tersebut, disisipkan 1 *bit* pesan pada tiap channel *RGB* nya.

b. *Extract*ing

```
public static String extractMessage(byte[] oneDPix, int imgCols,int
imgRows) {

        StringBuilder strbuilder=new StringBuilder();

        String builder = "";
        int shiftIndex = 8;
        byte tmp = 0x00;
        for (int i = 0; i < oneDPix.length; i++) {
tmp = (byte) (tmp | ((oneDPix[i] << toShift[shiftIndex % toShift.length]) &
andByte[shiftIndex++ % toShift.length]));
          if (shiftIndex % toShift.length == 0) {

              byte[] nonso = { tmp };
              String str = new String(nonso);

              builder=strbuilder.toString();
              if (builder.endsWith(END_MESSAGE_COSTANT)) {
                    break;
              } else {
                    strbuilder.append(str);

                    builder=strbuilder.toString();

                    if (builder.length() ==
START_MESSAGE_COSTANT.length()
&& !START_MESSAGE_COSTANT.equals(builder)) {
                          builder = "";
                          break;
                    }
              }

              tmp = 0x00;
          }

        }

        if (!builder.equals(""))
            builder =
builder.substring(START_MESSAGE_COSTANT.length(), builder
                    .length()
                    - END_MESSAGE_COSTANT.length());
        return builder;

}
```

Input parameter dari program di atas adalah kumpulan *pixel* suatu citra yang direpresentasikan dalam bentuk *array byte*. Yang dilakukan program di atas pada intinya adalah melakukan looping pada tiap *array byte*, dan pada tiap *array byte* tersebut, diambil 1 *bit* pesan. Ketika sudah terkumpul 8 *bit*, kemudian dibentuk menjadi 1 character yang akhir nya digabung menjadi suatu kata/kalimat.