# BAB IV

# HASIL DAN PEMBAHASAN

Bab ini menunjukkan hasil penelitian dan pembahasan tentang aplikasi pengamanan untuk Basisdata dengan metode Steganografi. Ada beberapa hal yang menjadi batasan implementasi, yaitu :

1. Tampilan aplikasi menggunakan bahasa Indonesia, kecuali bagian tertentu yang dianggap penulis lebih baik menggunakan bahasa asing.
2. Aplikasi steganografi untuk pengamanan basisdata dengan metode *pixel value differencing* (PVD) dan algoritma *rijndael* ini hanya sebatas pengolahan data anggota yang di enkripsi dan di sisipkan kedalam citra dan serta hasil output dari data anggota berupa citra yang telah di convert dari biner di database.

## 4.1. Hasil Perancangan

Hasil dari perancangan ini adalah perangkat lunak aplikasi steganografi sebagai impementasi dari metode *pixel value differencing* (PVD) dan algoritma *rijndael*.

### 4.1.1 Implementasi Perangkat Lunak

Pada implementasinya aplikasi atau program ini, spesifikasi perangkat lunak yang digunakan sepeti pada tabel Tabel 4.1.

Tabel 4.1 Spesifikasi Perangkat Lunak

| **Perangkat Lunak** | **Keterangan** |
|---|---|
| Sistem Operasi | Kali Linux x64 |
| Bahasa Pemrograman | Java |
| *Tools* Pemrogaman | Java Netbeans IDE 7.4 |
| Perangkat Pendukung | JDK 7 |
| *Picture Viewer* | *Image Viewer* |
| *Document Viewer* | *Libre Office* |
| *Picture* Editor | *Gimp* |

### 4.1.2 Implementasi Basis Data

Implementasi basis data dilakukan dengan menggunakan bahasa SQL, dimana DBMS yang digunakan adalah MySQL.

1. Tabel Anggota

```
CREATE TABLE IF NOT EXISTS `us_manggota` (
  `us_maID` varchar(10) NOT NULL,
  `us_maNAME` varchar(45) NOT NULL,
  `us_maIMAGE` longblob NOT NULL,
  PRIMARY KEY (`us_maID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

### 4.1.3 Implementasi Antarmuka

Perangkat lunak yang dihasilkan merupakan realisasi desain tampilan antarmuka dari setiap menu yang telah dirancang sebelumnya kedalam bahasa

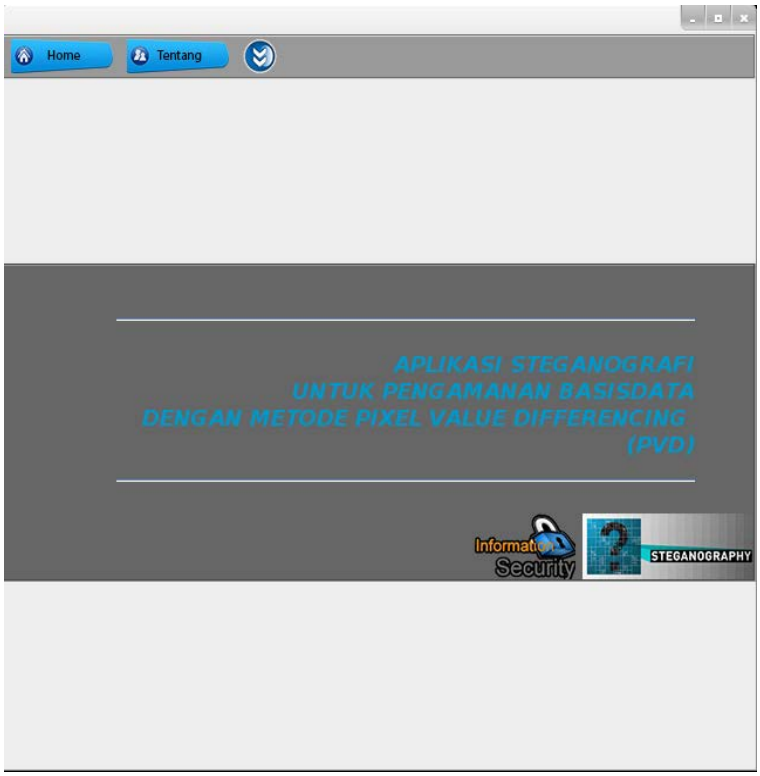
pemograman. Berikut ini adalah hasil dari implementasi antarmuka aplikasi yang terlihat pada Tabel 4.2.

Tabel 4.2 Hasil Implementasi Antarmuka

| **Menu** | **Proses** | ***File*** |
|---|---|---|
| *Home* | - | Fmain.java |
| Pelayanan<br><br>- Manajemen Anggota<br><br>*#Input*<br><br>#Penyisipan | Input data anggota, pemilihan pesan, enkripsi, pembuatan kunci, penyalinan kunci, pemilihan citra, *view imange*, *embed(save to database*). | Formin.java, Pesan.java, CoverImage.java, PvdMethod.java, Rijndael.java, KeySchedule.java |
| Pelaporan<br><br>- Data Anggota<br><br>*#Output*<br><br>#Ekstrak | Get data, *view image*, pilih cita stego, ekstrak, *save file*, masukan kunci, dekripsi. | FormOut.java, Pesan.java, CoverImage.java, PvdMethod.java, Rijndael.java, KeySchedule.java |
| Tentang | - | fTentang.java |
| *Options*<br><br>*#Login*<br><br>*#Logout*<br><br>*#Exit* | Masukan *username* dan *password* | Fmain.java |

Antarmuka dari aplikasi steganografi ini, terdiri dari lima menu utama, yaitu menu *home*, menu pelayanan, menu pelaporan, menu tentang, dan menu *options*. Masing – masing tampilan antarmuka tersebut diberikan pada gambar 4.1, gambar 4.2, gambar 4.3, gambar 4.4, gambar 4.5, gambar 4.6, gambar 4.7, gambar 4.8, dan gambar 4.9.

Gambar 4.1 “Tampilan Menu *Home*” menunjukan hasil dari perancangan antarmuka *home* pada aplikasi steganografi.



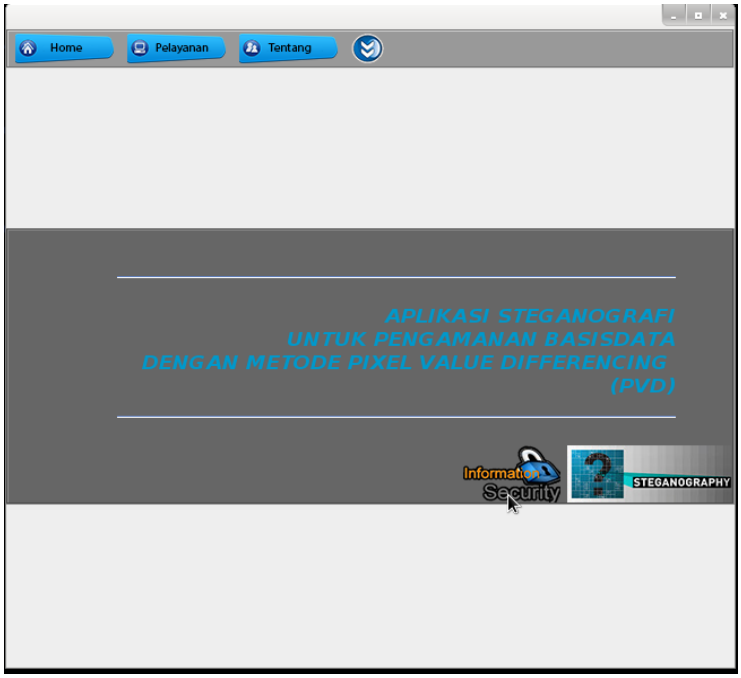

Gambar 4.1 Tampilan Menu *Home*

Gambar 4.2 “Tampilan Menu *Login*” menunjukan hasil dari perancangan antarmuka l*ogin* pada aplikasi steganografi.



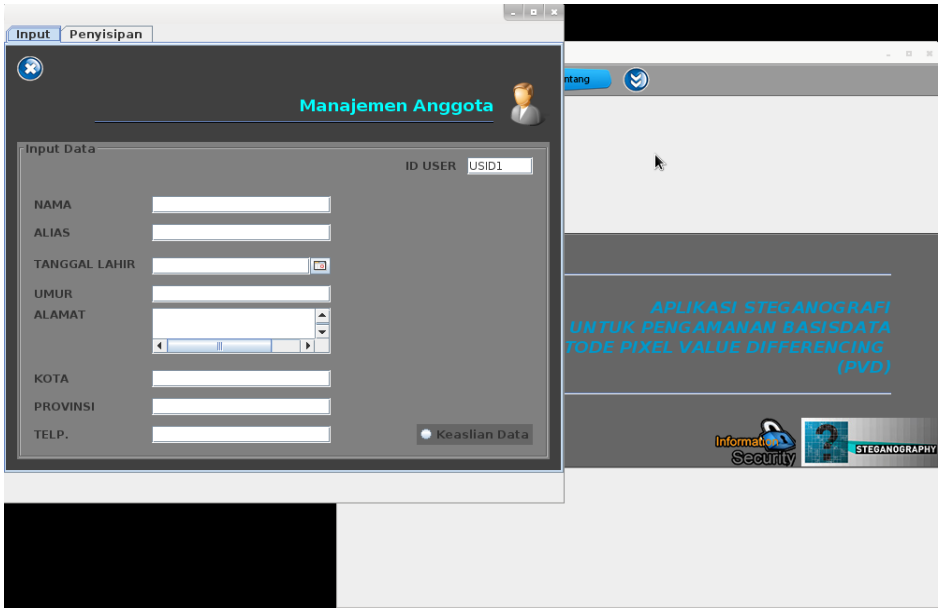
Gambar 4.2 Tampilan Menu *Login*

Gambar 4.3 “Tampilan Menu *Home* Admin” menunjukan hasil dari perancangan antarmuka *home* admin pada aplikasi steganografi.



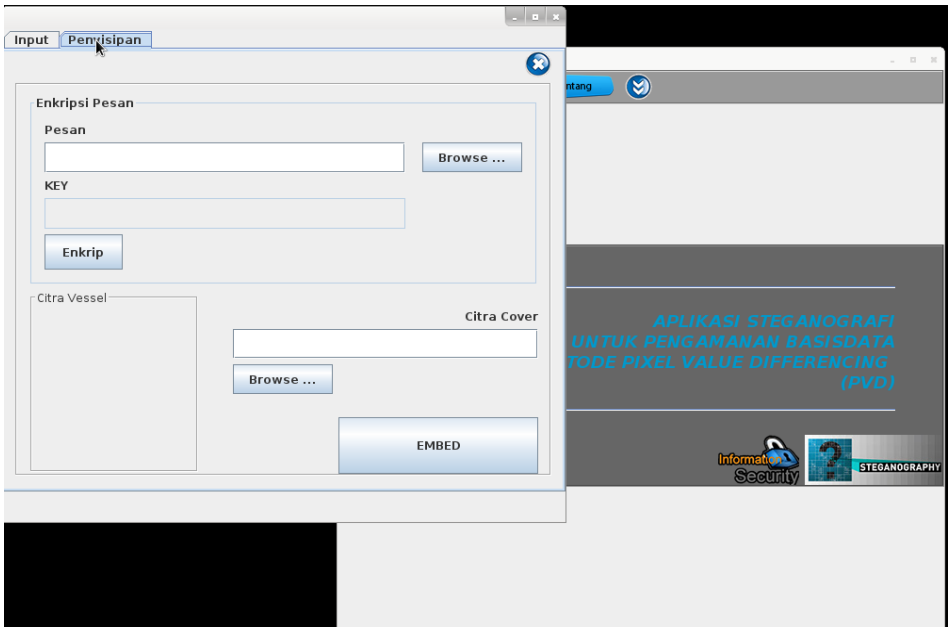

Gambar 4.3 Tampilan Menu *Home* Admin

Gambar 4.4 “Tampilan Menu Pelayanan Input Data” menunjukan hasil dari perancangan antarmuka *home* admin pada aplikasi steganografi.



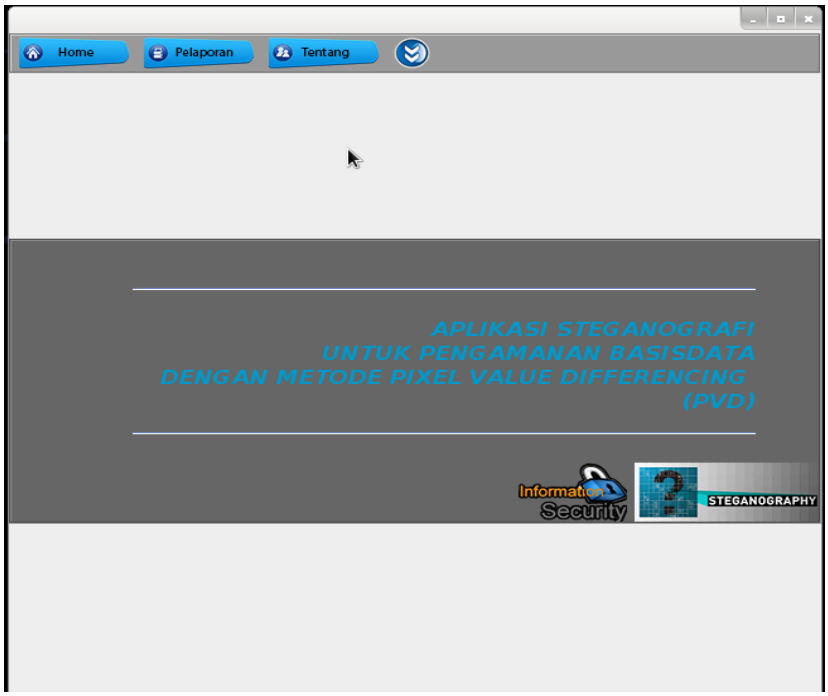

Gambar 4.4 Tampilan Menu Pelayanan Input Data

Gambar 4.5 “Tampilan Menu Pelayanan Penyisipab Data” menunjukan hasil dari perancangan antarmuka *home* admin pada aplikasi steganografi.



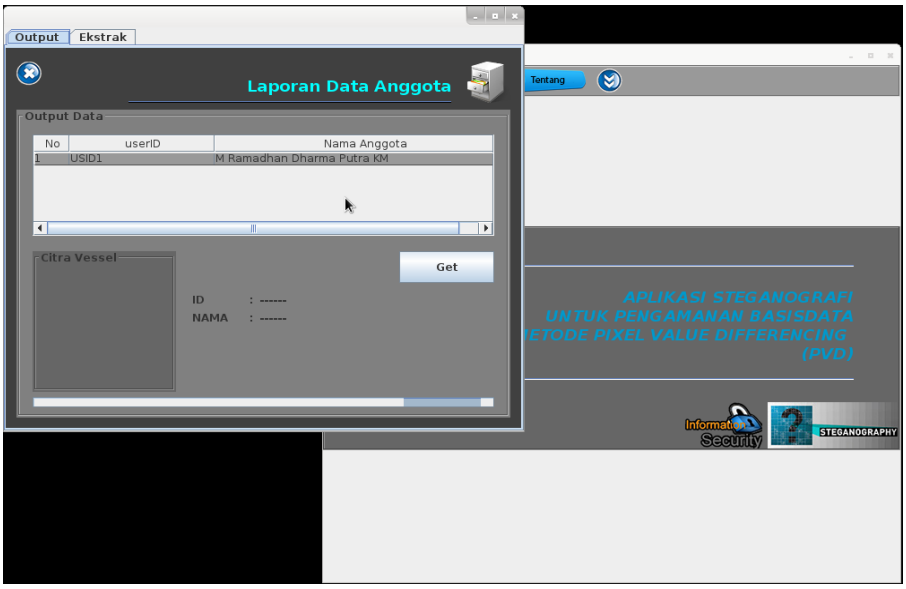

Gambar 4.5 Tampilan Menu Pelayanan Penyisipan Data

Gambar 4.6 “Tampilan Menu *Home* Sirkuasi” menunjukan hasil dari perancangan antarmuka *home* admin pada aplikasi steganografi.



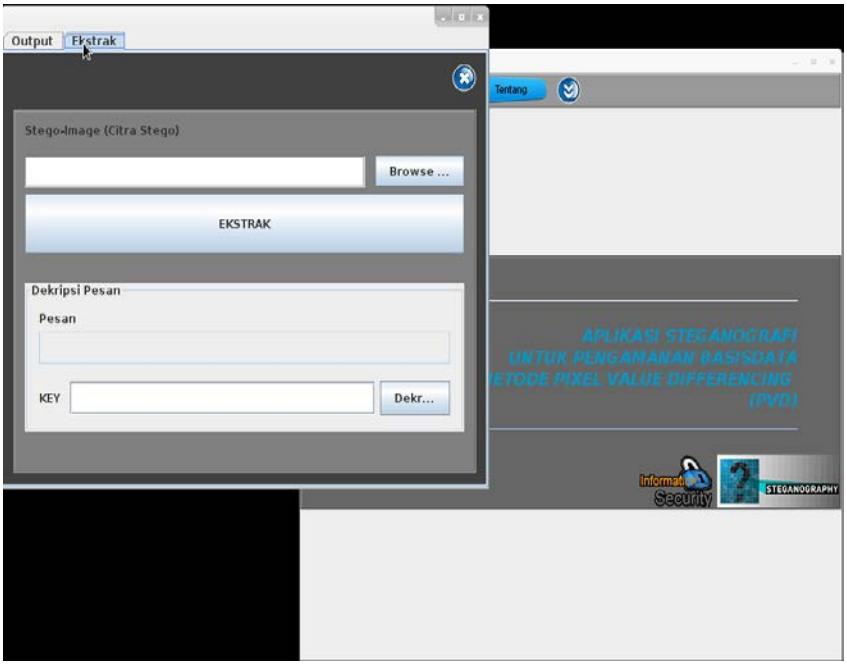

Gambar 4.6 Tampilan Menu *Home* Sirkuasi

Gambar 4.7 “Tampilan Menu Pelaporan Sirkuasi” menunjukan hasil dari perancangan antarmuka Pelaporan pada aplikasi steganografi.




Gambar 4.7 Tampilan Menu Pelaporan Sirkuasi

Gambar 4.8 “Tampilan Menu Pelaporan Sirkuasi Ekstrak” menunjukan hasil dari perancangan antarmuka Pelaporan pada aplikasi steganografi.




Gambar 4.8 Tampilan Menu Pelaporan Sirkuasi Ekstrak

Gambar 4.9 “Tampilan Menu Tentang” menunjukan hasil dari perancangan antarmuka Pelaporan pada aplikasi steganografi.




Gambar 4.9 Tampilan Menu Tentang

## 4.2. Pembahasan

Pembahasan penelitian ini dilakukan dengan hasil simulasi terhadap satu *file* data.doc dan *File* citra cover 4lena.bmp dan seperti pada Tabel 4.3. *File* data.doc adalah file hasil dari inputan data yang dilakukan, sedangkan cover 4lena.bmp adalah citra penampung pesan atau file yang akan di sisipkan dan di ekstrak. Serta *file* citra stego yang telah di sisipkan data di uji ketahanannya.

Tabel 4.3 *File* Yang Digunakan Dalam Simulasi

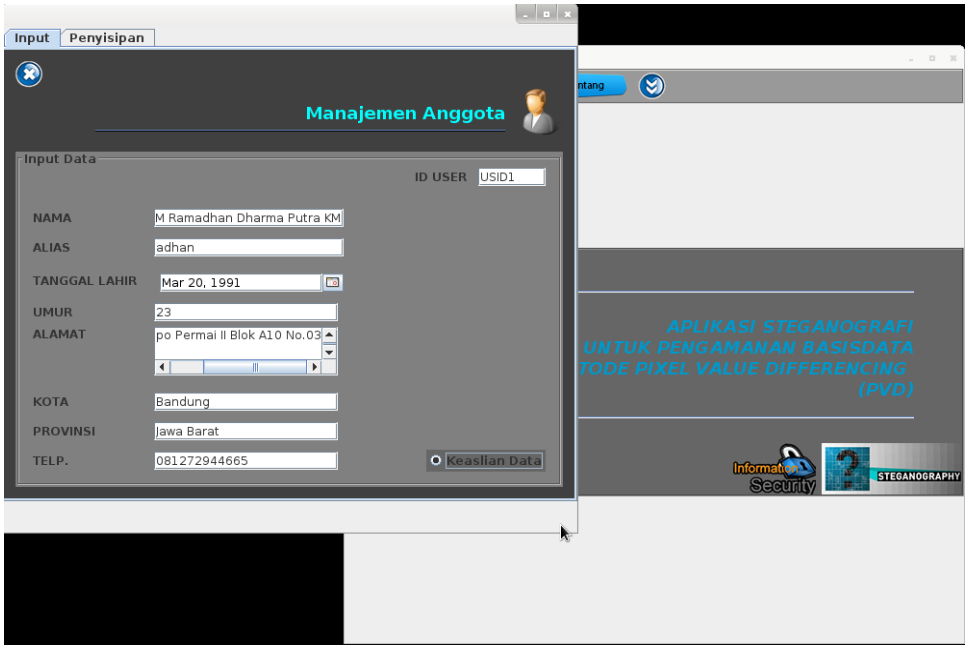
| Nama File | Ekstensi File |
|---|---|
| data | .doc |
| 4lena | .bmp |

### 4.2.1. Simulasi Pelayanan

Berikut ini disimulasikan penginputan data anggota dengan hasil file data.doc yang akan di enkripsi menggunakan algoritma rijndael dan penyisipan dengan metode *PVD* pada citra cover 4lena.bmp.

#### 4.2.1.1. Proses Penginputan Data

Gambar 4.10 menampilan hasil proses penginputan data anggota dengan nama file data.doc yang secara otomatis akan tersimpan pada folder temp.

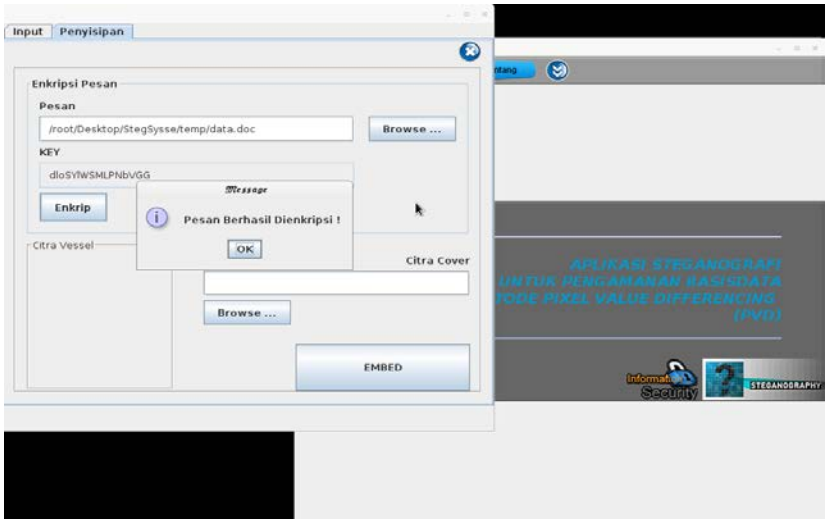



Gambar 4.10 Contoh Tampilan Proses Penginputan Data

Pada proses penginputan data anggota, terdiri dari isikan nama, alias, tanggal lahir, umur, alamat, kota, provinsi, telp, dan radio button. Karena semua field harus terisi.

### 4.2.1.2. Proses Enkripsi Data dan Penyisipan

Gambar 4.11 dan Gambar 4.12 menampilan hasil proses enkripsi data serta pembuatan kunci dan penyisipan dengan nama *file* data.doc yang akan disisipkan kedalam *file* 4lena.bmp. Sebelum *file* disisipkan, *file* data.doc di enkripsi dulu dan proses pembuatan kunci untuk mendekrip kembali isi *file* data.doc setelah itu baru *file* data.doc disisipkan kedalam citra 4lena.bmp. Data yang telah disisipkan kedalam citra 4lena.bmp tersimpan kedalam database MySQL dan file data.doc yang sebelumnya tersimpan di dalam folder temp ikut terhapus.




Gambar 4.11 Contoh Tampilan Proses Enkripsi dan Proses Pembuatan Kunci

Proses enkripsi *rijndael* dilakukan pada algoritma enkripsi seperti pada Tabel 4.4 dan proses pembuatan kunci *rijndael* dilakukan pada algoritma *key schedule* seperti pada Tabel 4.5.

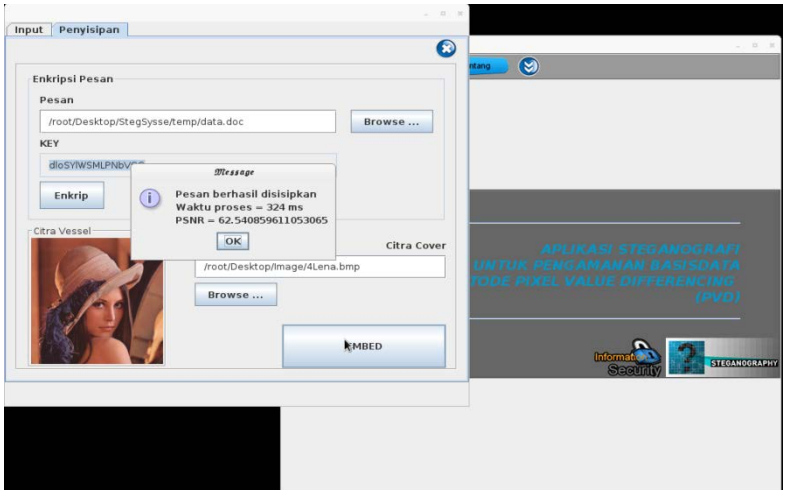
Tabel 4.4 Algoritma Enkripsi *Rijndael*

| ***Line*** | ***Source*** |
|---|---|
| 1 | byte[] tmp = new byte[input.length]; |
| 2 | byte[][] state = new byte[4][keySchedule.getNb()]; |
| 3 | for (int i = 0; i < input.length; i++) { |
| 4 | state[i / 4][i % 4] = input[i % 4 * 4 + i / 4]; |
| 5 | } |
| 6 | state = addRoundKey(state, w, 0); |
| 7 | for (int round = 1; round < keySchedule.getNr(); round++) { |
| 8 | state = subBytes(state); |
| 9 | state = shiftRows(state); |
| 10 | state = mixColumns(state); |
| 11 | state = addRoundKey(state, w, round); |
| 12 | } |
| 13 | state = subBytes(state); |
| 14 | state = shiftRows(state); |
| 15 | state = addRoundKey(state, w, keySchedule.getNr()); |
| 16 | for (int i = 0; i < tmp.length; i++) { |
| 17 | tmp[i % 4 * 4 + i / 4] = state[i / 4][i % 4]; |
| 18 | } |
| 19 | return tmp; |

Tabel 4.5 Algoritma *Key Schedule Rijndael*

| ***Line*** | ***Source*** |
|---|---|
| 1 | byte[][] tmp = new byte[nb * (nr + 1)][4]; |
| 2 | int i = 0; |
| 3 | while (i < nk) { |
| 4 | tmp[i][0] = key[i * 4]; |
| 5 | tmp[i][1] = key[i * 4 + 1]; |
| 6 | tmp[i][2] = key[i * 4 + 2]; |
| 7 | tmp[i][3] = key[i * 4 + 3]; |
| 8 | i++; |
| 9 | } |
| 10 | i = nk; |
| 11 | while (i < nb * (nr + 1)) { |
| 12 | byte[] temp = new byte[4]; |
| 13 | for (int k = 0; k < 4; k++) { |
| 14 | temp[k] = tmp[i – 1][k]; |
| 15 | } |
| 16 | if (i % nk == 0) { |
| 17 | temp = subWord(rotWord(temp)); |
| 18 | temp[0] = (byte) (temp[0] ^ (rCon[i / nk] & 0xff)); |
| 19 | } else if (nk > 6 && i % nk == 4) { |
| 20 | temp = subWord(temp); |
| 21 | } |
| 22 | tmp[i] = xorFunc(tmp[i - nk], temp); |
| 23 | i++; |
| 24 | } |
| 25 | return tmp; |

Gambar 4.12 menampilan proses penyisipan data yang telah di enkripsi, lalu di sisipkan kedalam citra.



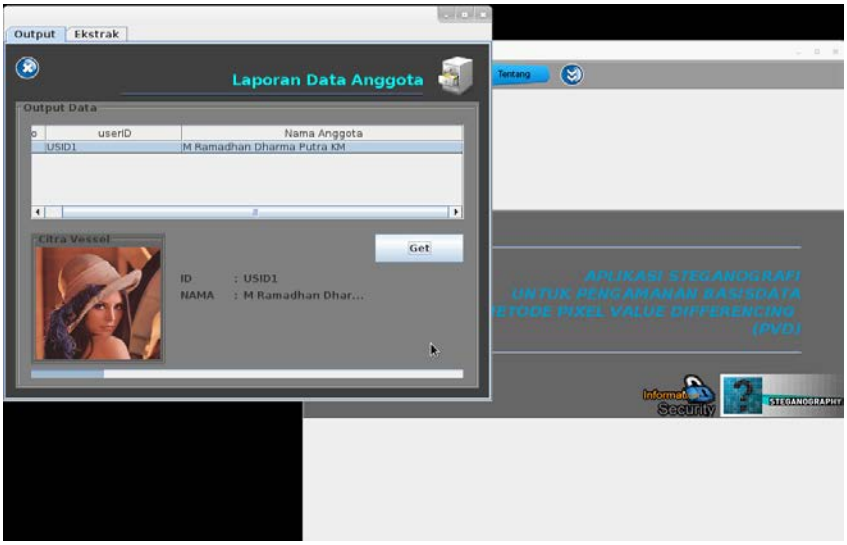

Gambar 4.12 Contoh Tampilan Proses Penyisipan

### 4.2.2 Simulasi *Output* Data Anggota

Berikut ini disimulasikan pelaporan data anggota dengan hasil file 4lena.bmp yang akan di ekstrak dengan metode *PVD* pada citra cover 4lena.bmp dan dekripsi data hasil ekstrak yaitu file data.doc menggunakan algoritma rijndael.

#### 4.2.2.1. Proses *Output* Data

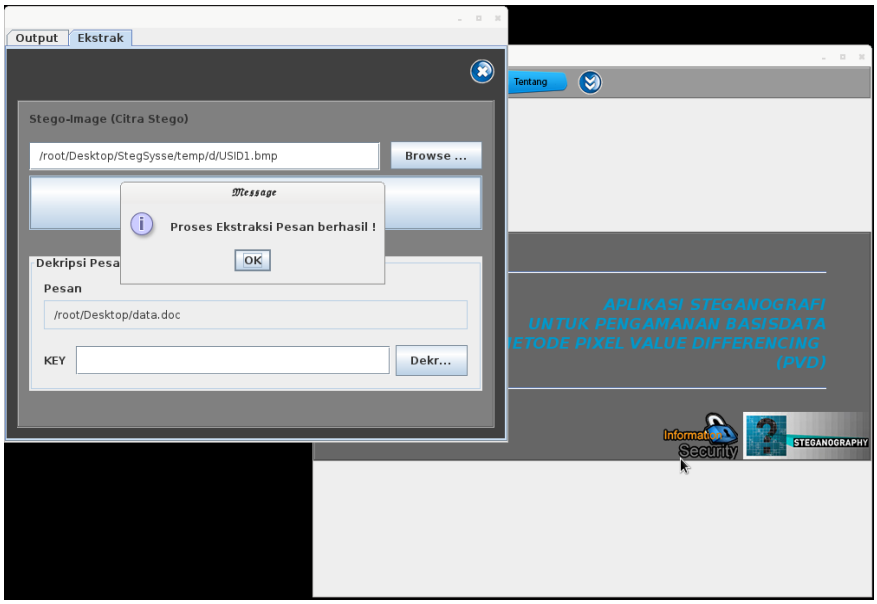
Gambar 4.13 menampilan hasil proses output data anggota dengan nama file 4lena.bmp yang secara otomatis akan tersimpan pada folder downloads pada saat data kita get untuk di ekstrak.




Gambar 4.13 Contoh Tampilan Proses Output Data

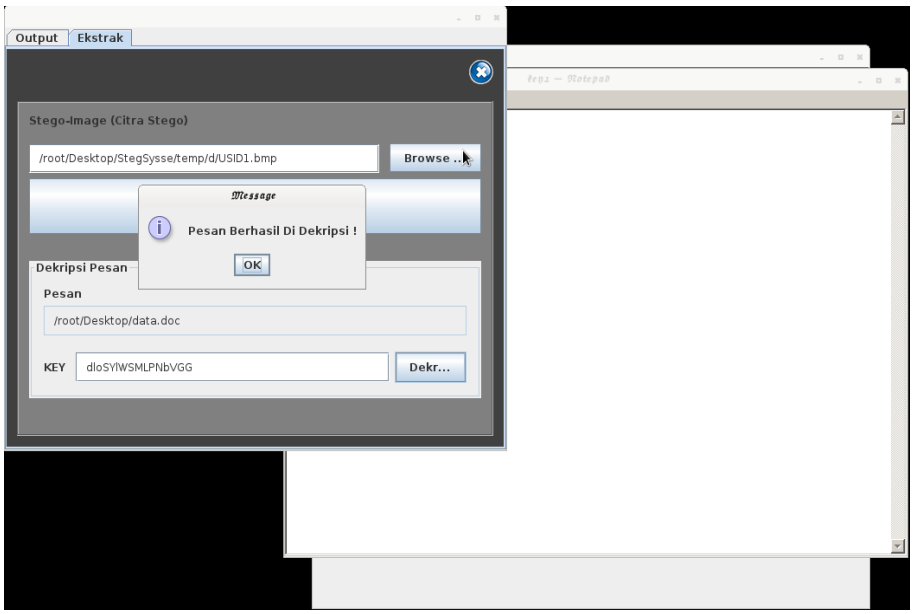
4.2.2.2. Proses Ekstrak Dan Dekripsi data

Gambar 4.14 menampilan hasil proses ekstraksi data anggota dengan nama file 4lena.bmp menjadi data.doc.




Gambar 4.14 Contoh Tampilan Proses Ekstraksi Data

Gambar 4.15 menampilan hasil proses dekripsi data anggota dengan file yang digunakan data.doc. File data.doc adalah data hasil ekstrakan yang sebelumnya di lakukan. Sebelum file didekrip untuk dijadikan seperti semula, kita harus memasukkan key yang telah di generated sebelumnya. Dan proses dekripsi rijndael dilakukan pada algoritma dekripsi seperti pada Tabel 4.6.




Gambar 4.15 Contoh Tampilan Proses Dekripsi Data

Tabel 4.6 Algoritma Dekripsi Rijndael

| ***Line*** | ***Source*** |
|---|---|
| 1 | byte[] tmp = new byte[input.length]; |
| 2 | byte[][] state = new byte[4][keySchedule.getNb()]; |
| 3 | for (int i = 0; i < input.length; i++) { |
| 4 | state[i / 4][i % 4] = input[i % 4 * 4 + i / 4]; |
| 5 | } |
| 6 | state = addRoundKey(state, w, keySchedule.getNr()); |
| 7 | for (int round = keySchedule.getNr() - 1; round >= 1; round--) { |
| 8 | state = invSubBytes(state); |
| 9 | state = invShiftRows(state); |
| 10 | state = addRoundKey(state, w, round); |
| 11 | state = invMixColumns(state); |
| 12 | } |
| 13 | state = invSubBytes(state); |
| 14 | state = invShiftRows(state); |
| 15 | state = addRoundKey(state, w, 0); |
| 16 | for (int i = 0; i < tmp.length; i++) { |
| 17 | tmp[i % 4 * 4 + i / 4] = state[i / 4][i % 4]; |
| 18 | } |
| 19 | return tmp; |

### 4.2.3 Uji Ketahanan

Pengujian ketahanan dilakukan untuk mengetahui ketahanan pesan yang telah tersisipi didalam citra, jika dilakukan manipulasi terhadap citra, dan untuk mengetahui ketahanan terhadap aplikasi *steganalysis.* Manipulasi yang dilakukan terhadap citra yaitu dengan melakukan *rotate left*, *rotate right*, *rotate 180o*, *resize -5%, Grayscale*, dan *Croping*. Pengujian ketahanan dilakukan terhadap format citra Bitmap.

4.2.3.1 Uji Ketahanan Citra Bitmap

Hasil pengujian ketahanan pada citra bitmap terhadap manipulasi dapat dilihat pada tabel 4.7.

Tabel 4.7 Pengujian Ketahanan Terhadap Manipulasi Citra Bitmap

| **Citra Bitmap** | **Rotate Left** | **Rotate Right** | **Rotate 180o** | **Resize -5%** | **Grayscale** | **Crop** |
|---|---|---|---|---|---|---|
| 4lena | Gagal | Gagal | Gagal | Gagal | Gagal | Gagal |

Pengujian ketahanan terhadap aplikasi *Steganalysis* dilakukan menggunakan *tools* StegSpy2.1. Pengujian ini untuk mengetahui apakah pesan dapat terdeteksi oleh aplikasi steganalysis, atau tidak. Hasil pengujian seperti pada table 4.8.

Tabel 4.8 Pengujian Terhadap Aplikasi *Steganalysis* Citra Bitmap

| **Citra** | **Hasil Deteksi** |
|---|---|
| 4lena.bmp | Terdeteksi |

Pengujian ketahanan terhadap manipulasi yang dilakukan dengan hasil seperti pada tabel 4.7, menunjukan bahwa citra *bitmap* tidak tahan terhadap manipulasi. Sedangkan pada tabel 4.8 pengujian terhadap aplikasi *steganalysis*, citra *bitmap* yang tersisipi pesan dapat terdeteksi oleh aplikasi *steganalysis*.

**4.2.4 Hasil Simulasi**

Dari simulasi yang telah dilakukan dapat dilihat pada Tabel 4.9 Kapasitas Citra Bitmap dan Tabel 4.10 Hasil File Enkrip dan Dekrip. Serta metode *pixel*

*value differencing* (PVD) tidak dapat mengekstrak citra yang telah dimanipulasi, dan metode PVD tidak tahan terhadap *steganalysis*.

Tabel 4.9 Kapasitas Citra *Bitmap*

| NO | Nama Citra | Resolusi | Bit Citra | Ukuran Citra (Byte) | Kapasitas (Byte) | Presentase |
|---|---|---|---|---|---|---|
| 1 | 4lena.bmp | 512x512 | 24 bit | 786,486 | 51,556 | 6.55% |

Bedasarkan pada Tabel 4.9, citra *bitmap* mempunyai ukuran yang lebih kecil dari citra aslinya, presentase kapasitas penyimpanan pesan pada citra bitmap yaitu 6.25% dari ukuran citra nya.

Tabel 4.10 Hasil File Enkip dan Dekrip

| Enkrip | Dekrip |
|---|---|
| V◆}◆◆ 0^◆ □◆IhtM+◆◆ ##◆X ◆◆ 6v#V□^◆co◆vC◆◆◆◆ Qe◆ )◆V◆}◆◆ 0^◆ □◆IhtM+.◆◆◆ 9 ◆◆ \|50◆n=◆◆◆ #◆◆◆ #◆◆ #◆e' ◆m◆#@◆E◆◆◆◆◆ m#◆◆ C##z #D◆◆ d◆#?(◆1H◆k◆G□d◆◆◆ ◆F◆OWI □ ◆w◆y◆1◆*e'◆◆ :&◆2◆V`#◆L ◆ơ◆◆◆ 6LX◆:yi◆#◆m◆x###lk ◆T◆$W- #◆◆◆ □n◆◆ #q◆◆◆◆ ##◆ñ◆◆ 6 ◆#◆◆ .◆J 劍 ◆'y◆K◆]◆A x◆:]◆)P◆◆ [◆G | ==========<br>LAPORAN DATA ANGGOTA<br>==========<br><br>Id : USID1<br>Nama : M Ramadhan Dharma Putra KM<br>Alias : Adhan<br>Tanggal : 20-03-1991<br>Umur : 23<br>Alamat : Kopo Permai II Blok A10 No.3<br>Kota : Bandung<br>Provinsi : Jawa Barat<br>Telp : 0812 72 944 665 |