**BAB V**

**IMPLEMENTASI DAN PENGUJIAN SISTEM**

* 1. **Implementasi**

Implementasi perancangan antarmuka terbagi menjadi empat bagian utama yaitu *form* *Main Menu*, *form* *Encrypt-Embed*, *form* *Retrieve-Decrypt*, dan *form* *Help*.

1. *Form* *Main Menu*

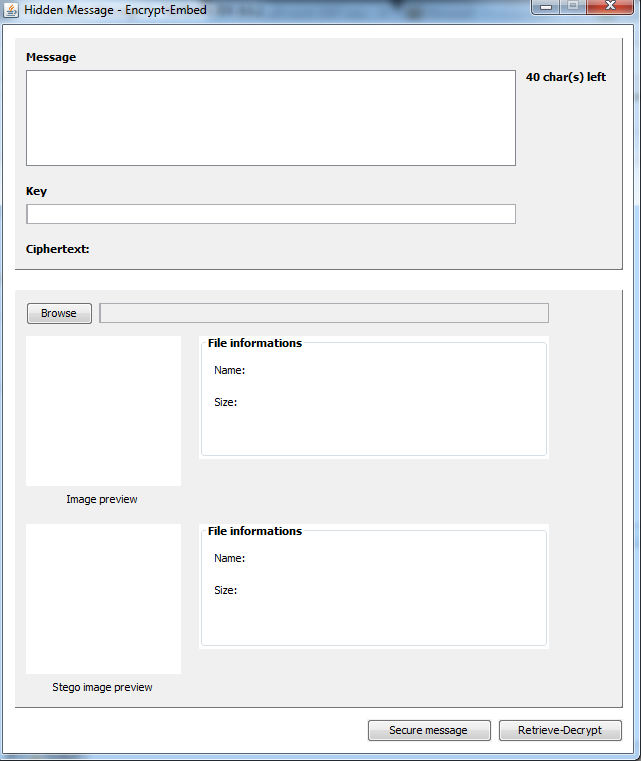
Pada *form* ini *user* dapat memilih menu yang diinginkan sesuai dengan kebutuhan. Terdapat tiga menu yaitu *Encrypt-Embed*, *Retrieve-Decrypt*, dan *Help*.



**Gambar 5.1 *Form* *Main Menu***

1. *Form* *Encrypt-Embed*

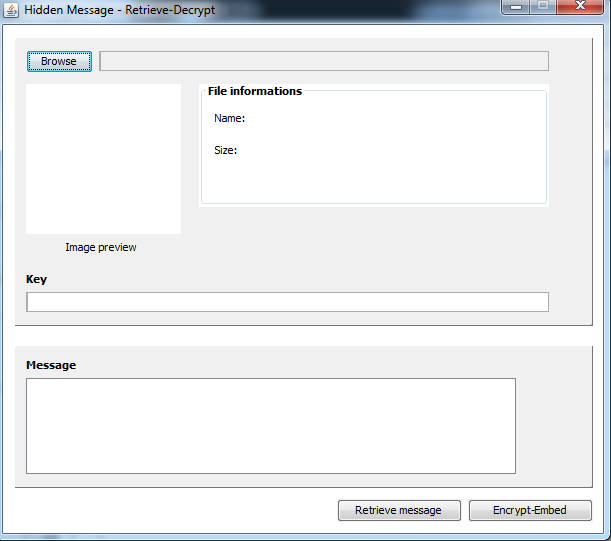
Pada *form* ini, *user* memasukkan pesan dan kunci enkripsi yang diinginkan serta memilih citra yang akan disisipi pesan dengan menekan tombol *Browse*. Selanjutnya, untuk melakukan proses enkripsi dan *embedding* *user* menekan tombol *Secure Message*. Hasil dari proses enkripsi dan *embedding* berupa *stego image* akan ditampilkan pada panel *Stego Image Preview*.



**Gambar 5.2 *Form* *Encrypt-Embed***

1. *Form* *Retrieve-Decrypt*

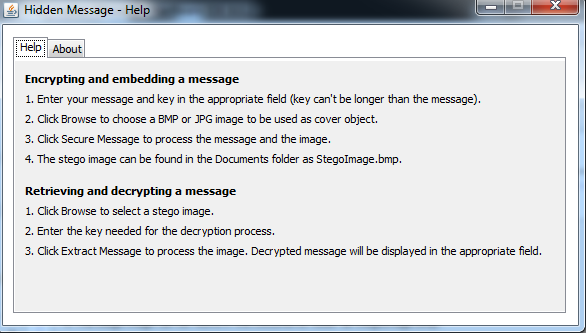
Pada *form* ini *user* dapat melakukan proses *retrieve* dan dekripsi pesan dengan cara memilih *stego image* menggunakan tombol *Browse*, kemudian memasukkan kunci dekripsi. Kemudian, *user* menekan tombol *Retrieve Message* dan pesan yang telah terdekripsi akan ditampilkan pada *field* yang tersedia.

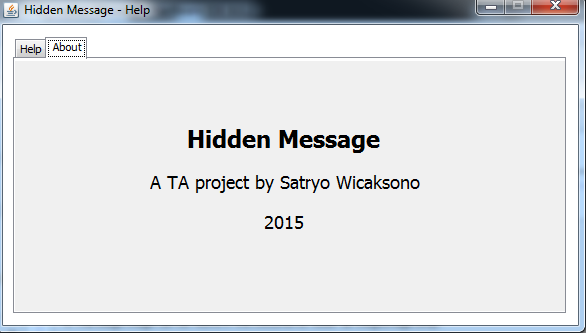
****

**Gambar 5.3 *Form* *Retrieve-Decrypt***

1. *Form* *Help*

*Form* ini menampilkan petunjuk pemakaian aplikasi untuk memudahkan *user* menggunakan aplikasi.





**Gambar 5.4 *Form* *Help***

* 1. **Analisis Hasil**

Pada tahap ini akan dijelaskan analisis hasil kinerja dari aplikasi *Hidden Message*.

* + 1. **Analisis Proses Enkripsi dan *Embedding***

Untuk melakukan enkripsi dan *embedding*, mula-mula *user* memasukkan pesan dan kunci enkripsi, kemudian *user* menekan tombol *Secure Message*. Sistem akan melakukan proses enkripsi dan *embedding* menggunakan algoritma Vigenere *cipher* dan *Pixel Value Differencing*.

Contoh:

Pesan : Informatika

Kunci enkripsi : Teknik

*Ciphertext* : bQOaYVtWRWH

Nama/ukuran citra asli : hyunyoung.jpg/16 KB

Nama/ukuran *stego image* : StegoImage.bmp/219 KB

Proses enkripsi pesan:

1. Pada contoh di atas, dapat dilihat bahwa panjang kunci lebih kecil dari panjang *plaintext* (jumlah karakter kunci lebih kecil dari jumlah karakter *plaintext*) sehingga perlu dilakukan dilakukan *loop* kunci untuk menyamakan panjang keduanya. Hasil dari *loop* kunci adalah sebagai berikut.

*Plaintext* : Informatika

Kunci : TeknikTekni

1. Tahap berikutnya adalah menginisialisasi nilai-nilai seluruh karakter. Pemberian nilai-nilai karakter dilakukan mulai dari karakter A dengan nilai 0 sampai dengan karakter *space* (spasi) dengan nilai 52.
2. *Ciphertext* didapatkan dengan memasangkan masing-masing karakter *plaintext* dengan karakter kunci, kemudian melakukan operasi dengan menggunakan persamaan Ci = Pi + Ki. Pada contoh di atas, karakter I dipasangkan dengan karakter T sehingga prosesnya adalah sebagai berikut.

C1 = P1 + K1

= 8 + 19

= 27

1. Karakter yang memiliki nilai 27 yaitu b, sehingga enkripsi karakter I dengan kunci T akan menghasilkan karakter b. Yang perlu diperhatikan adalah apabila nilai hasil operasi yang diperoleh berada di luar *range* 0-52 maka perlu dilakukan operasi penjumlahan atau pengurangan terhadap nilai tersebut untuk mendapatkan nilai yang berada di dalam *range*. Proses ini dilakukan pada semua karakter *plaintext* seperti di bawah ini.

C2 = P2 + K2

= 39 + 30

= 69

= 69 - 53 = 16

= Q

C3 = P3 + K3

= 31 + 36

= 67

= 67 - 53 = 14

= O

C4 = P4 + K4

= 40 + 39

= 79

= 79 - 53 = 26

= a

C5 = P5 + K5

= 43 + 34

= 77

= 77 - 53 = 24

= Y

C6 = P6 + K6

= 38 + 36

= 74

= 74 - 53 = 21

= V

C7 = P7 + K7

= 26 + 19

= 45

= t

C8 = P8 + K8

= 45 + 30

= 75

= 75 - 53 = 22

= W

C9 = P9 + K9

= 34 + 36

= 70

= 70 - 53 = 23

= R

C10 = P10 + K10

= 36 + 39

= 75

= 75 - 53 = 22

= W

C11 = P11 + K11

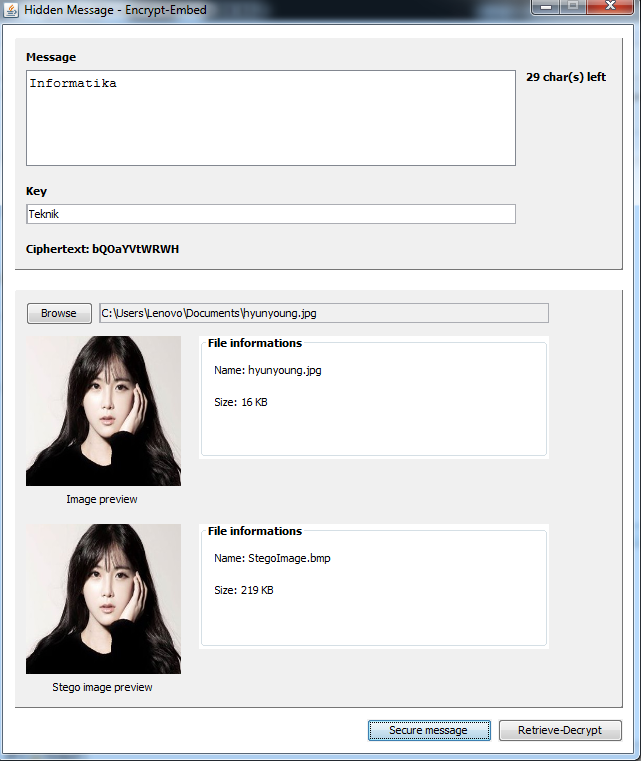
= 26 + 34

= 60

= 60 - 53 = 7

= H

1. Hasil enkripsi *plaintext* Informatika dengan kunci Teknik yaitu bQOaYVtWRWH.



**Gambar 5.5 Analisis proses enkripsi dan *embedding***

*Source code* proses enkripsi dan *embedding* ditunjukkan oleh gambar 5.6.

String plain=jTextArea1.getText();

String key=jTextField1.getText();

char[] nPlain=plain.toCharArray();

char[] nKey=key.toCharArray();

int i=0;

while(key.length()<plain.length()) {

key=key+nKey[i];

nKey=key.toCharArray();

i=i+1;

}

int[] kodePlain=new int[plain.length()];

int[] kodeKey=new int[kodePlain.length];

int[] kodeCipher=new int[kodePlain.length];

String cipher="";

for(int counter=0; counter<plain.length(); counter++) {

for(NilaiKarakterEnkripsi nke:

NilaiKarakterEnkripsi.values()) {

if(String.valueOf(nPlain[counter]).

equals(String.valueOf(nke)))

kodePlain[counter]=nke.getNilai();

else if(String.valueOf(nPlain[counter]).equals(" "))

kodePlain[counter]=nke.space.getNilai();

if(String.valueOf(nKey[counter]).

equals(String.valueOf(nke)))

kodeKey[counter]=nke.getNilai();

else if(String.valueOf(nKey[counter]).equals(" "))

kodeKey[counter]=nke.space.getNilai();

}

kodeCipher[counter]=kodePlain[counter]+kodeKey[counter];

if(kodeCipher[counter]>52)

kodeCipher[counter]=kodeCipher[counter]-53;

cipher=cipher+NilaiKarakterEnkripsi.get(kodeCipher[counter]);

cipher=cipher.replaceAll("space", " ");

byte[] kodeAscii=new byte[plain.length()];

String stringBiner="";

for(int counter=0; counter<plain.length(); counter++) {

kodeAscii=cipher.getBytes(StandardCharsets.US\_ASCII);

stringBiner=stringBiner+String.format("%8s",

Integer.toBinaryString(kodeAscii[counter] &

0xFF)).replace(' ', '0');

}

int[] params=new int[2];

int[] pixelBaru=new int[2];

int p1=0;

int p2=0;

int d1=0;

int d2=0;

int t=0;

int b=0;

int m=0;

int start=0;

int finish=0;

int indeksAkhir=0;

String fragmenBiner="";

for(int counter=0; counter<arrayPixel1DBlue.length; counter++) {

if(counter%2!=0) {

p1=arrayPixel1DBlue[counter-1];

p2=arrayPixel1DBlue[counter];

d1=Math.abs(p1-p2);

params=parameterRange(d1);

t=(int)(Math.log(params[0])/Math.log(2));

finish=start+t;

if(finish<stringBiner.length())

fragmenBiner=stringBiner.substring(start, finish);

else

fragmenBiner=stringBiner.substring(start);

b=Integer.parseInt(fragmenBiner, 2);

d2=params[1]+b;

m=Math.abs(d2-d1);

pixelBaru=prosesNilaiPixel(p1, p2, d1, d2, m);

p1=pixelBaru[0];

p2=pixelBaru[1];

arrayPixel1DBlue[counter-1]=p1;

arrayPixel1DBlue[counter]=p2;

indeksAkhir=counter;

}

if(finish<stringBiner.length())

start=finish;

else

break;

}

**Gambar 5.6 *Source code* proses enkripsi dan *embedding***

* + 1. **Analisis Proses *Retrieve* dan Dekripsi**

Untuk melakukan *retrieve* dan dekripsi, *user* memilih *stego image* dan memasukkan kunci dekripsi. Selanjutnya sistem akan melakukan proses *retrieve* dan dekripsi dan menampilkan pesan terdekripsi pada *field* yang tersedia.

Contoh:

Nama/ukuran *stego image* : StegoImage.bmp/219 KB

Kunci dekripsi : Teknik

Pesan terdekripsi : Informatika

Proses dekripsi pesan:

1. Pada contoh di atas, dapat dilihat bahwa panjang kunci lebih kecil dari panjang *ciphertext* (jumlah karakter kunci lebih kecil dari jumlah karakter *ciphertext*) sehingga perlu dilakukan dilakukan *loop* kunci untuk menyamakan panjang keduanya. Hasil dari *loop* kunci adalah sebagai berikut.

*Ciphertext* : Informatika

Kunci : TeknikTekni

1. Tahap berikutnya adalah menginisialisasi nilai-nilai seluruh karakter. Pemberian nilai-nilai karakter dilakukan mulai dari karakter A dengan nilai 0 sampai dengan karakter *space* (spasi) dengan nilai 52.
2. *Plaintext* didapatkan dengan memasangkan masing-masing karakter *ciphertext* dengan karakter kunci, kemudian melakukan operasi dengan menggunakan persamaan Pi = Ci - Ki. Karakter pertama pada *ciphertext* yaitu b dipasangkan dengan karakter pertama kunci yaitu T sehingga prosesnya adalah sebagai berikut.

P1 = C1 - K1

= 27 - 19

= 8

1. Karakter yang memiliki nilai 8 yaitu I, sehingga dekripsi karakter b dengan kunci T akan menghasilkan karakter I. Yang perlu diperhatikan adalah apabila nilai hasil operasi yang diperoleh berada di luar *range* 0-52 maka perlu dilakukan operasi penjumlahan atau pengurangan terhadap nilai tersebut untuk mendapatkan nilai yang berada di dalam *range*. Proses ini dilakukan pada semua karakter *ciphertext* seperti di bawah ini.

P2 = C2 - K2

= 16 - 30

= -14

= -14 + 53 = 39

= n

P3 = C3 - K3

= 14 - 36

= -22

= -22 + 53 = 31

= f

P4 = C4 - K4

= 26 - 39

= -13

= -13 + 53 = 40

= o

P5 = C5 - K5

= 24 - 34

= -10

= -10 + 53 = 43

= r

P6 = C6 - K6

= 21 - 36

= -15

= -15 + 53 = 38

= m

P7 = C7 - K7

= 45 - 19

= 26

= a

P8 = C8 - K8

= 22 - 30

= -8

= -8 + 53 = 45

= t

P9 = C9 - K9

= 17 - 36

= -19

= -19 + 53 = 34

= i

P10 = C10 - K10

= 22 - 39

= -17

= -17 + 53 = 36

= k

P11 = C11 - K11

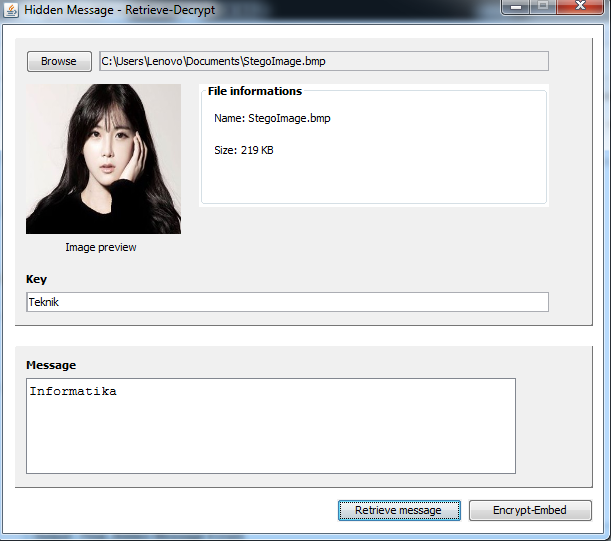
= 7 - 34

= -27

= -27 + 53 = 26

= a

1. Hasil dekripsi *ciphertext* bQOaYVtWRWH dengan kunci Teknik yaitu Informatika.



**Gambar 5.7 Analisis proses *retrieve* dan dekripsi**

*Source code* proses *retrieve* dan dekripsi ditunjukkan oleh gambar 5.8.

int[] params=new int[2];

int[] pixelBaru=new int[2];

int p1=0;

int p2=0;

int d1=0;

int d2=0;

int t=0;

int b=0;

String fragmenBiner="";

String stringBiner="";

for(int counter=0; counter<arrayPixel1DBlue.length; counter++) {

if(counter%2!=0) {

p1=arrayPixel1DBlue[counter-1];

p2=arrayPixel1DBlue[counter];

d1=Math.abs(p1-p2);

params=parameterRange(d1);

t=(int)(Math.log(params[0])/Math.log(2));

b=d1-params[1];

fragmenBiner=Integer.toBinaryString(b);

if(panjangPesan>1) {

if(fragmenBiner.length()<t && counter<indeksAkhir)

fragmenBiner=String.

format("%"+Integer.toString(t)+"s",

Integer.toBinaryString(b)).

replace(" ", "0");

if(fragmenBiner.length()<t && counter==indeksAkhir)

fragmenBiner=Integer.toBinaryString(b);

}

else {

if(fragmenBiner.length()<t && counter<indeksAkhir)

fragmenBiner=String.

format("%"+Integer.toString(t)+"s",

Integer.toBinaryString(b)).

replace(" ", "0");

if(fragmenBiner.length()<t && counter==indeksAkhir)

fragmenBiner="0"+Integer.toBinaryString(b);

}

stringBiner=stringBiner+fragmenBiner;

if(stringBiner.length()>=panjangPesan\*8)

break;

}

}

int multiplier=1;

int start=0;

int finish=0;

int kodeAscii=0;

String binerKeAscii="";

String cipher="";

for(int counter=0; counter<stringBiner.length(); counter++) {

if(counter==multiplier\*8-1) {

finish=multiplier\*8;

if(finish<stringBiner.length())

binerKeAscii=stringBiner.substring(start, finish);

else

binerKeAscii=stringBiner.substring(start);

kodeAscii=Integer.parseInt(binerKeAscii, 2);

cipher=cipher+(char)kodeAscii;

start=finish;

multiplier=multiplier+1;

}

}

String key=jTextField1.getText();

char[] nCipher=cipher.toCharArray();

char[] nKey=key.toCharArray();

int i=0;

while(key.length()<cipher.length()) {

key=key+nKey[i];

nKey=key.toCharArray();

i=i+1;

}

int[] kodeCipher=new int[cipher.length()];

int[] kodeKey=new int[kodeCipher.length];

int[] kodePlain=new int[kodeCipher.length];

String plain="";

for(int counter=0; counter<cipher.length(); counter++) {

for(NilaiKarakterDekripsi nkd:

NilaiKarakterDekripsi.values()) {

if(String.valueOf(nCipher[counter]).

equals(String.valueOf(nkd)))

kodeCipher[counter]=nkd.getNilai();

else if(String.valueOf(nCipher[counter]).equals(" "))

kodeCipher[counter]=nkd.space.getNilai();

if(String.valueOf(nKey[counter]).

equals(String.valueOf(nkd)))

kodeKey[counter]=nkd.getNilai();

else if(String.valueOf(nKey[counter]).equals(" "))

kodeKey[counter]=nkd.space.getNilai();

}

kodePlain[counter]=kodeCipher[counter]-kodeKey[counter];

if(kodePlain[counter]<0)

kodePlain[counter]=kodePlain[counter]+53;

plain=plain+NilaiKarakterEnkripsi.get(kodePlain[counter]);

plain=plain.replaceAll("space", " ");

}

**Gambar 5.8 *Source code* proses *retrieve* dan dekripsi**

* 1. **Pengujian**

Pengujian merupakan tahap yang utama dalam pembuatan suatu aplikasi. Hasil pengujian yang didapat akan dijadikan tolak ukur dalam proses pengembangan selanjutnya. Pengujian ini dilakukan untuk mengetahui hasil yang didapat dari aplikasi yang dibuat.

Aplikasi *Hidden Message* ini menggunakan sepuluh citra untuk diuji, masing-masing lima *file* citra dengan format BMP dan lima *file* citra dengan format JPG. Selain itu, pengujian juga dilakukan dengan menggunakan tiga pesan dengan panjang yang berbeda. Rincian pengujian akan ditampilkan dalam tabel yang memperlihatkan bahan uji coba dan hasil pengujian.

**Tabel 5.1 Citra yang akan diuji**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Nama *file*** | **Format *file*** | **Ukuran *file* (KB)** |
| 1 | Baboon | BMP | 704 |
| 2 | Baseball | BMP | 2.305 |
| 3 | Goldhill | BMP | 1.216 |
| 4 | Lena | BMP | 769 |
| 5 | OnePiece | BMP | 507 |
| 6 | BABYMETAL | JPG | 112 |
| 7 | Hyunyoung | JPG | 17 |
| 8 | Lubudw8 | JPG | 107 |
| 9 | SCANDAL | JPG | 317 |
| 10 | Sunshangxiang | JPG | 751 |

**Tabel 5.2 Pesan yang akan diuji**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Pesan** | **Panjang pesan** | **Kunci** |
| 1 | Satryo Wicaksono | 16 | wise knight |
| 2 | Universitas Halu Oleo | 21 | tes |
| 3 | Teknik Informatika Universitas Halu Oleo | 40 | kendari |

Hasil pengujian citra dengan menyisipkan pesan “Satryo Wicaksono” ditunjukkan pada tabel 5.2.

**Tabel 5.3 Hasil pengujian citra dengan menyisipkan pesan “Satryo Wicaksono” (panjang 16 karakter)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Nama *file*** | **Ukuran *file* asli (KB)** | **Ukuran *file* setelah *embedding* (KB)** | **MSE rata-rata** | **PSNR rata-rata** |
| 1 | Baboon.bmp | 703 | 703 | 1,3981 | 114,098 |
| 2 | Baseball.bmp | 2.304 | 2.304 | 2,873067 | 110,6741 |
| 3 | Goldhill.bmp | 1.215 | 1.215 | 2,929267 | 130,3652 |
| 4 | Lena.bmp | 768 | 768 | 1,072033 | 102,7719 |
| 5 | OnePiece.bmp | 506 | 506 | 0,088667 | 82,29013 |
| 6 | BABYMETAL.jpg | 111 | 1.940 | 0,011733 | 102,491 |
| 7 | Hyunyoung.jpg | 16 | 219 | 0,018467 | 95,4656 |
| 8 | Lubudw8.jpg | 106 | 1.426 | 0,117067 | 76,87277 |
| 9 | SCANDAL.jpg | 316 | 5.934 | 19,8602 | 36,05037 |
| 10 | Sunshangxiang.jpg | 750 | 3.354 | 0,0177 | 97,1674 |

Hasil pengujian citra dengan menyisipkan pesan “Universitas Halu Oleo” ditunjukkan pada tabel 5.3.

**Tabel 5.4 Hasil pengujian citra dengan menyisipkan pesan “Universitas Halu Oleo” (panjang 21 karakter)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Nama *file*** | **Ukuran *file* asli (KB)** | **Ukuran *file* setelah *embedding* (KB)** | **MSE rata-rata** | **PSNR rata-rata** |
| 1 | Baboon.bmp | 703 | 703 | 3,1411 | 108,0471 |
| 2 | Baseball.bmp | 2.304 | 2.304 | 0,0071 | 104,296 |
| 3 | Goldhill.bmp | 1.215 | 1.215 | 3,5596 | 120,5678 |
| 4 | Lena.bmp | 768 | 768 | 0,0408 | 94,37803 |
| 5 | OnePiece.bmp | 506 | 506 | 0,0779 | 85,80783 |
| 6 | BABYMETAL.jpg | 111 | 1.940 | 0,0093 | 105,5915 |
| 7 | Hyunyoung.jpg | 16 | 219 | 0,0282 | 93,729 |
| 8 | Lubudw8.jpg | 106 | 1.426 | 1,5469 | 112,4692 |
| 9 | SCANDAL.jpg | 316 | 5.934 | 19,8592 | 36,0533 |
| 10 | Sunshangxiang.jpg | 750 | 3.354 | 0,0156 | 99,88443 |

Hasil pengujian citra dengan menyisipkan pesan “Teknik Informatika Universitas Halu Oleo” ditunjukkan pada tabel 5.4.

**Tabel 5.5 Hasil pengujian citra dengan menyisipkan pesan “Teknik Informatika Universitas Halu Oleo” (panjang 40 karakter)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Nama *file*** | **Ukuran *file* asli (KB)** | **Ukuran *file* setelah *embedding* (KB)** | **MSE rata-rata** | **PSNR rata-rata** |
| 1 | Baboon.bmp | 703 | 703 | 0,063667 | 94,23573 |
| 2 | Baseball.bmp | 2.304 | 2.304 | 0,009033 | 105,8294 |
| 3 | Goldhill.bmp | 1.215 | 1.215 | 0,0322 | 103,4593 |
| 4 | Lena.bmp | 768 | 768 | 0,052733 | 88,07947 |
| 5 | OnePiece.bmp | 506 | 506 | 0,061967 | 87,0446 |
| 6 | BABYMETAL.jpg | 111 | 1.940 | 0,006067 | 110,6153 |
| 7 | Hyunyoung.jpg | 16 | 219 | 0,125567 | 92,1337 |
| 8 | Lubudw8.jpg | 106 | 1.426 | 0,128767 | 75,9393 |
| 9 | SCANDAL.jpg | 316 | 5.934 | 19,85757 | 36,0567 |
| 10 | Sunshangxiang.jpg | 750 | 3.354 | 0,011733 | 104,7498 |

Dari hasil pengujian yang telah dilakukan, dapat dilihat bahwa ukuran *stego image* tidak mengalami perubahan ukuran (*size*) untuk citra dengan format BMP. Hal ini dikarenakan proses *embedding* pada algoritma *Pixel Value Differencing* hanya melakukan operasi pada nilai intensitas *pixel* citra dan tidak merubah dimensi atau jumlah *bit* citra sehingga ukuran citra tidak mengalami perubahan. Hal yang berbeda terjadi pada citra dengan format JPG. Perubahan signifikan terjadi pada ukuran (*size*) citra karena citra mengalami konversi dari format JPG (*lossy compression*) ke BMP (*lossless compression*).

Salah satu kelemahan algoritma *Pixel Value Differencing* yaitu ketidakmampuannya untuk mendeteksi panjang pesan yang telah disisipkan pada suatu citra. Apabila hal ini tidak diperhatikan, maka pada saat proses *retrieve* akan muncul karakter-karakter lain selain yang disisipkan karena sistem akan memproses semua *pixel*. Salah satu cara yang dapat dilakukan untuk mengatasi kelemahan ini adalah dengan menyisipkan informasi yang dibutuhkan pada proses *retrieve* pada sebuah *pixel* di dalam citra. Informasi tersebut yaitu panjang pesan dan indeks *pixel* terakhir yang diproses pada saat melakukan proses *embedding*. Karena disisipkan pada *pixel*, maka kedua nilai tersebut tidak dapat melebihi 255. Dengan mempertimbangkan hal tersebut, jumlah karakter maksimal yang dapat disisipkan dapat dihitung. Dengan asumsi bahwa satu karakter membutuhkan enam *pixel* untuk dapat disisipkan pada citra, maka jumlah karakter maksimal yang dapat disisipkan yaitu 255 : 6 = 42,5.

Pengujian berikut dilakukan untuk melihat apa yang terjadi apabila jumlah karakter yang disisipkan melebihi batas tersebut. Pengujian dilakukan dengan menyisipkan pesan “Teknik Informatika Fakultas Teknik Universitas Halu Oleo” (panjang 56 karakter) dengan kunci “tes”.

**Tabel 5.6 Pengujian batas karakter maksimal (panjang pesan 56 karakter)**

|  |  |  |
| --- | --- | --- |
| **No.** | **Nama *file*** | **Pesan terdekripsi** |
| 1 | Baboon.bmp | Teknik InIXJwXJIXJIXTvKJIXvIUJIXJrXJsXrIXJoXJIXJIXJIXJIXJ |
| 2 | Baseball.bmp | Teknik InIXJwXJIXJIXTwXJuXJIXJIGJIXJIVJIXJIXtIXpIGJcXgIXJ |
| 3 | Goldhill.bmp | Teknik InIXJwXJIXJIXTvKJIXvIUJIXJrXJsXrIXJoXsIXJIXJclJIXJ |
| 4 | Lena.bmp | Teknik JJIXJVXHIXJIXJIlJIXJIjJIXJIXHUHJIXJDXJhXJIwJIXJIXJ |
| 5 | OnePiece.bmp | TeknjJIBJIAJIXJIXJIXJIXJIXJAXJIXJIXJIXJIXJIXfSXfIXjIXOIIJ |
| 6 | BABYMETAL.jpg | TekniJIXCIXiIXJIXGIIJIXJIXJIXJIXv XJIXJIXJIXJIXJIXJIXYIXJ |
| 7 | Hyunyoung.jpg | Teknik InIXJwXJIXJIXJvKJIXvIUJIXJIVVtXJIXEIXJIXJHXJIXjIXJ |
| 8 | Lubudw8.jpg | Teknik InIXJIXJIXJIXuIXJIXJIXJIXJIGJIXEIXJuXJIXJIXJIXJhXJ |
| 9 | SCANDAL.jpg | Teknik InIXJwXJIXJIXTvKJIXvIUJIXJrXJsXrIXJoXtcJJsXPIXJIXJ |
| 10 | Sunshangxiang.jpg | Teknik InIXJwXJIXJIXTvKJIXvIUJIXJrXJsXrIXJoXtIlJIXJIXJIXJ |

Dari hasil pengujian dapat dilihat bahwa apabila pesan yang dimasukkan melebihi batas 40 karakter, maka akan terjadi *error* pada proses ekstraksi *bit* pada proses dekripsi. Hal ini disebabkan oleh perubahan nilai parameter indeks *pixel* terakhir pada saat melakukan proses *embedding*. Sebagai contoh, pada proses *embedding* *file* Baboon.bmp, indeks *pixel* terakhir yang diproses adalah 299. Karena nilai intensitas maksimum pada sebuah *pixel* adalah 255, hal tersebut menyebabkan kesalahan yang mengakibatkan nilai indeks *pixel* terakhir pada saat melakukan proses *embedding* berubah menjadi 43. Akibatnya, proses ekstraksi *bit* pesan tidak berjalan dengan semestinya dan menghasilkan pesan terdekripsi yang tidak sesuai dengan pesan asli.

Pengujian juga dilakukan terhadap *file* citra berukuran besar (2 MB dan 4 MB) untuk melihat apakah ukuran *file* berpengaruh terhadap proses penyisipan. Penyisipan dilakukan pada tiga *file* citra berukuran 2 MB, satu *file* citra berukuran 3 MB, dan dua *file* citra berukuran 4 MB. Pengujian dilakukan dengan menyisipkan pesan “Teknik Informatika Universitas Halu Oleo” dengan kunci “tes”.

**Tabel 5.7 Pengujian penyisipan pada *file* berukuran besar (panjang pesan 40 karakter)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Nama *file*** | **Ukuran *file* sebelum penyisipan (KB)** | **Ukuran *file* setelah penyisipan (KB)** | **Pesan terdekripsi** |
| 1 | Truck.jpg | 2.314 | 15.314 | Teknik Informatika Universitas Halu Oleo |
| 2 | Kingsborough.jpg | 2.045 | 6.053 | Teknik Informatika Universitas Halu Oleo |
| 3 | MtRobson.jpg | 2.044 | 22.419 | Teknik Informatika Universitas Halu Oleo |
| 4 | Armadillo.jpg | 4.546 | 19.116 | Teknik Informatika Universitas Halu Oleo |
| 5 | Blackwater.jpg | 3.727 | 21.119 | Teknik Informatika Universitas Halu Oleo |
| 6 | Duck.jpg | 4.097 | 53.626 | Teknik Informatika Universitas Halu Oleo |

Dari hasil pengujian dapat dilihat bahwa pesan terdekripsi yang dihasilkan dari proses *retrieve* dan dekripsi sama dengan pesan yang disisipkan pada proses penyisipan sehingga dapat disimpulkan bahwa ukuran *file* citra tidak mempengaruhi proses penyisipan.