BAB III

ANALISIS DAN DESAIN SISTEM

III.1. Analisis Masalah

Masalah dalam sisitem ini adalah bagaimana agar sistem ini dapat membantu pengguna sistem untuk melakukan pengamanan data (*data security*). Dalam sistem ini terdapat dua pengguna yaitu *ekstractor* dan *embeddor* dimana keduanya memiliki peran sesuai kebutuhan.

Yang menjadi masalah utama penelitian ini adalah bagaimana membandingkan metode *LSB* dan metode Adaptif dalam melakukan proses penyisipan dan ekstraksi pesan. Kedua metode ini akan diteliti dan ditentukan yang terbaik dengan parameter *fidelity* yang ada. Setelah dipilah dan dibagi dalam beberapa ketegori, dapat diuraikan sebagai berikut :

1. User 1 (*embeddor*) : adalah seorang yang melakukan proses penyisipan, ekstraksi serta perhitungan nilai persamaan dan bineri. Dengan kedua parameter *fidelity* maka user 1 dapat menentukan kelayakan *stego image* dapat disampaikan kepada User 2 (*extractor*).
2. User 2 (*extractor*) : adalah seseorang yang melakukan proses ekstraksi yaitu pengurain kembali pesan rahasia yang sudah disisipkan.
3. Proses : Sistem ini akan berjalan dengan membandingkan kedua metode yang digunakan yaitu metode *LSB* dan metode Adaptif. Pada umumnya kedua sistem ini memiliki kesamaan dalam teknik penyembunyian pesan dengan merubah pesan kedalam bentuk biner,

akan tetapi keduanya memiliki perbedaan dalam proses menentukan letak (koordinat) bit yang dapat disisipi pesan rahasia.

4. Sistem : Sistem ini akan menunjukkan metode mana yang terbaik dalam melakukan teknik steganografi, sesuai dengan parameter yang ada yaitu teknik mana yang lebih baik dan aman dalam melakukan proses penyembunyian pesan rahasia.

III.2. Analisis Proses Penyembunyian Pesan dengan Metode *LSB* dan Adaptif

III.2.1. Analisis Proses Penyembunyian dengan Metode *LSB*

1. Proses (*Embedded)* atau Penyisipan Pesan

Metode ini memodifikasi nilai yang tidak signifikan dari jumlah bit dalam *byte* file *carrier*. Sebagai contoh, akan dilakukan proses penyembunyian karakter ‘G’ (ASCII 71) pada berkas *carrrier*. *Least Significant Bit* dari file *carrier* di tandain dengan garis bawah. Berkas carrier dalam biner dengan ukuran 8 *byte* :

1001010**1** 0000110**1** 1100100**1** 1100100**1**

0000111**1** 1100101**1** 1001111**1** 0001000**0**

Karakter ‘G’ dalam biner dengan ukuran 1 *byte* : 01000111. Delapan bit tersebut nantinya akan dimasukkan kedalam *Least Significant Bit* dari tiap – tiap *byte* pada file *carrier* seperti berikut ini :

1001010*0* 0000110*1* 1100100*0* 1100100*0*

0000111*0* 1100101*1* 1001111*1* 0001000*1*

Pada contoh diatas, hanya sebagian dari *Least Significant Bit* file *carrier* yang berubah (ditunjukan dengan angka yang dicetak miring). Berdasarkan teori yang di dapat adalah bahwa kemungkinan terjadinya perubahan bit adalah sekitar 50 %, karena peluang perubahannya adalah antara 0 atau 1 dan dengan mengubah *Least Significant Bit* maka ukuran dari file pembawa tidak akan berubah sehingga akan sulit untuk terdeteksi. Untuk lebih jelas pemaparan proses penyisipan pesan kedalam gambar, berikut studi kasus penyisipan pesan kedalam gambar.

Pesan yang akan disisipkan kedalam file *image* adalah RAHASIA yang akan di*convert* ke bentuk desimal dan dari desimal kedalam bentuk biner yang di jelaskan pada tabel III.1 dibawah ini :

Tabel III.1. Tabel Convert

Karakter	Desimal	Biner
R	82	1010010
A	65	1000001
H	72	1001000
A	65	1000001
S	83	1010011
I	73	1001001
A	65	1000001

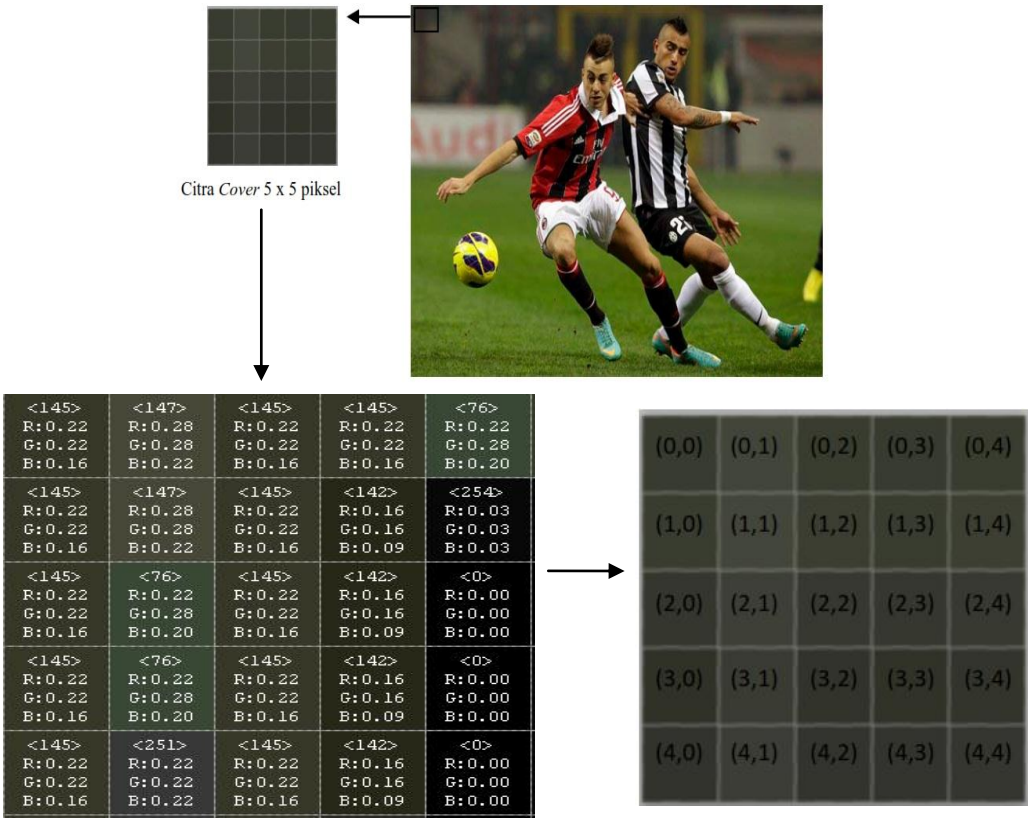
Dari tabel diatas yang akan disisipkan kedalam *file image* adalah nilai biner misalkan nilai biner R = 1010010, karena file gambar menggunakan 8 bit maka biner dari nilal R yang masih berjumlah 7 bit maka akan ditambahkan nilai 0 di barisan depan sehingga berjumlah 8 bit menjadi 01010010. Dan dilanjutkan ke nilai biner A = 1000001. Sama seperti cara di atas karena bentuk biner nilai A hanya 7 bit ditambahkan nilai 0 di barisan depan sehingga menjadi 8 bit sehingga

menjadi 01000001. Begitu juga selanjutnya di karakter H hingga A sampai pesan RAHASIA selesai dirubah kebentuk biner sehingga didapat pesan yang akan dimasukkan adalah 010100100100000101001000010000010101001101001001010000001 dan siap untuk dilakukan penyisipan ke dalam *file image*.

Setelah pesan siap untuk disisipi ke *file image* kita harus baca nilai *pixel* atau mengatahui perhitungan nilai biner dari *pixel* citra. Sebagai contoh disini menggunakan citra warna berdimensi 525 x 388 *pixel* seperti pada gambar III.1.

Gambar III.1. Citra 1 (525 x 236 *pixel*)

Pada citra gambar III.1 diatas diambil contoh *pixel* dari masing – masing komponen warna RGB-nya untuk kita sisipkan pesan, seperti yang kita ketahui untuk file bitmap 24 bit maka *pixel* (titik) pada gambar tersebut terdiri dari 3 warna (Red, Green, Blue) yang masing – masing disusun oleh bilangan 8 bit. Penentuan *pixel* mana yang diambil ditentukan oleh bilngan *pseudo random* atau secara semu acak. Sebagai contoh diberikan cuplikan 5 x 5 *pixel* yang berasal dari citra yang dapat di lihat seperti gambar III.2.



Gambar III.2. Citra Penomoran Metode *LSB* (5 x 5)

Setelah didapat *pixel* yang akan dilakukan penyisipan pesan, selanjutnya dilakukan perhitungan nilai Red Green Blue (RGB) sebagai berikut :

a. Nilai *pixel* (0,0) = 00010000001011000010110.

Dimana nilai tersebut di dapat dari RGB yang nilai R = 22 dirubah kedalam bentuk biner menjadi 10110 dan karena *pixel* bernilai 8 bit maka hasil nilai R yang diubah kebentuk biner masih bernilai 5 bit maka ditambahkan 0 dibagian depan agar melengkapi menjadi 8 bit seperti berikut 00010110. Nilai biner yang digaris bawahi adalah nilai yang ditambakan 0 untuk melengkapi menjadi 8 bit. Kemudian

dilanjutkan ke nilai G = 22 dirubah kedalam bentuk biner 10110 dan sama seperti langkah diatas dilengkapi menjadi 8 bit sehingga hasilnya 00010110. Selanjutnya ke nilai B = 16 dirubah kedalam bentuk biner 10000 dilengkapi menjadi 8 bit dengan nilai 00010000.

b. Nilai *pixel* (0,1) = 0001110*0*0001110*0*00010110

Dimana nilai tersebut di dapat dari RGB yang nilai R = 28 dirubah kedalam bentuk biner menjadi 11100 dan karena *pixel* bernilai 8 bit maka hasil nilai R yang diubah kebentuk biner masih masih bernilai 5 bit maka ditambahkan 0 dibarisan depan agar melengkapi menjadi 8 bit menjadi seperti berikut 00011100. Nilai biner yang digaris bawahi adalah nilai yang ditambakan 0 untuk melengkapi menjadi 8 bit. Kemudian dilanjutkan ke nilai G = 28 dirubah kedalam bentuk biner 11100 dan sama seperti langkah diatas dilengkapi menjadi 8 bit sehingga hasilnya 00011100. Selanjutnya ke nilai B = 22 dirubah kedalam bentuk biner 10110 dilengkapi menjadi 8 bit dengan nilai 00010110. Dan begitu selanjutnya hingga *pixel* tercukupi untuk melakukan penyembunyian pesan.

Setiap 1 *pixel* (0,0) RGB dapat disisipi 3 bit pesan yang akan dimasukkan kedalam file *image*, dan lokasi penyisipan dapat di lihat pada nilai *pixel* (0,0) yang di cetak miring 0001000*0*0001011*0*00010110*.* Pesan yang akan disisipi yaitu 0100100100000101001000010000010101001101001001010000 01, tiga bit paling depan akan di masukkan terlebih dahulu kedalam

pixel (0,0) menjadi 0001000**0**0001011**1**0001011**0** sehingga tersisa bit pesan 10010010000010100100001000001010100110100100101000001 untuk kemudian diteruskan disisipi ke *pixel* (0,1). Selanjutnya *pixel* (0,1) yang memiliki nilai bit 0001110*0*0001110*0*0001011*0* dimasukan pesan yang sama seperti *pixel* (0,0) yang dapat menampung 3 bit pesan 100100100000101001000010000010101001101001001010000001 sehingga didapat hasil 0001110***1***0001110***0***0001011***0*** sehingga tersisa pesan 001000001010010000100000101010011010010010100000 untuk di teruskan ke *pixel* (0,2) dan seterusnya hingga nilai pesan yang berbentuk biner selesai disisipi kedalam *pixel*.

2. Proses Ekstraksi dengan Teknik *LSB*

Pada proses ini, akan dilakukan pengungkapan kembali pesan rahasia yang tersisip pada berkas Citra. Caranya adalah dengan mengambil 1 bit dari tiap 8 bit *pixel* berkas Citra. Sebagai contoh akan dilakukan proses ekstraksi dari berkas Citra dalam bentuk biner sebagai berikut :

Berkas *carrier* dalam biner dengan ukuran 8 *byte*.

1001010*0* 0000110*1* 1100100*0* 1100100*0*

0000111*0* 1100101*1* 1001111*1* 0001000*1*

Least Significant Bit (*LSB*) adalah bit yang di tandai dengan garis bawah. Selanjutnya bit *LSB* akan disalin secara urut dari depan yang

akan menghasilkan 1 *byte* yaitu : 01000111 yang merupakan biner dari karakter ‘G’.

Dan berikut studi kasus proses pengekstraksian pesan yang pertama dilakukan merupakan proses pengekstraksian pesan berdasarkan proses penyisipan akan di ekstraksi atau diungkap dari nilai biner dimulai dari *pixel* (0,0) sampai ke *pixel* terakhir yang disisipkan. Nilai *pixel* (0,0) yang bernilai 00010000000101110001011<u>0</u> dimana setiap *pixel* RGB (0,0) dapat disisipi 3 bit pesan R disisipi 1 bit, G disisipi 1 bit dan B disisipi 1 bit yang tiap – tiap bit nya diletakan pada biner terakhir sehingga di dapat pesan 010. Dan dilanjutkan ke *pixel* RGB (0,1) yang bernilai 00011101000111000001011<u>0</u> dimana setiap *pixel* RGB (0,1) disisipi 3 bit pesan R disisipi 1 bit, G disisipi 1 bit dan B disisipi 1 bit yang tiap – tiap bit nya diletakan pada biner terakhir sehingga di dapat pesan 010100. Begitulah proses pengekstraksian pesan hingga ke *pixel* RGB yang terakhir hingga di dapatkan pesan yang terakhir.

III.2.2. Analisis Proses Penyembunyian Pesan pada Citra *GIF* dengan Menggunakan Metode Adaptif

1. Proses Penyembunyian dengan Metode Adaptif

Untuk melakukan penyisipan pesan pada citra *GIF*, dibutuhkan pesan yang akan disisipkan (*message*) .txt. Setelah itu dilakukan proses pemilihan *pixel* secara semu acak, sesuai dengan bilangan semu acak yang telah dibangkitkan. Setelah itu dilakukan pencarian nilai paritas

warna palet dari *pixel- pixel* yang telah dipilih. Setelah nilai paritas dari palet warna sebuah citra *bitmap* ditentukan, proses penyisipan pesan berlanjut pada tahap pengecekan paritas. *Pixel* yang telah dipilih secara semu acak menjadi tempat penyisipan pesan akan dicek nilai paritasnya dengan bit pesan. Jika perbandingan menghasilkan persamaan bit, maka *pixel* tersebut tidak dimodifikasi dan pengecekan berlanjut kepada *pixel* selanjutnya. Jika terjadi perbedaan bit, maka *pixel* tersebut akan di modifikasi warnanya dengan cara mencari warna tetangga terdekat menggunakan rumus persamaan jarak $d = \sqrt{(R_2 - R_1)^2 + (G_2 - G_1)^2 + (B_2 - B_1)^2}$ yang memiliki nilai paritas yang berbeda.

Warna tetangga terdekat mempunyai nilai *d* yang terkecil dibanding warna awal dan jika pada awalnya *pixel* tersebut mempunyai paritas warna palet 1, maka dicari warna tetangga terdekat yang mempunyai nilai paritas 0. Lalu warna tetangga terdekat tersebut akan menggantikan warna awal pada *pixel* tersebut. Pada proses penyisipan pesan perlu diketahui pula ukuran pesan yang disisipkan. Citra *GIF* memiliki fitur *comment extension* yang merupakan suatu blok pada *data stream* pada berkas citra *GIF* yang bertujuan untuk memperbolehkan sistem pengkode citra *GIF* untuk menambahkan *metadata* ataupun informasi tambahan mengenai berkas citra *GIF* tersebut. Informasi yang ditambahkan ke dalam *comment extension* adalah ukuran pesan dan nama berkas yang berisi pesan yang disisipkan. Hal ini diperlukan untuk memberi informasi akhir pesan kepada sistem pengekstraksian

pesan. Informasi akhir pesan akan menjadi batas akhir pembacaan pesan saat proses ekstraksi. Pembacaan pesan akan dilakukan maksimal sebanyak ukuran pesan.

Nama berkas pesan yang disisipkan disimpan pada *comment extension* untuk keperluan pengekstrasian pesan. Setelah proses penyisipan pesan selesai, dilakukan proses kompresi citra untuk mengembalikannya ke dalam bentuk citra *GIF*. Dari hasil analisis didapatkan bahwa setiap *pixel* pada citra *GIF* dapat disisipkan oleh satu buah bit pesan, sehingga maksimum jumlah ukuran pesan yang dapat disisipkan pada citra *GIF* menggunakan metode adaptif adalah sesuai dengan persamaan berikut :

$$P = \frac{m\ x\ n}{8}$$

Keterangan :

p : ukuran pesan.

m : ukuran panjang citra *GIF* dalam satuan *pixel*.

n : ukuran lebar citra *GIF* dalam satuan *pixel*.

Untuk lebih jelas pemaparan proses penyisipan pesan kedalam gambar, berikut studi kasus penyisipan pesan kedalam gambar. Sama dengan pesan yang disisipkan kedalam metode *LSB*, pesan yang disisipkan kedalam metode Adaptif juga harus dirubah ke bentuk biner, hanya pada metode *LSB* 1 *pixel* (titik) dapat disisipi 3 bit pesan, kalau dalam metode Adaptif 1 *pixel* (titik) hanya dapat disisipi 1 bit pesan saja. Pesan yang akan disisipkan kedalam *file image* adalah RAHASIA

yang akan di*convert* ke bentuk desimal dan dari desimal kedalam bentuk biner yang di jelaskan pada tabel III.2 dibawah ini :

Tabel III.2. Tabel Convert 2

Karakter	Desimal	Biner
R	82	1010010
A	65	1000001
H	72	1001000
A	65	1000001
S	83	1010011
I	73	1001001
A	65	1000001

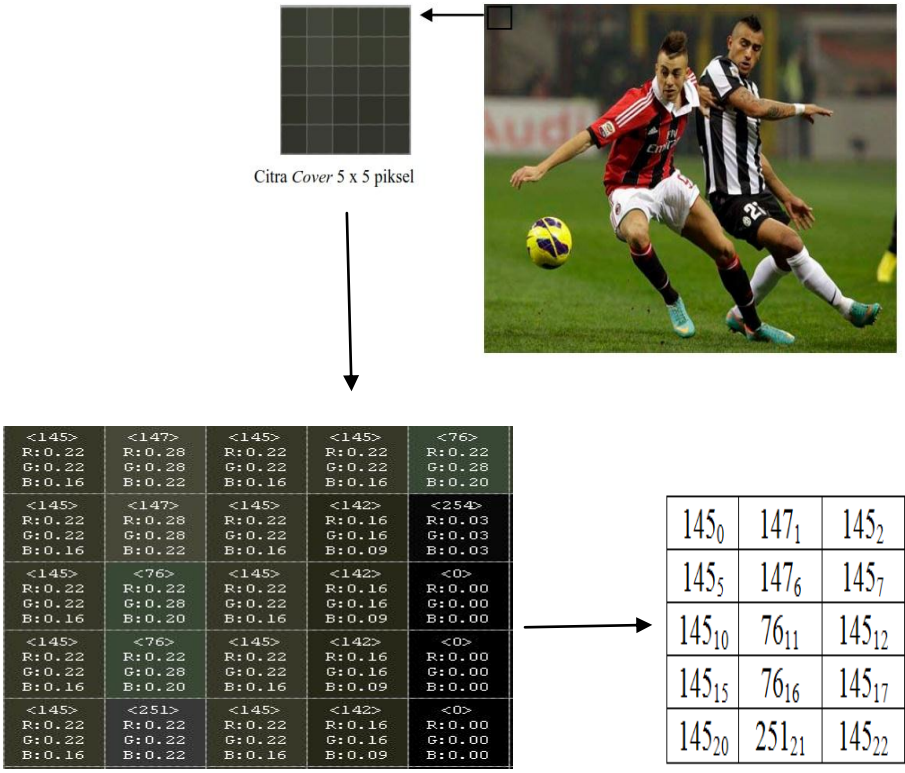
Dari tabel diatas yang akan disisipkan kedalam *file image* adalah nilai biner misalkan nilai biner R = 1010010, karena file gambar menggunakan 8 bit maka biner dari nilal R yang masih berjumlah 7 bit maka akan ditambahkan nilai 0 di barisan depan sehingga berjumlah 8 bit menjadi 01010010. Dan dilanjutkan ke nilai biner A = 1000001. Sama seperti cara di atas karena bentuk biner nilai A hanya 7 bit ditambahkan nilai 0 di barisan depan sehingga menjadi 8 bit sehingga menjadi 01000001. Begitu juga selanjutnya di karakter H hingga A sampai pesan RAHASIA selesai dirubah kebentuk biner sehingga didapat pesan yang akan dimasukkan adalah 0101001001000001010010000100000101010011010010010100000 1 dan siap untuk dilakukan penyisipan ke dalam *file image*.

Setelah pesan siap untuk disisipi ke *file image* pada metode Adaptif pemilihan *pixel* dipilih secara himpunan bilangan semu acak, tetapi sebelumnya *pixel – pixel* akan diberi nomor urut dimulai dari kiri ke kanan lalu kebaris selanjutnya dari kiri kekanan dan begitu seterusnya.

Sama dengan studi kasus pada *LSB* citra yang kita gunakan warna berdimensi 525 x 388 *pixel* seperti pada gambar III.3.

Gambar III.3. Citra 2 (525 x 236 *pixel*)

Dan berikut tabel *pixel* dengan penomoran dan diurutkan dari kiri ke kanan dimulai dari nomor urut 0 yang di tunjukan gambar III.4, dan setelah *pixel* diberikan nomor urut, bilangan semu acak yang telah dibangkitkan menjadi acuan *pixel* mana yang akan disisipkan pesan.



Gambar III.4. Gambar Penomoran *Pixel* Metode Adaptif

Kemudian pada metode Adaptif pesan disisipkan berdasarkan nilai suatu *pixel* yang telah dipilih secara semu acak misalkan proses pembangkitan himpunan bilangan semu acak di dapat nomor *pixel* 2,6,9,11,15,17,18,21,22,23 dan nilai paritas warna citra akan ditentukan menggunakan persamaan (R + G + B) mod 2, sehingga hasil persamaan tersebut adalah 0 atau 1. Berikut contoh perhitungan persamaannya sebagai berikut.

145 → R : 22, G : 22, B : 16 → (22 + 22 + 16) mod 2 = 0

147 → R : 28, G : 28, B : 22 → (28 + 28 + 22) mod 2 = 0

254 → R : 03, G : 03, B : 03 → (03 + 03 + 03) mod 2 = 1

76 → R : 22, G : 28, B : 20 → (22 + 28 + 20) mod 2 = 0

145 → R : 22, G : 22, B : 16 → (22 + 22 + 16) mod 2 = 0

145 → R : 22, G : 22, B : 16 → (22 + 22 + 16) mod 2 = 0

142 → R : 16, G : 16, B : 09 → (16 + 16 + 09) mod 2 = 1

251 → R : 22, G : 22, B : 22 → (22 + 22 + 22) mod 2 = 0

145 → R : 22, G : 22, B : 16 → (22 + 22 + 16) mod 2 = 0

142 → R : 16, G : 16, B : 09 → (16 + 16 + 09) mod 2 = 1

Hasil dari penentuan paritas warna yang telah dirubah kebentuk persamaan yang menjadi tempat penyisipan akan dicek nilai paritasnya dengan bit pesan. Jika perbandingan menghasilkan persamaan bit, maka *pixel* tersebut dilakukan penyisipan dan berlanjut kepenyisipan pesan ke dalam *pixel* berikutnya, namun jika di dapati perbedaan bit maka *pixel* tersebut menggunakan persamaan $d = \sqrt{(R_2 - R_1)^2 + (G_2 - G_1)^2 + (B_2 -}$

$\overline{B_1})^2$ untuk mencari paritas warna terdekat. Warna tetangga terdekat nilai *d* dibanding warna awal dan jika pada awalnya *pixel* tersebut mempunyai paritas warna *pixel* 1 dan pesan yang ingin disisipi bernilai 0 maka dicari warna tetangga terdekat yang mempunyai nilai paritas 0, kemudian warna tetangga tersebut akan menggantikan warna awal pada *pixel* tersebut. Selanjutnya kita ambil *sample* potongan *pixel* dan pesan untuk memaparkan proses penyisipan pesan pada tabel III.3.

Tabel III.3. Proses Pengecekan dan Penyisipan Pesan

Indeks Warna	Paritas	Pesan	Hasil Penyembunyian	Keterangan
145	0	0	0	-
147	0	1	0	Dilakukan persamaan rumus
254	1	0	0	Dilakukan persamaan rumus
76	0	1	0	Dilakukan persamaan rumus
145	0	0	0	-
145	0	0	0	-
142	1	1	1	-
251	0	0	0	-
145	1	0	0	Dilakukan persamaan rumus
142	1	1	1	-

Hasil penyisipan dengan metode Adaptif tentunya lebih baik dari pada *LSB* karena hasil perubahan gambar tidak akan terlihat, karena metode adaptif menyembunyikan pesan ke dalam paritas warna *pixel* yang sama.

2. Proses Ekstraksi dengan Teknik Adaptif

Untuk ekstraksi pesan dari citra *GIF* menggunakan metode Adaptif membutuhkan masukan berupa *stego object*. Proses ekstraksi pesan diawali dengan pembangkit bilangan semu acak lalu dicari nilainya. Setelah itu dihitung nilai PSNR.

Berikut studi kasus pengekstraksian pesan yang di awali pembacaan *comment extension* pada berkas citra *GIF* dan ukuran pesan. Informasi ukuran pesan dibutuhkan untuk mengetahui batas akhir pembacaan pesan pada *pixel* nantinya. Setelah itu dilanjutkan kebilangan semu acak ini memberikan informasi *pixel* mana yang disisipkan pesan dan menjadi acuan *pixel* mana yang akan dibaca. Setelah bilangan semu acak diketahui himpunan bilngan semu acak diketahui, maka *pixel – pixel* yang mempunyai nomor urut sesuai bilangan semu acak yang dibangkitkan akan ditelusuri dan tiap *pixel* tersebut dicari nilai paritas warna. Sekumpulan paritas warna *pixel – pixel* tersebut adalah bit pesan yang akan dikembalikan oleh proses ekstraksi sebagai pesan yang disisipkan. Dan berikut berkas citra yang digunakan untuk penelusuran pesan.

Tabel III.4. Berkas Citra yang Dibaca

145_0	147_1	145_2	145_3	76_4
145_5	147_6	145_7	142_8	254_9
145_{10}	76_{11}	145_{12}	142_{13}	0_{14}
145_{15}	76_{16}	145_{17}	142_{18}	0_{19}
145_{20}	251_{21}	145_{22}	142_{23}	0_{24}

Setelah pembangkit bilangan semu acak mendapatkan berkas nomor *pixel* 2,6,9,11,15,17,18,21,22,23 yang disisipi pesan telah teridentifikasi, dilakukan penelusuran tiap *pixel* dan dicari nilai paritas warna dari tiap *pixel* yang dibaca. Berikut proses pencarian nilai paritas dijelaskan pada tabel III.5.

Tabel III.5. Proses Pencarian Nilai Paritas *Pixel*

Nomor *Pixel*	**Indeks Warna**	**Paritas**
2	145	0
6	147	0
9	254	1
11	76	0
15	145	0
17	145	0
18	142	1
21	251	0
22	145	0
23	142	1

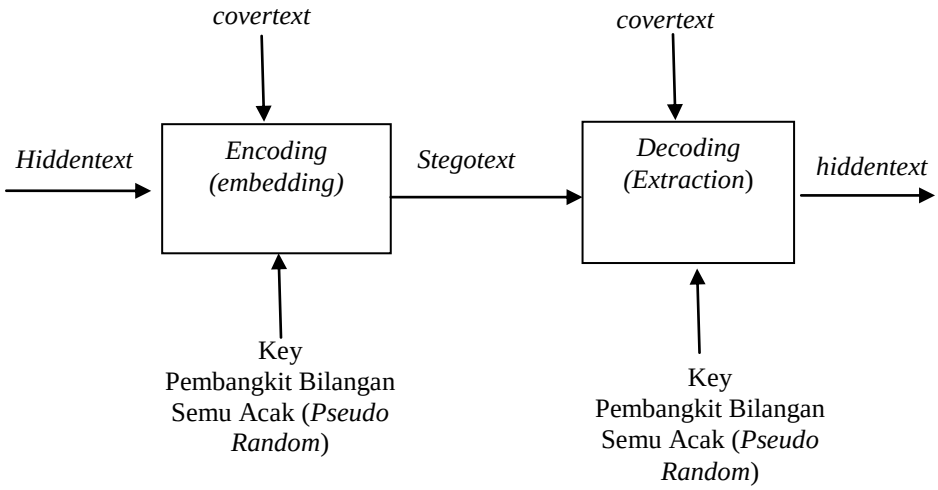
Proses selanjutnya setelah nilai paritas dari *pixel* ditemukan adalah pembacaan pesan. Nilai paritas yang ditemukan diurutkan sesuai urutan pembacaan *pixel*, dan kumpulan nilai paritas yang ditemukan diurutkan pembacaan *pixel*, dan kumpulan nilai paritas tersebut adalah bit – bit dari pesan yang disisipkan. Dari tabel III.5 di dapat bahwa pesan yang disisipkan adalah 0010001001.

III.3. Desain Sistem

III.3.1. Blok Diagram

1. Blok Diagram Penyembunyian Pesan Menggunakan Metode *LSB*

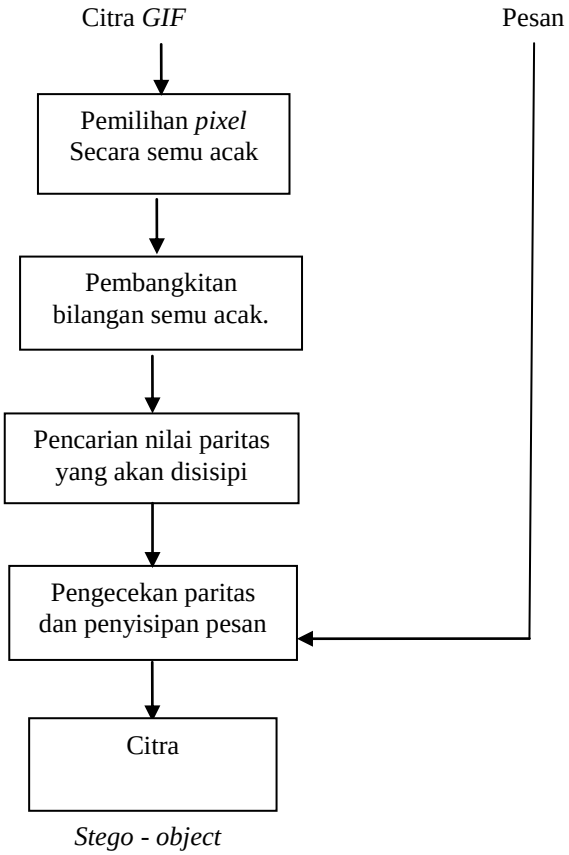
Berikut blok diagram penyembunyian dan ekstraksi pesan teks ke dalam citra *GIF* atau citra penampung :



Gambar III.5. Blok Diagram Proses Penyembunyian dan Ekstraksi Pesan

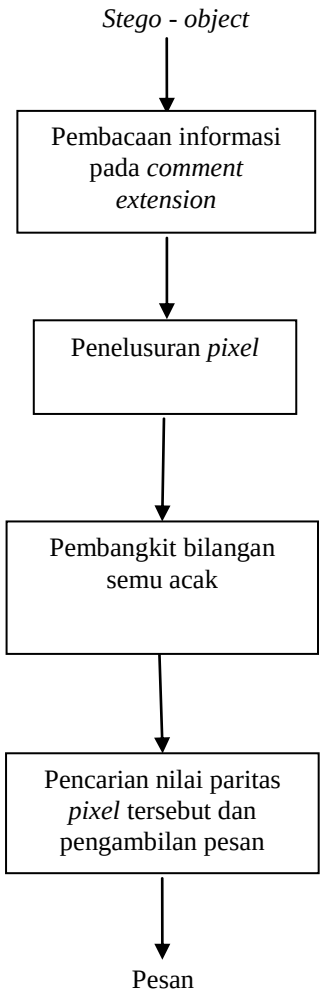
2. Blok Diagram Penyembunyian Pesan Menggunakan Metode Adaptif

Berikut blok diagram penyisipan dan ekstraksi pesan teks ke dalam citra *GIF* atau citra penampung :



Gambar III.6. Blok Diagram Proses Penyembunyian Pesan

Dan blok diagram proses ekstraksi pesan menggunakan metode Adaptif adalah sebagai berikut :

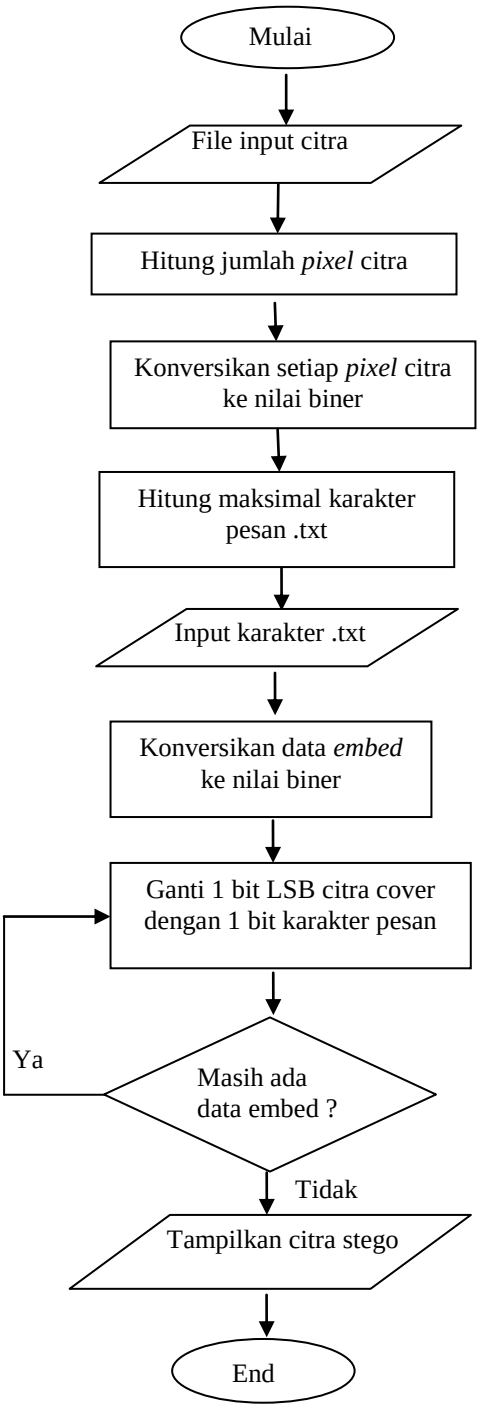


Gambar III.7. Blok Diagram Proses Ekstraksi Pesan

III.3.2. *Flow Chart*

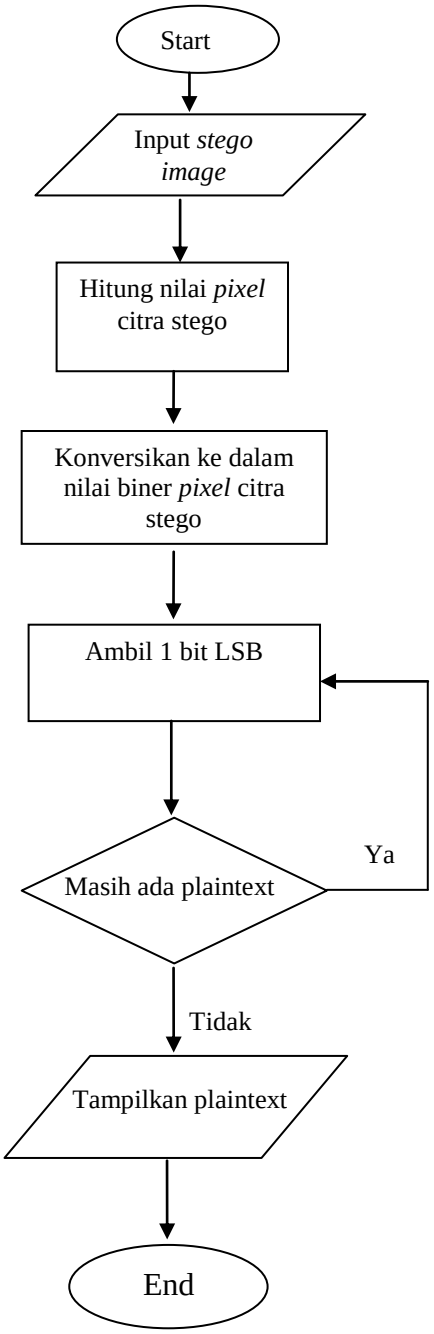
1. *Flow Chart* Penyembunyian Pesan Menggunakan Metode *LSB*

Berikut *flowchart* penyembunyian pesan teks ke dalam citra penampung :



Gambar III.8. *Flowchart* Penyembunyian Pesan (*Embedding Message*)

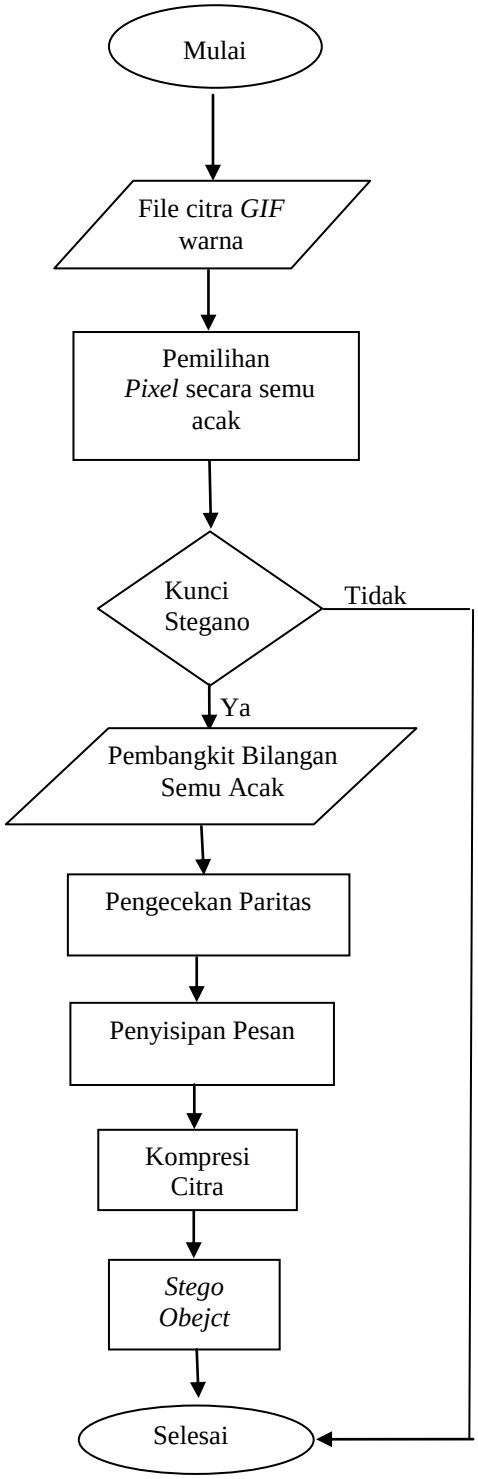
Dan *flowchart* proses ekstraksi pesan menggunakan metode *LSB* adalah sebagai berikut :



Gambar III.9. *Flowchart* Ekstraksi Pesan (*decoder*)

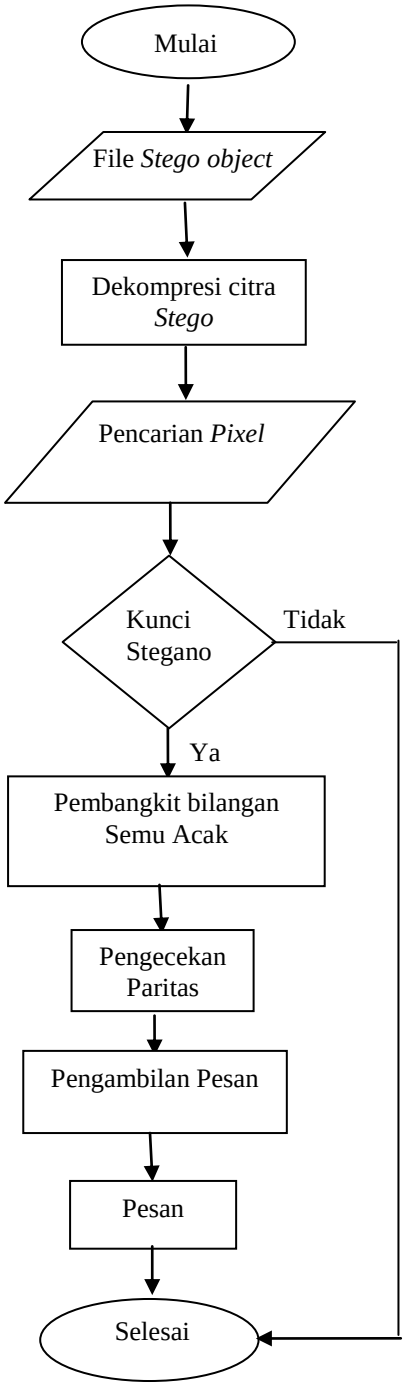
2. *Flow Chart* Penyembunyian Pesan Menggunakan Metode Adaptif

Berikut *flowchart* penyisipan pesan teks ke dalam citra penampung :



Gambar III.10. *Flowchart* Proses Penyembunyian Pesan

Dan *flowchart* proses ekstraksi pesan menggunakan metode Adaptif adalah sebagai berikut :



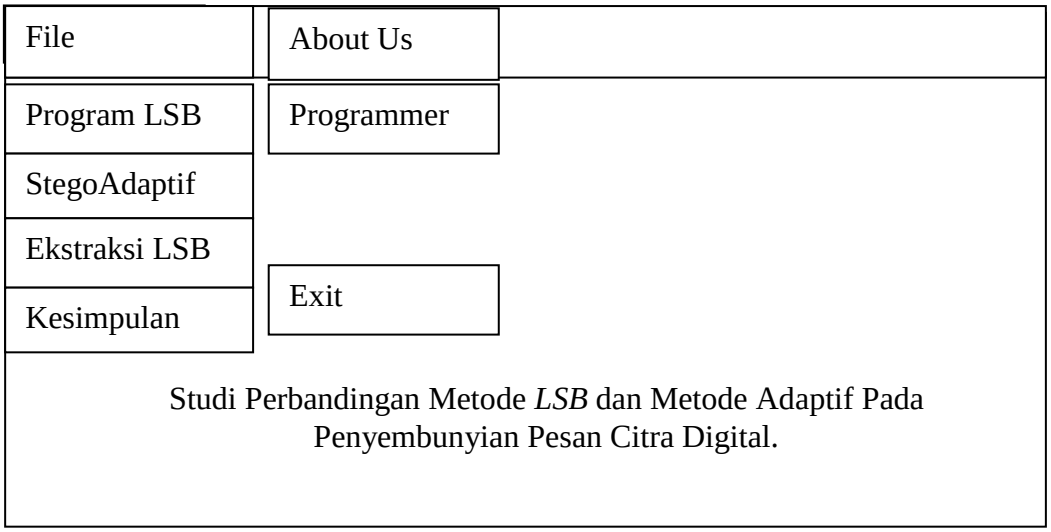
Gambar III.11. Proses Ekstraksi Pesan dengan Metode Adaptif

III.4. Desain User Interface

Berikut adalah desain sistem studi perbandingan dari skripsi ini :

1. Desain Sistem Menu Utama

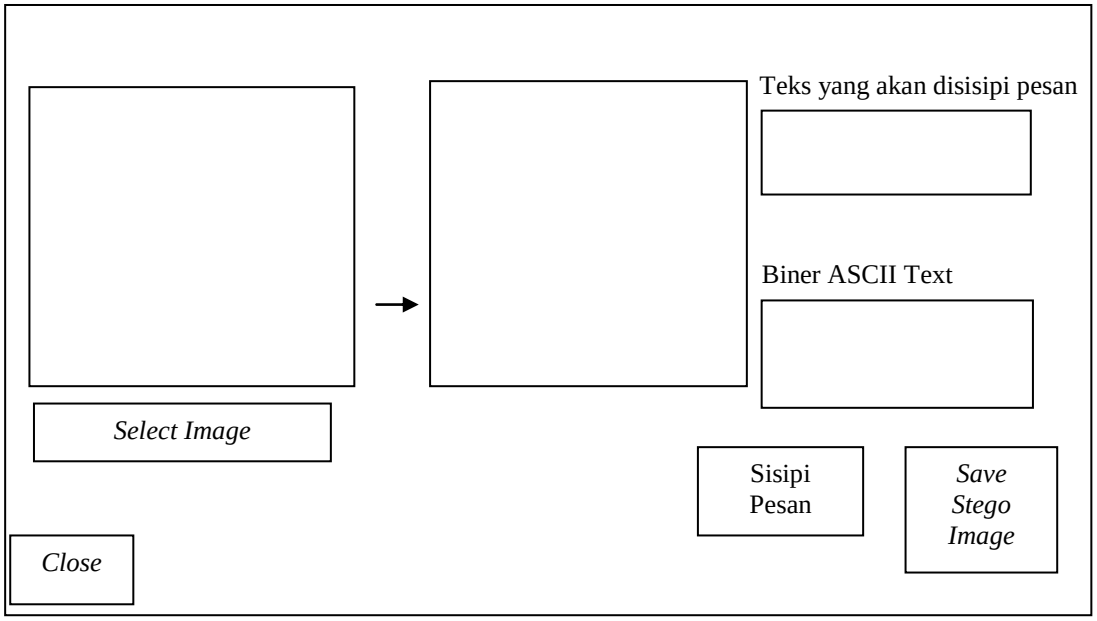
   Adalah menu awal yang ditampilkan saat aplikasi dijalankan. Pada menu utama ini ditampilkan 7 buah pilihan menu, yaitu program LSB, program Adaptif, Ekstraksi *LSB*, Ekstraksi Adaptif, Kesimpulan, tentang pembuat dan satu lagi menu untuk keluar. Jika tombol program *LSB* dipilih maka aplikasi akan menampilkan menu program *LSB* dan jika menu program Adaptif yang dipilih maka aplikasi akan menampilkan menu program Adaptif, begitu juga dengan pilihan yang lainnya. Berikut merupakan desain tampilan dari program menu utama yang akan di rancang.



Gambar III.12. Desain Sistem Menu Utama Studi Perbandingan

2. Desain Sistem Menu Program *LSB*

Menu program *LSB* ditampilkan saat tombol program *LSB* pada menu di pilih atau di *double clik*. Pada menu program *LSB* ditampilkan 4 buah *field* yaitu open *image*, sisipi teks, *save* dan *close*. *Select image* digunakan untuk memanggil file citra *GIF* yang akan di sisipi pesan dan teks yang akan sisipi diisi oleh pengguna dengan pesan yang ingin di sembunyikan ke file gambar dan *field save* digunakan untuk menyimpan gambar yang telah disisipi pesan dan satu *field close* untuk keluar.

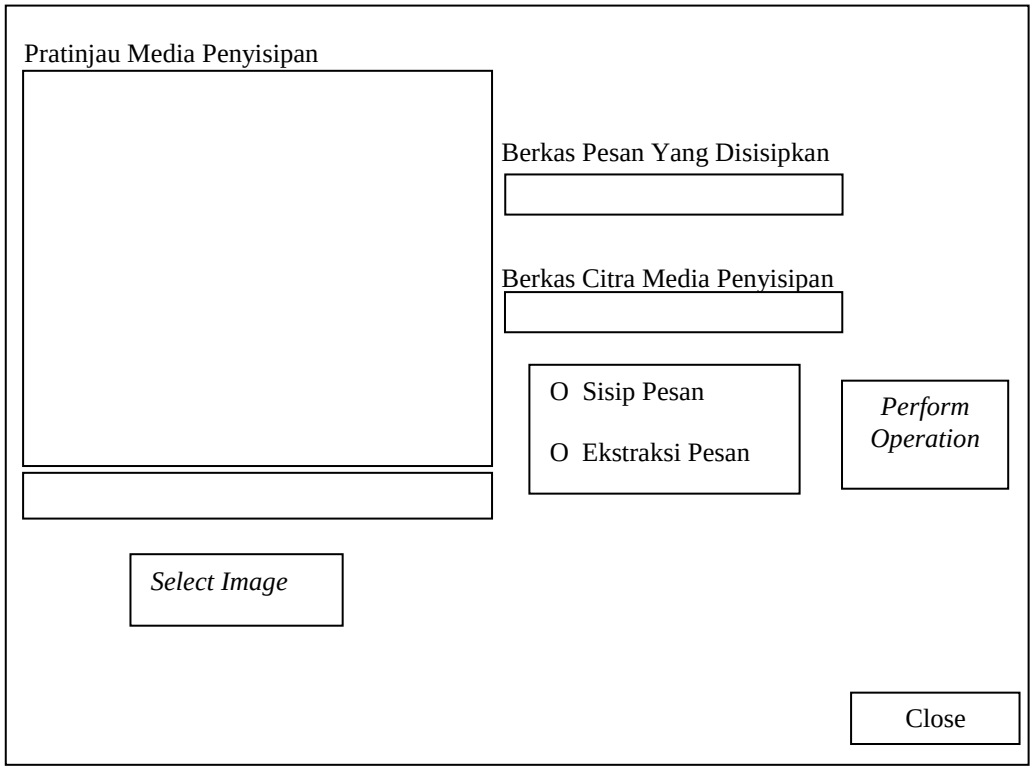


Gambar III.12. Desain Sistem Program *LSB*

3. Desain Sistem Menu Program Adaptif

Menu program Adaptif ditampilkan saat tombol program Adaptif pada menu di pilih atau di *double clik*. Pada menu program Adaptif ditampilkan 8 buah *field* yaitu berkas citra yang disisipkan, pesan yang

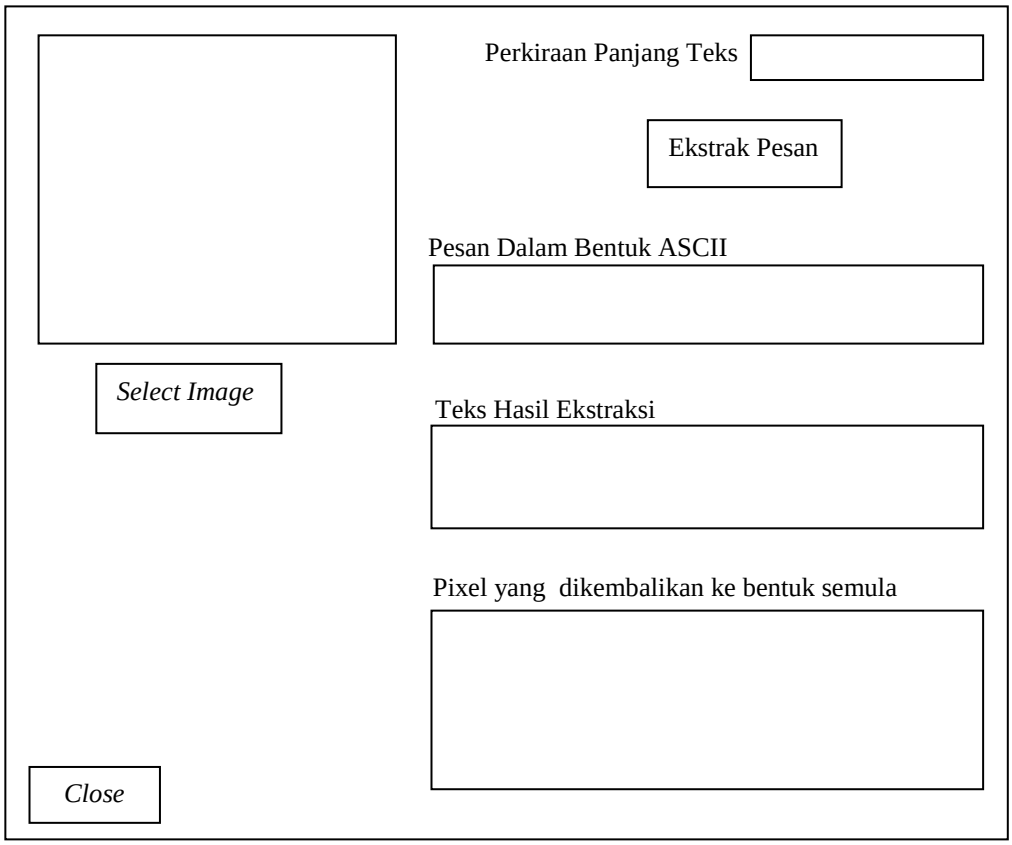
ingin disisip yg berekstensi .txt, button *select image* dan *field* sisipi dan ekstraksi serta berkas hasil pengekstraksian. *Field* pertama berisikan berkas citra yang akan disisipi dan diekstraksi pesan dan *field* kedua merupakan pesan akan akan kita sisipi kedalam gambar yang berektensikan .txt, ketiga *field* sisip berfungsi untuk melakukan proses penyisipan pesan kedalam file *image* apabila *field* sisip di pilih dan menekan button perform untuk menjalankan proses penyisipan. Yang ke empat *field* ekstraksi yang apa bila kita pilih akan melakukan proses ekstraksi ketika button *perform* di klik. *Field* Kelima yaitu button *perform* yang berfungsi untuk menjalankan proses yang terpilih pada *field* sisip dan ekstraksi. *Field* keenam yaitu *field* textbox sebagai tempat keluaran hasil proses ekstraksi pesan. *Field* Ketujuh yaitu *field textbox* yang berisikan alamat citra dimana dilakukan penyimpanan dan yang terakhir button *close* untuk keluar dari halaman menu metode Adaptif. Untuk mendaptkan alamat direktori, pengguna dapat menjalankan proses penyisipan pesan dan ekstraksi dengan memilih tombol “Sisip Pesan“ atau pun “Ekstraksi” yang akan dipilih dan mengklik button perform akan muncul sebuah *dialog* untuk menentukan alamat direktori dan nama berkas citra *GIF* yang akan disimpan dan yang ingin di ekstraksi.



Gambar III.14. Desain Sistem Program StegoAdaptif

4. Desain Sistem Menu Ekstraksi *LSB*

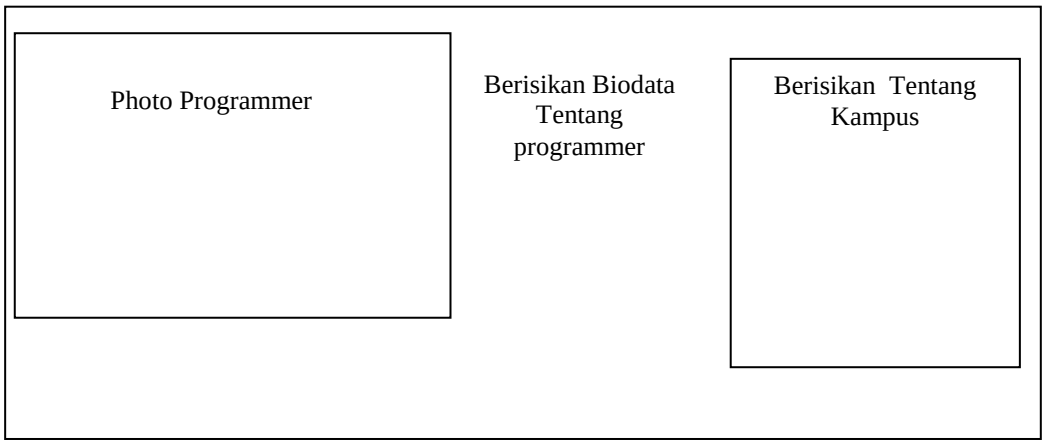
Menu program Ekstraksi *LSB* ditampilkan saat tombol program *LSB* pada menu di pilih atau di *double clik*. Pada menu program *LSB* ditampilkan 3 buah *field* yaitu perkiraan panjang teks yang di sembunyikan, *field* ekstraksi pesan yang di *double clik* agar proses pengembalian atau ekstraksi pesan berjalan dan yang ketiga *field close* untuk keluar dari program *LSB*.



Gambar III.15. Desain Tampilan Program Ekstraksi *LSB*

5. Desain Sistem Menu Tampilan *Programmer*

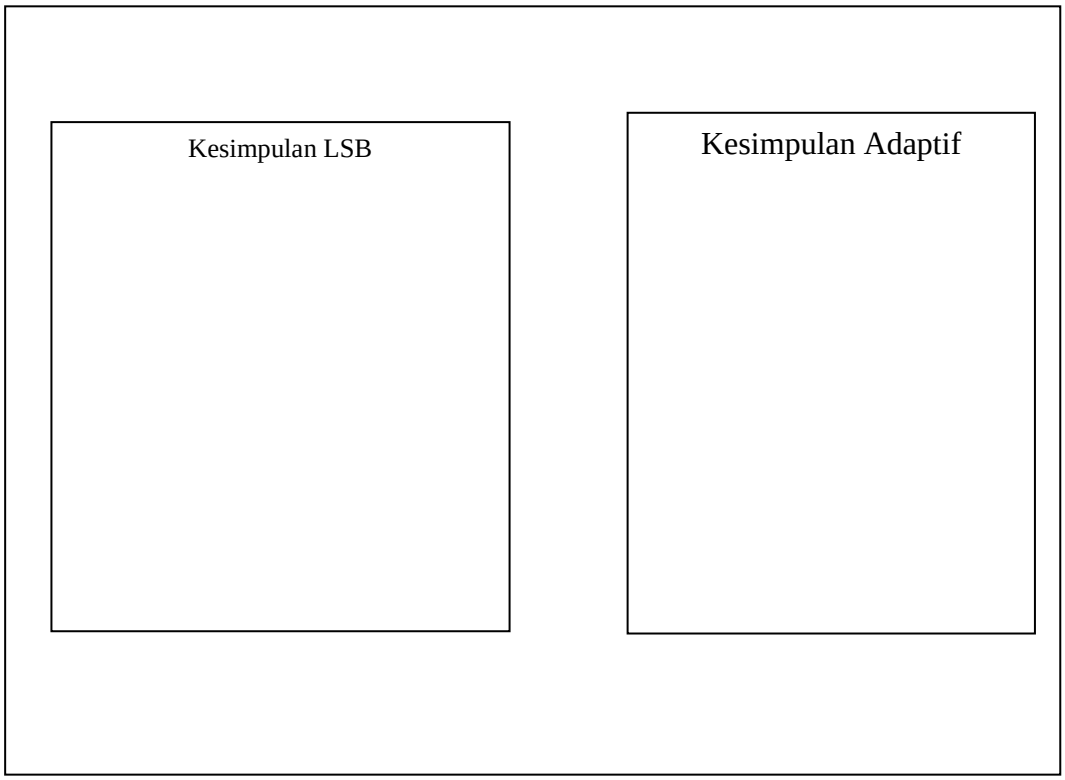
Berisikan photo, biodata peneliti dan tentang Kampus.



Gambar III.16. Desain Tampilan Data Programmer

6. Desain Sistem Menu Tampilan Kesimpulan

   Pada menu kesimpulan berisikan teoritis utama kedua metode dalam melakukan penyembunyian pesan.



Gambar III.17. Desain Tampilan Data Kesimpulan