BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

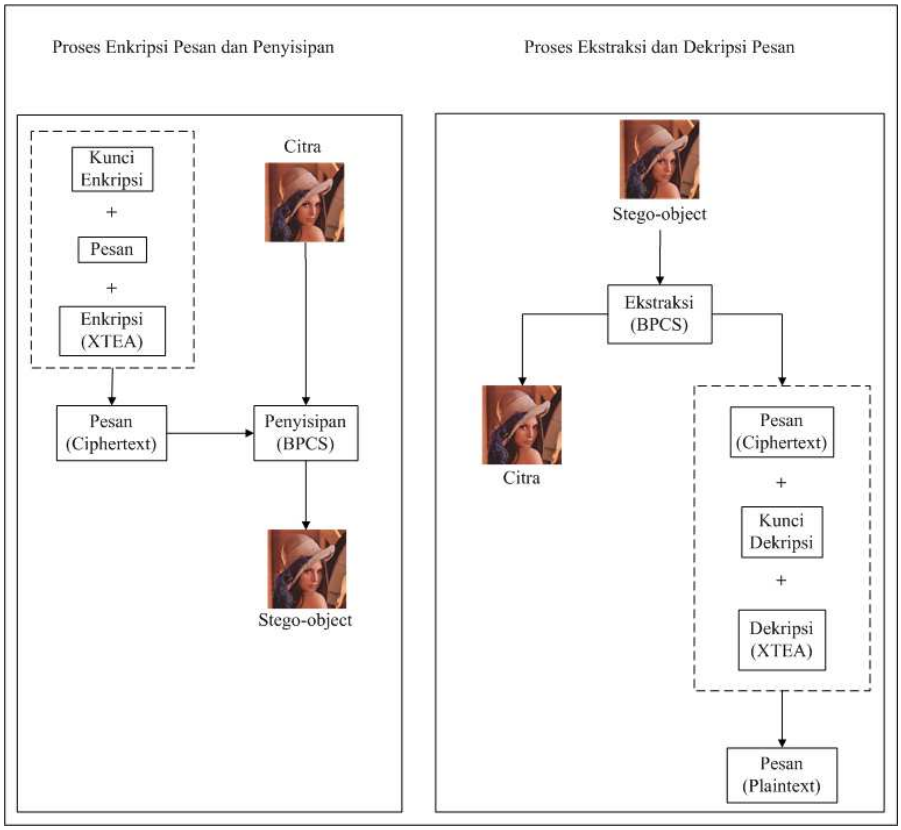
Sebelum memasuki tahap perancangan program, maka akan dilakukan tahap analisis sistem terlebih dahulu, yang bertujuan untuk mempelajari prosedur yang sedang berjalan saat ini dan kebutuhan dari pengguna aplikasi. Dalam analisis sistem dilakukan penguraian dari suatu sistem informasi yang utuh menjadi bagian-bagian yang lebih terperinci dengan maksud untuk mengidentifikasi dan mengevaluasi permasalahan – permasalahan, hambatan – hambatan yang terjadi dan kebutuhan – kebutuhan yang diharapkan sehingga dapat diusulkan perbaikannya. Dalam pembuatan program aplikasi ini diperlukan suatu analisis yang benar agar dihasilkan suatu program aplikasi yang sesuai dengan yang diharapkan.

3.1.1 Analisis Masalah

Analisis masalah dilakukan untuk mengetahui masalah-masalah apa saja yang sering terjadi dalam pembangunan aplikasi steganografi. Masalah yang terjadi ketika pembangunan steganografi pada citra digital yaitu keamanan pesan, kapasitas penyisipan informasi ke dalam citra relatif kecil, kualitas citra digital yang disisipkan informasi, dan ketahanan pesan yang tersisip dalam citra stego yang telah dimanipulasi seperti citra stego yang dirotasi dan dilakukan proses *croping*/pemotongan citra stego.

3.1.1.1 Deskripsi Sistem

Gambar 3.1 adalah deksripsi sistem yang digunakan pada steganografi yang akan dibangun



Gambar 3.1 Deskripsi Sistem Steganografi

Untuk mengatasi permasalahan yang sudah dijelaskan pada analisis masalah, berikut dijelaskan pada gambar 3.1 yang memiliki 2 proses, yaitu :

1. **Proses enkripsi pesan dan penyisipan**
   a. Pertama masukkan kunci enkripsi
   b. Masukkan pesan yang akan disisipkan ke dalam citra
   c. Pesan yang akan disisipkan ke dalam citra di enkripsi terlebih dahulu, proses enkripsi membutuhkan kunci enkripsi untuk menyandikan pesan sehingga jika pesan bisa diambil dari citra maka pesan tidak bisa langsung dibaca. Enkripsi dilakukan dengan menggunakan algoritma XTEA
   d. Pesan yang sudah di enkripsi selanjutnya akan disisipkan ke dalam citra digital menggunakan metode BPCS
   e. Pesan yang sudah disisipkan ke dalam citra digital. Kualitas citra sebelum dan sesudah disisipi pesan tidak jauh berbeda.

2. Proses ekstraksi dan dekripsi pesan

a. Citra-stego yaitu citra yang sudah disisipi pesan akan di ekstraksi menggunakan metode BPCS sehingga pesan yang ada di dalamnya dipisahkan dengan citra asli
b. Pesan yang sudah terpisah dari citra asli merupakan pesan yang belum bisa dibaca karena masih terenkripsi
c. Masukkan kunci dekripsi yang sama dengan kunci enkripsi
d. Pesan di dekripsi menggunakan metode XTEA
e. Pesan sudah bisa dibaca.

3.2 Analisis Algoritma

Analisis algoritma digunakan untuk mengetahui alur proses dari algoritma yang digunakan untuk dapat diterapkan ke dalam aplikasi yang dibangun. Pembangunan aplikasi ini menggunakan metode steganografi yang digunakan yaitu metode *Bit Plane Compexity Segmentation* (BPCS) untuk proses penyisipan dan ekstraksi informasi. Sedangkan keamanan terhadap informasi yang akan disisipkan menggunakan algoritma XTEA.

Bit plane complexity segmentation (BPCS) adalah salah satu teknik steganografi yang diperkenalkan oleh Eiji Kawaguchi dan R.O. Eason pada tahun 1997. Pada BPCS dokumen citra dibagi menjadi segmen-segmen dengan ukuran 8x8 piksel setiap segmennya. Pada dokumen citra 8-bit, setiap satu segmen akan memiliki 8 buah *bit-plane* yang merepresentasikan piksel-piksel dari setiap bit tersebut. Proses penyisipan data dilakukan pada segmen yang memiliki kompleksitas yang tinggi. Segmen yang mmemiliki kompleksitas tinggi disebut *noise-like region*. Pada segmen-segmen ini penyisipan dilakukan hanya pada *least significant bit*, tapi pada seluruh *bit-plane.*

Sistem penyandian XTEA merupakan pengembangan dari sistem penyandian TEA. Algoritma XTEA merupakan algoritma simetris dimana kunci enkripsi sama dengan kunci dekripsi. Sistem penyandian XTEA menambahkan fungsi matematik berupa penambahan dan pengurangan

sebagai operator pembalik selain XOR dan juga pergeseran dua arah (ke kiri dan ke kanan) menyebabkan semua bit kunci dan data bercampur secara berulang ulang.

3.2.1 Proses Enkripsi dan Penyisipan Pesan

Proses enkripsi menggunakan algoritma XTEA, enkripsi dilakukan untuk mengubah pesan asli menjadi pesan yang disandikan sehingga menambah keamanan data. Sedangkan untuk penyisipan data yang sudah terenkripsi menggunakan metode *Bit plane complexity segmentation* BPCS. Tahap-tahap yang untuk melakukan proses enkripsi dan penyisipan pesan yaitu analisis pembangkit kunci, analisis enkripsi pesan dan analisis penyisipan pesan.

3.2.2 Analisis Algoritma XTEA

Algoritma XTEA merupakan algoritma kriptografi dengan metode enkripsi simetris yaitu algoritma yang menggunakan kunci untuk proses enkripsi sama dengan kunci untuk proses dekripsi. Panjang kunci algoritma XTEA yaitu 128 bit untuk enkripsi pesan yang akan disisipkan kedalam citra. Algoritma XTEA terdapat tiga proses yaitu proses pembangkit kunci, enkripsi pesan dan dekripsi pesan.

3.2.2.1 Analisis Pembangkit Kunci Algoritma XTEA

Proses pembangkit kunci merupakan proses dimana *cipherkey* dibangkitkan untuk menghasilkan nilai penjadwalan kunci yang digunakan untuk proses enkripsi dan dekripsi pada algoritma XTEA. Contoh pembangkit kunci pada algoritma XTEA, jika diketahui kunci yang digunakan untuk enkripsi dengan panjang 16 byte yaitu :

***cipherkey* : 1234567890123456**

Tahap awal ubah *cipherkey* kedalam bentuk *decimal* menjadi sebagai berikut :

1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
49	50	51	52	53	54	55	56	57	48	49	50	51	52	53	54

Tahap selanjutnya masukkan nilai *decimal cipherkey* kedalam fungsi pengacakan kunci. Dengan melakukan pergeseran dan operasi OR untuk menghasilkan *subkey*.

Masukkan *cipherkey* kedalam blok 16 byte, dimana 1 proses pengacakan membutuhkan 4 byte *cipherkey*.

```
S[0] = ((49 AND 255) Shl 24) OR ((50 AND 255) Shl 16)
       OR ((51 AND 255) Shl 8) OR ((52 AND 255))
S[0] =  822083584 OR 3276800 OR 13056 OR 52
```

S[0] = 825373492

S[0] ini merupakan pembangkit kunci yang digunakan untuk proses enkripsi dan dekripsi pada algoritma XTEA, selanjutnya dilakukan perulangan hingga didapat nilai S[0] hingga S[3].

```
S[1] = ((53 AND 255) Shl 24) OR ((54 AND 255) Shl 16)
       OR ((55 AND 255) Shl 8) OR ((56 AND 255))
S[1] = 889192448 OR 3538944 OR 14080 OR 56
```

S[1] = 892745528

```
S[2] = ((57 AND 255) Shl 24) OR ((48 AND 255) Shl 16)
       OR ((49 AND 255) Shl 8) OR ((50 AND 255))
S[2] = 956301312 OR 3145728 OR 12544 OR 50
```

S[2] = 959459634

```
S[3] = ((51 AND 255) Shl 24) OR ((52 AND 255) Shl 16)
       OR ((53 AND 255) Shl 8) OR ((54 AND 255))
S[3] = 855638016 OR 3407872 OR 13568 OR 54
```

S[3] = 859059510

Sehingga didapat nilai penjadwalan kunci, sebagai berikut :

S[0] = 825373492

S[1] = 892745528

S[2] = 959459634

S[3] = 859059510

Subkey ini yang digunakan untuk proses enkripsi atau dekripsi pada algoritma XTEA.

3.2.2.2 Analisis Enkripsi Algoritma XTEA

Proses enkripsi pada algoritma XTEA dilakukan dengan cara mengambil setiap plaintext per blok 8 byte dan memecahnya menjadi round ganjil dan genap. Ulangi langkah tersebut sebanyak 32 round.

Contoh enkripsi pada algoritma XTEA, jika diketahui kunci dengan panjang 16 byte dan plaintext yang akan digunakan untuk enkripsi dengan panjang 16 byte

***cipherkey* : 1234567890123456**

***Plaintext* :** "**unikom bandung** "

Tahap awal ubah *plaintext* kedalam bentuk *decimal* menjadi sebagai berikut:

u	n	i	k	o	m	spasi	b	a	n	d	u	n	g	spasi	spasi
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
117	110	105	107	111	109	32	98	97	110	100	117	110	103	32	32

Pecah *plaintext* per blok 8 byte. Untuk setiap blok dipecah menjadi round ganjil dan round genap yang masing-masing memiliki 4 byte karakter.

```
v0 = ((117 AND 255) Shl 24) OR ((110 AND 255) Shl 16)
     OR ((105 AND 255) Shl 8) OR ((107 AND 255))
v0 = (1962934272 OR 7208960 OR 26880 OR 107 )
v0 = 1970170219

v1 = ((111 AND 255) Shl 24) OR ((109 AND 255) Shl 16)
     OR ((32 AND 255) Shl 8)  OR ((98 AND 255))
v1 = (1862270976 OR 7143424 OR 8192 OR 98 )
v1 = 1869422690
```

Hitung sebanyak 32 round. Inisialisasikan DELTA dengan nilai 0x9E3779B9 (-1640531527 dalam integer) dan putaran (n) sebanyak 32 round. Hitung nilai v0 dan v1 dengan nilai awal sum = 0. Perhitungan dilakukan dengan menggunakan nilai integer yang memiliki batas -2147483648 s/d 2147483648.

Nilai S[0..3] diambil dari hasil pembangkit kunci yang telah didapatkan dari proses pembangkitan kunci.

```
v0 += (((v1 Shl 4) XOR (v1 Shr 5) + v1) XOR
```

```
       (sum + S[sum AND 3])
v0 += (((1869422690 Shl 4) XOR (1869422690 Shr 5) +
       1869422690) XOR (0 + S[0 AND 3])
v0 += (((-154008032 XOR 58419459) + 1869422690)
       XOR S[0]
v0 += ((-173445341) + 1869422690) XOR 825373492
v0 +=  1869422690 XOR 825373492
v0 += 1411693745
v0  = 1970170219 + 1411693745 = -913103332
sum += DELTA
sum = 0 + (-1640531527) = -1640531527
v1 += (((v0 Shl 4) XOR (v0 Shr 5) + v0) XOR
       (sum + S[sum >> 11 AND 3])
v1 += (((-913103332 Shl 4) XOR (-913103332 Shr 5) +
       -913103332) XOR (-1640531527 + S[(-1640531527)
       Shr 11 AND 3])
v1 += (((-1724751424 XOR 105683248) + -913103332)
       XOR (-1640531527 + S[3])
v1 += ((2136085186) + (-913103332)) XOR (-1640531527
       + 859059510)
v1 += 1762795276 XOR -781472017
v1 +=  -1199984157
v1  = 1869422690 + (-1199984157) = 669438533
```

ulangi perhitungan tersebut sampai 32 round sehingga didapatkan nilai akhir v0 dan v1 yaitu :

v0 = -1525469918 v1 = 373980262

Ubah nilai akhir v0 dan v1 menjadi karakter ASCII dengan melakukan pergeseran. Nilai desimal yang diambil berupa nilai byte (8 bit).

v0 Shr 24

-1525469918 Shr 24 = **-91**

v0 Shr 16

-1525469918 Shr 16 = **19**

v0 Shr 8

-1525469918 Shr 8 = **45**

v0 = **34**

v1 Shr 24

373980262 Shr 24 = **22**

v1 Shr 16

373980262 Shr 16 = **74**

v1 Shr 8

373980262 Shr 8 = **124**

v1 = **102**

Desimal (byte)	-91	19	45	34
Biner	10100101	00010011	00101101	00100010
Karakter	¥		-	"

Desimal (byte)	22	74	124	102
Biner	00010110	01001010	01111100	01100110
Karakter		J	|	f

Sehingga di dapat 8 karakter *chipertext*. Lakukan perulangan sehingga didapatkan *chipertext*.

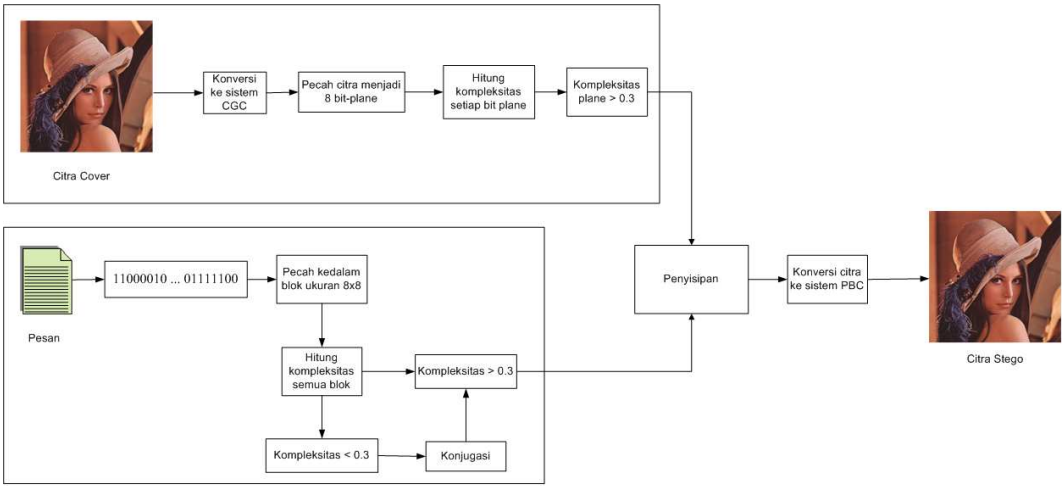
Chipertext dari *Plaintext* “unikom bandung " adalah **¥ -" J|fýÃ±?« A¨**

3.2.3 Analisis Metode *Bit Plane Complexity Segmentation* (BPCS)

Metode *Bit plane complexity segmentation* (BPCS) adalah salah satu teknik steganografi yang diperkenalkan oleh Eiji Kawaguchi dan R.O. Eason pada tahun 1997. Pada BPCS dokumen citra dibagi menjadi segmen-segmen dengan ukuran 8x8 piksel setiap segmennya. Pada dokumen citra 8-bit, setiap satu segmen akan memiliki 8 buah *bit-plane* yang merepresentasikan piksel-piksel dari setiap bit tersebut. Proses penyisipan data dilakukan pada segmen yang memiliki kompleksitas yang tinggi. Seperti halnya metode steganografi lainnya, metode ini terdapat dua proses yaitu proses penyisipan/*embedding* dan pengungkapan/ekstraksi.

3.2.3.1 Analisis Proses Penyisipan/*Embedding*

Proses penyisipan yaitu proses menyembunyikan informasi/pesan kedalam sebuah media penampung, dalam hal ini media penampung berupa citra digital. Proses penyisipan ini akan menghasilkan citra yang telah disisipi pesan (*stego-object*) yang menyerupai dengan citra sebelum disisipi pesan. Proses penyisipan pada metode *bit plane complexity segmentation* terlihat pada gambar 3.2



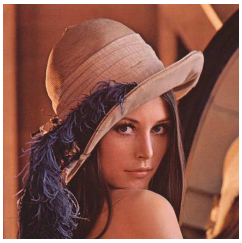
Gambar 3.2 Proses Penyisipan Metode BPCS

Contoh proses penyisipan, jika diketahui pesan yang akan disisipkan berupa file text yang berisi pesan yaitu “unikom bandung ” yang sudah di enkripsi menjadi “ **¥ -" J|fýÃ±?« A¨** “. Citra yang digunakan adalah lena.bmp

Proses penyisipan pesan dengan metode BPCS seperti berikut :

Pesan yang akan disisipkan : ¥ -" J|fýÃ±?« A¨

Tahap selanjutnya yaitu memilih citra yang akan dijadikan sebagai media penampung yang akan disisipkan pesan. Citra yang dipakai adalah lena.bmp



Gambar 3.3 Citra Penampung (lena.bmp)

Tahap selanjutnya adalah mengubah citra dari sistem PBC menjadi sistem CGC. Jika diketahui citra yang digunakan yaitu lena.bmp, contoh nilai piksel diambil dari potongan 8x8 citra tersebut dengan nilai piksel seperti pada gambar 3.4



-4100278	-3771313	-3968692	-3969205	-3640240	-3574960	-3706546	-3509167
-3902899	-3837106	-4034485	-3837619	-3508654	-3838132	-4035511	-3509167
-3706034	-4166836	-3639987	-3837617	-4100789	-3771573	-3969457	-3903413
-3640240	-4100792	-3640238	-3771826	-4034998	-3771824	-3969206	-3969205
-3640238	-3968954	-3640490	-3771827	-4034999	-3837869	-3903161	-3969203
-3705774	-3968695	-3771565	-3771314	-4034486	-3968944	-3968695	-4100532
-3771314	-3903151	-3902902	-3771567	-4034739	-4100281	-3903151	-4166072
-4033716	-4100014	-4165306	-3968176	-4165555	-4296892	-3968428	-4296888

Gambar 3.4 Nilai PBC potongan 8x8 dari citra lena.bmp

Maka tahap selanjutnya yaitu melakukan proses ekstraksi menggunakan metode *Bit Plane Complexity Segmentation* dengan tahap-tahapan yaitu sebagai berikut :

1. Ubah citra dari sistem PBC menjadi sistem CGC, sistem CGC dimaksudkan untuk menghindari perubahan piksel yang tidak terlalu mencolok, sehingga ketika pesan rahasia yang disisipkan kedalam citra tidak akan terlihat perbedaannya dengan citra asli oleh mata manusia.

   Perubahan sistem PBC menjadi CGC yaitu dengan menggunakan persamaan $g_i = b_{i-1}$ XOR b_i, contoh piksel[0][0] pada sistem PBC yaitu -4100278. Maka perhitungan nilai cgc adalah sebagai berikut :

```
cgc[0][0] = (pbc[0][0] Shr 1) XOR pbc[0][0]
          = (-4100278 Shr 1) XOR -4100278
          = -2050139 XOR -4100278
cgc[0][0] = 2218223
```



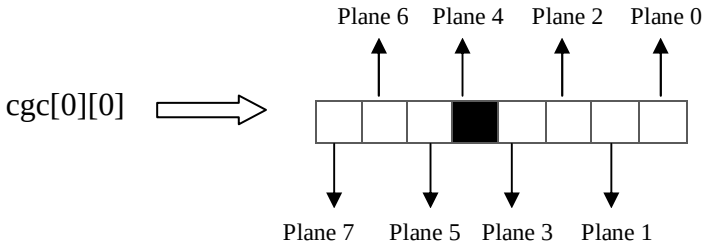
2218223	2444904	2279914	2283758	2903672	3001080	2410985	3100281
2509675	2607849	2312302	2607595	3100027	2611434	2316909	3100281
2411241	2121450	2904043	2607592	2218990	2444014	2283880	2508910
2903672	2218988	2903675	2444137	2316655	2444152	2283759	2283758
2903675	2279525	2902781	2444139	2316653	2607226	2509284	2283755
2412155	2279917	2444026	2444905	2312303	2279544	2279917	2218346
2444905	2509305	2509679	2444025	2316523	2218212	2509305	2120044
2312938	2214011	2116069	2280184	2115691	6379750	2280318	6379756

Gambar 3.5 Nilai CGC potongan 8x8 dari citra lena.bmp

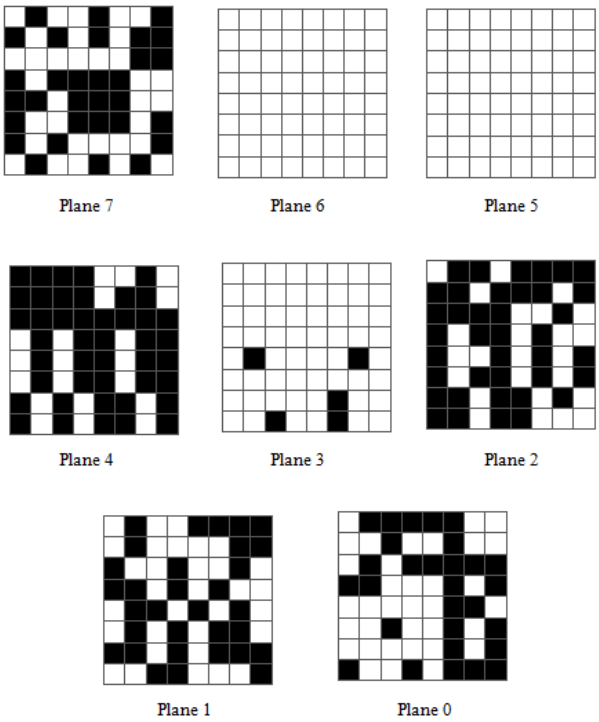
2. Melakukan proses *slicing* pada citra dengan sistem CGC. Piksel adalah nomor digit yang terdiri dari bit. Dalam gambar 8 bit terdiri dari 8 daerah, dari *bit plane* ‘0’(LSB) ke *bit plane* ‘7’(MSB). Plane ‘0’ berisi semua bit urutan terendah (LSB) dari semua piksel dalam gambar sementara *plane* ‘7’ berisis semua bit orde tinggi (MSB).

   Setelah didapatkan nilai dari gambar lena.bmp dengan sistem CGC maka tahap selanjutnya adalah mengubah nilai setiap piksel tersebut ke dalam nilai biner. Warna hitam untuk nilai biner 0 dan warna putih untuk nilai biner 1.

   Contoh pada gambar 3.5 piksel cgc[0][0] = 2218223 memiliki nilai biner 11101111. Maka blok hitam putihnya adalah sebagai berikut :

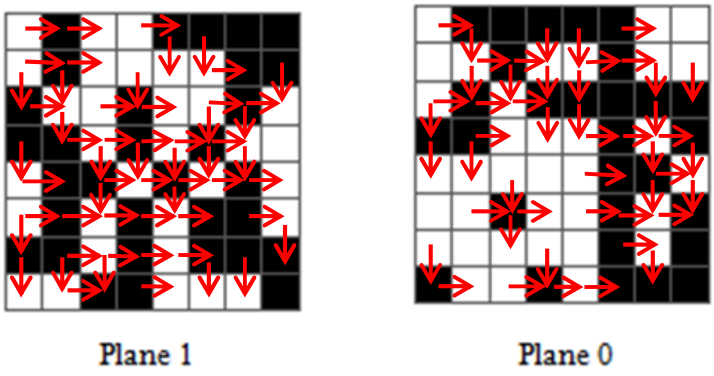


Ubah semua nilai piksel ke nilai biner. Hasil slicing dari potongan 8x8 dari citra lena.bmp dilihat pada gambar 3.6 :



Gambar 3.6 Bit Plane Slicing

3. Menghitung kompleksitas (α) setiap plane pada *bit plane*. Untuk menghitung kompleksitas setiap plane maka setiap piksel yang ada pada blok plane dilakukan perubahan warna hitam-putih secara vertikal dan horizontal. Contoh pada gambar 3.7 plane 0 dan plane 1 akan dihitung nilai kompleksitasnya.



Gambar 3.7 Citra Penampung Plane 0 dan Plane 1

Tahap selanjutnya adalah menghitung jumlah kompleksitas dengan menggunakan rumus

$$\alpha = \frac{k}{2\ x\ 2^n x\ (2^n - 1)}$$

Dengan *k* = jumlah perubahan warna hitam-putih pada setiap bit-plane. Untuk sebuah citra biner persegi dengan ukuran 2^n x 2^n, maka *n* adalah 2 x 2^n x (2^n-1).

Jumlah pergantian warna hitam-putih Plane 0 adalah

$$\alpha = \frac{k}{2 \, x \, 2^n \, x \, (2^n - 1)}$$

$$= \frac{51}{2 \, x \, 2^3 \, x \, (2^3 - 1)}$$

$$= \frac{51}{2 \, x \, 8 \, x \, (8 - 1)}$$

$$= \frac{51}{112}$$

$$\alpha = 0.46$$

Jumlah pergantian warna hitam-putih pada Plane 1 adalah

$$\alpha = \frac{k}{2 \, x \, 2^n \, x \, (2^n - 1)}$$

$$= \frac{61}{2 \, x \, 2^3 \, x \, (2^3 - 1)}$$

$$= \frac{61}{2 \, x \, 8 \, x \, (8 - 1)}$$

$$= \frac{61}{112}$$

$$\alpha = 0.54$$

Setiap *bit-plane* menggunakan nilai batas / *threshold* α_0 = 0.3, setiap bit-plane yang memiliki $\alpha \geq \alpha_0$ merupakan daerah *noise-like* dan bit-plane yang memiliki $\alpha < \alpha_0$ merupakan daerah *informative*. Daerah *noise-like* merupakan daerah plane yang bisa disisipi pesan, sedangkan daerah *informative* merupakan daerah yang tidak bisa disisipi pesan. Hitung semua nilai kompleksitas setiap plane sehingga didapatkan nilai plane 0 sampai plane 7, yaitu :

Plane 7 memiliki nilai α = 0.54 $\geq$ 0.3 → *noise-like*

Plane 6 memiliki nilai α = 0.00 < 0.3 → *informative*

Plane 5 memiliki nilai α = 0.00 < 0.3 → *informative*

Plane 4 memiliki nilai α = 0.4 $\geq$ 0.3 → *noise-like*

Plane 3 memiliki nilai α = 0.14 < 0.3 → *informative*

Plane 2 memiliki nilai α = 0.51 ≥ 0.3 → *noise-like*

Plane 1 memiliki nilai α = 0.54 ≥ 0.3 → *noise-like*

Plane 0 memiliki nilai α = 0.46 ≥ 0.3 → *noise-like*

4. Pecah pesan kedalam blok berukuran 8x8. Sehingga masing-masing segmen berukuran 64 bit.

Pesan : ¥ -" J

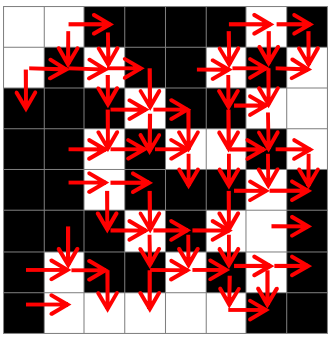
Konversi karakter pesan menjadi nilai biner

¥			-	"	
↓	↓	↓	↓	↓	↓
11000010	10100101	00010011	00101101	00100010	00010110

J	|
↓	↓
01001010	01111100

karena type data byte pada java terbatas hanya 8 bit (-128 hingga 127) maka setiap nilai yang melebihi 127 akan berubah menjadi negatif. Sehingga nilai ¥ → 165 dipecah menjadi 2 nilai yaitu -54 dan -123.

5. Hitung kompleksitas semua blok pesan



Gambar 3.8 Representasi Blok Pesan dalam Gambar Biner

Jumlah pergantian warna hitam-putih pada gambar 3.8 adalah

$$\alpha = \frac{k}{2 \times 2^n \times (2^n - 1)}$$

$$= \frac{63}{2 \times 2^8 \times (2^8 - 1)}$$

$$= \frac{63}{2 \times 8 \times (8 - 1)}$$

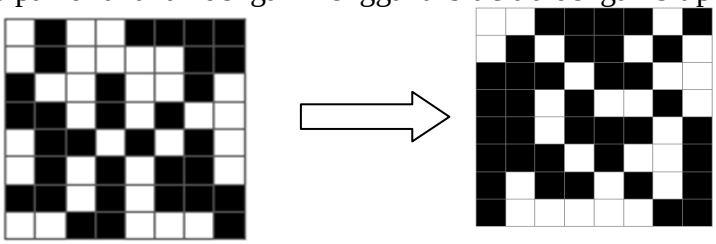
$$= \frac{63}{112}$$

$$\alpha = 0.56$$

Sebelum melakukan penyisipan, gambar biner yang merupakan representasi blok pesan tersebut dihitung nilai kompleksitasnya terlebih dahulu. Pada blok pesan , jumlah perubahan warna adalah 63 kali, sehingga diperoleh $\alpha = 0.56$. Karena blok pesan ini memiliki kompleksitas $\alpha \geq \alpha_0$ ($0.56 \geq 0.3$), maka blok bit-plane yang *noise-like* pada citra diganti oleh 64 bit pesan ini. Blok pesan akan mengganti blok citra terpilih yang ditandai dengan label berupa angka dari 0 sampai 7 untuk mengetahui di plane mana pesan akan disisipkan. Pada analisis ini blok plane citra yang dipakai untuk menggantikan blok pesan adalah plane 1.

6. Lakukan proses penyisipan. Sisipkan setiap blok pesan rahasia ke *bit-plane* yang merupakan daerah *noise-like* yang terpilih.

   Penyisipan dilakukan dengan mengganti bit citra dengan bit pesan.



Plane 1

Blok pesan rahasia

Sehingga didapat nilai piksel citra yang telah disisipi pesan yaitu :



2218221	2444904	2279912	2283756	2903672	3001080	2410985	3100281
2509673	2607849	2312300	2607593	3100025	2611432	2316909	3100283
2411241	2121448	2904041	2607592	2218988	2444012	2283880	2508908
2903672	2218988	2903673	2444137	2316653	2444152	2283757	2283756
2903673	2279525	2902781	2444137	2316653	2607224	2509284	2283753
2412153	2279917	2444024	2444905	2312301	2279544	2279917	2218344
2444905	2509305	2509677	2444025	2316521	2218212	2509305	2120044
2312936	2214009	2116069	2280184	2115689	6379748	2280316	6379756

Gambar 3.9 Nilai CGC potongan 8x8 dari citra stego

Tahap – tahap pada metode *Bit Plane Complexity Segmentation* tersebut diulang hingga semua pesan dapat terekstrak di dalam citra.

Ubah citra stego dari sistem CGC menjadi sistem PBC. Perubahan sistem CGC menjadi PBC yaitu dengan menggunakan persamaan $b_i = g_i$

XOR b_{i-1}. contoh piksel[0][0] pada sistem CGC yaitu 2218221. Maka perhitungan nilai pbc adalah sebagai berikut :

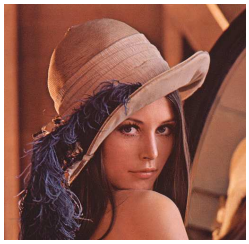
```
pbc[0][0] = (cgc[0][0]) XOR (pbc[0][0] Shr 1)
          = 2218221 XOR (-4100278 Shr 1)
          = 2218221 XOR -2050139
pbc[0][0] = -4100280
```



-4100280	-3771313	-3968690	-3969207	-3640240	-3574960	-3706546	-3509167
-3902897	-3837106	-4034487	-3837617	-3508656	-3838130	-4035511	-3509165
-3706034	-4166834	-3639985	-3837617	-4100791	-3771575	-3969457	-3903415
-3640240	-4100792	-3640240	-3771826	-4035000	-3771824	-3969208	-3969207
-3640240	-3968954	-3640490	-3771825	-4034999	-3837871	-3903161	-3969201
-3705776	-3968695	-3771567	-3771314	-4034488	-3968944	-3968695	-4100530
-3771314	-3903151	-3902904	-3771567	-4034737	-4100281	-3903151	-4166072
-4033714	-4100016	-4165306	-3968176	-4165553	-4296890	-3968426	-4296888

Gambar 3.10 Nilai PBC potongan 8x8 dari citra stego

7. Hasil citra yang telah disisipi pesan



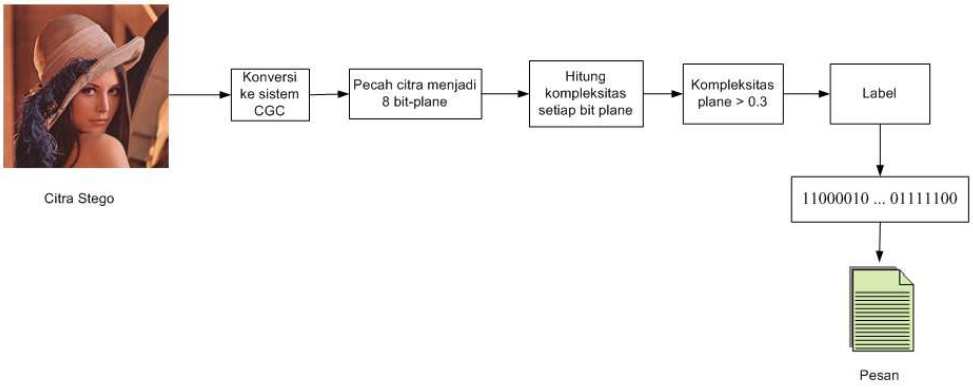
Gambar 3.11 Citra Stego

3.2.4 Proses Ekstraksi dan Dekripsi Pesan

Ekstraksi pesan menggunakan metode *Bit plane complexity segmentation* BPCS. Ekstraksi dilakukan untuk untuk memisahkan pesan yang tertanam dalam citra-stego. Sedangkan dekripsi menggunakan algoritma XTEA yang merupakan algoritma simetris, yang artinya kunci enkripsi sama dengan kunci dekripsi. Dekripsi dilakukan untuk mengubah pesan yang tersandikan menjadi pesan asli. Tahap-tahap yang untuk melakukan proses ekstraksi dan dekripsi pesan yaitu analisis enkrtaksi pesan dan analisis dekripsi pesan

3.2.4.1 Analisis Proses Pengungkapan / Ekstraksi

Proses Ekstraksi yaitu proses pengambilan informasi yang tersembunyi pada citra digital. Proses ini akan menghasilkan file informasi yang disembunyikan, dengan masukkan berupa citra *stego-object*. Proses ekstraksi pada metode bit plane complexity segmentation terlihat pada gamb



Gambar 3.12 Proses Ekstraksi Metode BPCS

Tahap awal pada proses ekstraksi pesan yaitu mengambil citra *stego-object*. Jika diketahui citra stego yang digunakan adalah LenaStego.bmp maka ubah citra tersebut dari sistem PBC menjadi sistem CGC.



Gambar 3.13 Citra Stego (LenaSteg.bmp)



-4100280	-3771313	-3968690	-3969207	-3640240	-3574960	-1586502	-3509167
-3902897	-3837106	-4034487	-3837617	-3508656	-3838130	-4035511	-3509165
-3706034	-4166834	-3639985	-3837617	-4100791	-3771575	-3969457	-3903415
-3640240	-4100792	-3640240	-3771826	-4035000	-3771824	-3969208	-3969207
-3640240	-3968954	-3640490	-3771825	-4034999	-3837871	-3903161	-3969201
-3705776	-3968695	-3771567	-3771314	-4034488	-3968944	-3968695	-4100530
-3771314	-3903151	-3902904	-3771567	-4034737	-4100281	-3903151	-4166072
-4033714	-4100016	-4165306	-3968176	-4165553	-4296890	-3968426	-4296888

Gambar 3.14 Nilai PBC potongan 8x8 dari citra LenaSteg.bmp

1. Ubah citra dari sistem PBC menjadi sistem CGC. Perubahan sistem PBC menjadi CGC yaitu dengan menggunakan persamaan $g_i = b_{i-1}$ XOR b_i, contoh piksel[0][0] pada sistem PBC yaitu -4100280. Maka perhitungan nilai cgc adalah sebagai berikut :

```
cgc[0][0] = (pbc[0][0] Shr 1) XOR pbc[0][0]
          = (-4100280 Shr 1) XOR -4100280
          = -2050140 XOR -4100278
cgc[0][0] = 2218220
```



2218220	2444904	2279913	2283757	2903672	3001080	2410985	3100281
2509672	2607849	2312301	2607592	3100024	2611433	2316909	3100282
2411241	2121449	2904040	2607592	2218989	2444013	2283880	2508909
2903672	2218988	2903672	2444137	2316652	2444152	2283756	2283757
2903672	2279525	2902781	2444136	2316653	2607225	2509284	2283752
2412152	2279917	2444025	2444905	2312300	2279544	2279917	2218345
2444905	2509305	2509676	2444025	2316520	2218212	2509305	2120044
2312937	2214008	2116069	2280184	2115688	6379749	2280317	6379756

Gambar 3.15 Nilai CGC potongan 8x8 dari citra LenaSteg.bmp

2. Tentukan segmen informatif dan *noise-like* dengan nilai batas/*threshold* yang sama dengan nilai batas/*threshold* ketika proses penyisipan yaitu 0.3.

   Nilai plane pada citra penampung adalah sebagai berikut :

   Plane 7 = 0.54 ≥ 0.3 → *noise-like*

   Plane 6 = 0.00 < 0.3 → *informative*

   Plane 5 = 0.00 < 0.3 → *informative*

   Plane 4 = 0.4 ≥ 0.3 → *noise-like*

   Plane 3 = 0.14 < 0.3 → *informative*

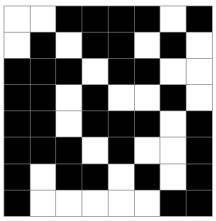
   Plane 2 = 0.51 ≥ 0.3 → *noise-like*

   Plane 1 = 0.56 ≥ 0.3 → *noise-like*

   Plane 0 = 0.49 ≥ 0.3 → *noise-like*

3. Ambil segmen data dari segmen *bit-plane* yang dianggap *noise-like* yang sudah diberi tanda label angka ketika proses penyisipan.

   Segmen yang diambil adalah segmen pada plane 1 yang merupakan blok data pesan yang sudah disisipkan.



Plane 0

4. Terjemahkan setiap segmen tersebut menjadi blok data

   Blok data pada plane 0 adalah

   11000010

   10100101

   00010011

   00101101

   00100010

   00010110

   01001010

   01111100

5. Ubah blok data biner menjadi nilai desimal dalam byte, sehingga diambil 8 bit data dan rangkai blok-blok tersebut menjadi pesan rahasia, nilai desimal tersebut diubah menjadi karakter.

¥			-	"		J	|
↓	↓	↓	↓	↓	↓	↓	↓
11000010	10100101	00010011	00101101	00100010	01001010	01111100	00010110

Tahap – tahap pada metode Bit Plane Complexity Segmentation tersebut diulang hingga semua pesan dapat terekstrak di dalam citra.

Sehingga didapatkan pesan rahasia yaitu **¥ -" J|fýÃ±?« A¨**

3.2.4.2 Analisis Proses Dekripsi Algoritma XTEA

Proses enkripsi pada algoritma XTEA dilakukan dengan cara mengambil setiap plaintext per blok 8 byte dan memecahnya menjadi round ganjil dan genap. Ulangi langkah tersebut sebanyak 32 round.

Contoh dekripsi pada algoritma XTEA

1. *Chipertext :* **¥ -" J|fýÃ±?« A¨**
2. Konversi karakter *plaintext* ke dalam bentuk desimal

¥		-	"		J	|	f
↓	↓	↓	↓	↓	↓	↓	↓
165	19	45	34	22	74	124	102
ý	**Ã**	**±**	**?**	**«**		**A**	**¨**
↓	↓	↓	↓	↓	↓	↓	↓
253	195	177	129	171	23	65	168

3. Masukkan kunci dekripsi yang sama dengn kunci enkripsi yaitu **1234567890123456** sehingga di dapat nilai penjadwalan kunci S[0..3] yaitu

   S[0] = 825373492

   S[1] = 892745528

   S[2] = 959459634

   S[3] = 859059510

4. Pecah plaintext per blok 8 byte. Untuk setiap blok dipecah menjadi round ganjil dan round genap yang masing-masing memiliki 4 byte karakter.

   ```
   v0 = ((165 AND 255) Shl 24) OR ((19 AND 255) Shl 16)
        OR ((45 AND 255) Shl 8)  OR ((34 AND 255))
   v0 = ((-1526726656) OR 1245184 OR 11520 OR 34 )
   ```

   v0 = -1525469918

   ```
   v1 = ((22 AND 255) Shl 24) OR ((74 AND 255) Shl 16)
        OR ((124 AND 255) Shl 8) OR ((102 AND 255))
   v1 = (369098752 OR 4849664 OR 31744 OR 102)
   ```

   v1 = 373980262

   Hitung sebanyak 32 round. Inisialisasikan DELTA dengan nilai 0x9E3779B9 (-1640531527 dalam integer) dan putaran (n) sebanyak 32 round. Hitung nilai v0 dan v1 dengan nilai awal n = 32 dan nilai awal sum = DELTA * round = -957401312. Perhitungan dilakukan dengan menggunakan nilai integer yang memiliki batas -2147483648 s/d 2147483648.

   ```
   v1 -= (((v0 Shl 4) XOR (v0 Shr 5) + v0) XOR
          (sum + S[sum Shr 11 AND 3])
   v1 -= (((-1525469918 Shl 4) XOR (-1525469918 Shr 5) +
   ```

```
       -1525469918) XOR (-957401312 + S[(-957401312)
       Shr 11 AND 3])
v1 -= (((1362285088 XOR -47670935) + (-1525469918))
       XOR (-957401312 + S[2])
v1 -= (-1407562935 + (-1525469918)) XOR (-957401312
       + 959459634)
v1 -= 1361934443 XOR 2058322
v1 -= 1362235449
v1  = 373980262 - 1362235449 = -988255187
sum -= DELTA
sum = (-957401312) – (-1640531527) = 683130215
v0 -= (((v1 Shl 4) XOR (v1 Shr 5) + v1) XOR
       (sum + S[sum AND 3])
v0 -= ((373980262  Shl 4) XOR (373980262 Shr 5)
       + 373980262 XOR (683130215 +
       S[683130215 AND 3])
v0 -= (((1688716896 XOR 11686883) + 373980262)
       XOR (683130215 + S[3])
v0 -= (1679136131 + 373980262) XOR
       (683130215 + 859059510)
v0 -= 2053116393 XOR 1542189725
v0 -= 562815860
v0  = (-1525469918) - 562815860 = -2088285778
```

ulangi perhitungan tersebut sampai selesai sehingga didapatkan nilai akhir v1 dan v0 yaitu

v0 = 1970170219 v1 = 1869422690

5. Ubah nilai integer dari hasil perhitungan menjadi karakter ASCII dengan melakukan pergeseran. Nilai desimal yang diambil berupa nilai byte (8 bit).

v0 Shr 24

`1970170219` Shr 24 = **117**

v0 Shr 16

`1970170219` Shr 16 = **110**

v0 Shr 8

`1970170219` Shr 8 = **105**

`1970170219` = **107**

v1 Shr 24

`1869422690` Shr 24 = **111**

v1 Shr 16

`1869422690` Shr 16 = **109**

v1 Shr 8

`1869422690` Shr 8 = **32**

`1869422690` = **98**

Desimal (byte)	117	110	105	107
Biner	01110101	01101110	01101001	01101011
Karakter	u	n	i	k

Desimal (byte)	111	109	32	98
Biner	01101111	01101101	00100000	01100010
Karakter	o	m	spasi	b

Sehingga di dapat 8 karakter *plaintext*. Ulangi perhitungan hingga didapat semua *plaintext.*

6. Maka *Plaintext* dari *Chipertext* : **¥ -" J|fýÃ±?« A¨**

   Plaintext : "**unikom bandung** "

3.3 Analisis Kebutuhan Non Fungsional

Analisis non fungsional bertujuan untuk menghasilkan spesifikasi kebutuhan non fungsional. Spesifikasi kebutuhan non fungsional adalah spesifikasi yang rinci mengenai hal-hal yang akan dilakukan sistem ketika di implementasikan. Analisis kebutuhan ini diperlukan untuk menentukan keluaran yang akan dihasilkan sistem, masukan yang diperlukan sistem, lingkup proses yang digunakan untuk mengolah masukan menjadi keluaran. Kebutuhan ini terdiri dari kebutuhan perangkat lunak (*software*) dan kebutuhan perangkat keras (*hardware*).

3.3.1 Analisis Pengguna

Untuk menggunakan perangkat lunak ini, dibutuhkan 2 pengguna dengan karakteristik seperti pada tabel 3.1.

Tabel 3.1 Kebutuhan Penggunaan Aplikasi

Tipe Pengguna	Hak Akses	Tingkat Pendidikan	Tingkat Keterampilan	Jenis Pelatihan
Pengguna	Memasukkan kunci enkripsi atau dekripsi pesan, dan juga melakukan penyisipan pesan kedalam citra dan ekstraksi pesan	Minimal SMA/SMK/Sederajat	Dapat mengoprasikan komputer dan memahami konsep steganografi dan kriptografi	Tidak diperlukan pelatihan khusus

3.3.2 Analisis Kebutuhan Perangkat Keras

Perangkat keras yang digunakan dalam membangun perangkat lunak ini memiliki spesifikasi sebagai berikut :

1. Processor Intel Core i3-2330M,
2. Memori RAM 2 Gb,
3. *Harddisk* 500 Gb,
4. Seperangkat I/O (*mouse, keyboard, monitor*)

Sedangkan spesifikasi minimum yang dibutuhkan untuk menjalankan perangkat lunak ini adalah sebagai berikut :

1. Processor Intel Core 2.26 GHz,
2. Memori RAM 1 Gb,
3. *Harddisk* 320 Gb,
4. Seperangkat I/O (*mouse, keyboard, monitor*)

3.3.3 Analisis Kebutuhan Perangkat Lunak

Kebutuhan software merupakan kebutuhan akan perangkat lunak yang digunakan untuk membangun program aplikasi steganografi ini. Diantara kebutuhan terebut adalah :

1. Sistem operasi berbasis windows, yang penulis gunakan adalah sistem operasi Windows 7 Ultimate 32-bit.
2. Tools / bahasa pemrograman, yang penulis gunakan adalah Netbeans IDE 6.8

3.4 Analisis Kebutuhan Fungsional

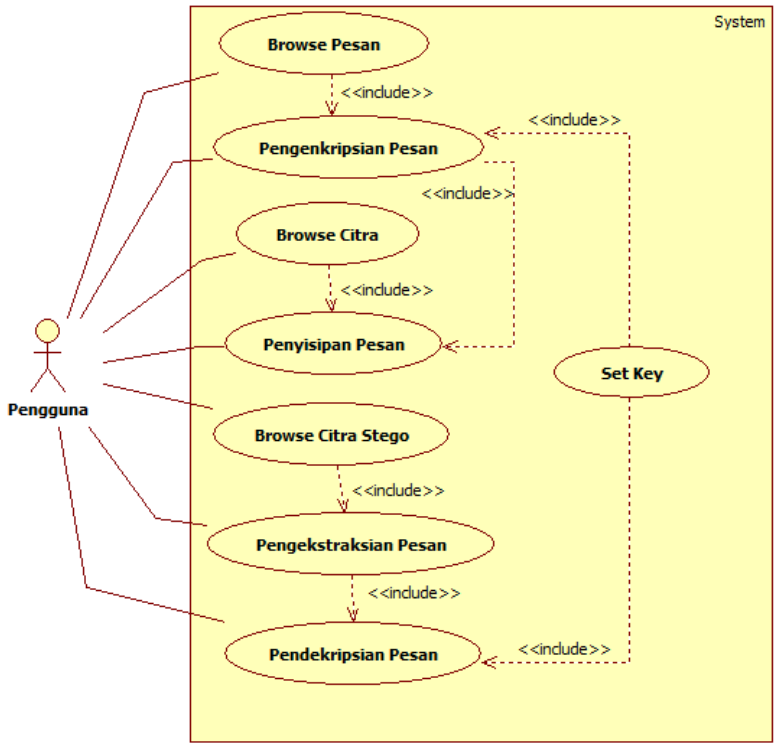
Analisi kebutuhan fungsional adalah segala bentuk data yang dibutuhkan oleh sistem agar sistem dapat berjalan sesuai dengan prosedur yang dibangun. Aplikasi yang dibangun akan dimodelkan menggunakan UML (*Unified Modeling Language*), dan *tools* yang akan digunakan yaitu *use case* diagram, *activity* diagram, *sequence* diagram, dan *class* diagram.

3.4.1 Use Case Diagram

Use case diagram menggambarkan suatu urutan interaksi antara satu atau lebih aktor dan sistem. Yang ditekankan adalah apa yang dilakukan oleh sistem/apa yang terjadi pada sistem bukan bagaimana sistem melakukan. *Use cas*e mempresentasikan sebuah interaksi antar *user* sebagai aktor dengan sistem. Seseorang atau sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali use case yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*.

Use case diagram dari program aplikasi steganografi ini digambarkan pada gambar 3.16



Gambar 3.16 *Use Case Diagram* Aplikasi Steganografi

3.4.2 Skenario *Use Case*

Skenario *use case* merupakan penjelasan secara rinci setiap fungsionalitas yang terdapat pada aplikasi. Skenario ini menjelaskan setiap use case yang terdapat pada diagram use case. Skenario use case pada aplikasi ini yaitu sebagai berikut:

1. Skenario *Set Key*

Skenario *use case set key* dapat dilihat pada tabel 3.2.

Tabel 3.2 Skenario *Use Case Set Key*

IDENTIFIKASI	
Nama	*Set Key*
Deskripsi	Pengguna memasukkan kunci untuk melakukan pengenkripsian dan pendekripsian pesan.
Actor	Pengguna
Pre-condition	Kunci belum terisi
Post-condition	Kunci sudah terisi

Aktor	Sistem
1. Memasukkan kunci	
	2. Sistem menerima masukkan kunci

2. Skenario *Browse* Pesan

Skenario *use case Browse* Pesan dapat dilihat pada tabel 3.3.

Tabel 3.3 Skenario *Use Case Browse* Pesan

IDENTIFIKASI		
Nama	*Browse* Pesan	
Deskripsi	*Browse* pesan merupakan proses yang dilakukan oleh pengguna. Untuk melakukan steganografi pengguna melakukan *browse* pesan/text yang akan disisipkan kedalam media penampung berupa citra digital.	
Actor	Pengguna	
Pre-condition	Pesan tidak tersedia	
Post-condition	Pesan tersedia	
Aktor		**Sistem**
1. Menekan tombol *browse* pesan		
		2. Sistem memunculkan *open file dialog* untuk memilih pesan yang akan disisipkan
3. Memilih pesan yang akan disisipkan		
		4. Sistem memunculkan pesan yang sudah dipilih

3. Skenario Pengenkripsian Pesan

Skenario *use case* pengenkripsian pesan dapat dilihat pada tabel 3.4.

Tabel 3.4 Skenario *Use Case* Enkripsi Pesan

IDENTIFIKASI		
Nama	Pengenkripsian Pesan	
Deskripsi	Pengguna melakukan enkripsi pesan terhadap pesan sebelum disisipkan ke dalam citra untuk menambah keamanan data.	
Actor	Pengguna	
Pre-condition	Pesan dapat terbaca atau belum terenkripsi	
Post-condition	Pesan tidak terbaca atau sudah terenkripsi	
Aktor		**Sistem**
1. Memasukkan kunci enkripsi		
		2. Sistem menerima kunci enkripsi
3. Masukkan pesan		
		4. Sistem menampilkan file pesan

5. Menekan tombol Enkripsi Pesan	
	6. Sistem mengenkripsi pesan

4. Skenario *Browse* Citra

Skenario *use case browse* citra dapat dilihat pada tabel 3.5.

Tabel 3.5 Skenario *Use Case Browse* Citra

IDENTIFIKASI	
Nama	*Browse* Citra
Deskripsi	Pengguna memilih citra yang akan digunakan sebagai media penampung pesan.
Actor	Pengguna
Pre-condition	Citra tidak tersedia
Post-condition	Tersedia citra untuk menampung pesan

Aktor	**Sistem**
1. Menekan tombol *Browse* citra	
	2. Sistem memunculkan *open file dialog* untuk memilih citra yang akan digunakan sebagai media penampung
2. Memilih citra penampung	
	3. Sistem memunculkan citra yang dipilih

5. Skenario Penyisipan Pesan

Skenario *use case* penyisipan pesan dapat dilihat pada tabel 3.6.

Tabel 3.6 Skenario *Use Case* Penyisipan Pesan

IDENTIFIKASI	
Nama	Penyisipan Pesan
Deskripsi	Pengguna menyisipkan pesan ke dalam citra digital sehingga tidak bisa terbaca oleh pihak lain yang tidak berhak mengetahuinya. Untuk menambah keamanan pesan, pesan terlebih dahulu dienkripsi sebelum disisipkan ke dalam citra.
Actor	Pengguna
Pre-condition	Pesan dapat terbaca oleh semua pihak
Post-condition	- Pesan sudah terenkripsi - Pesan tersembunyi di dalam sebuah citra

Aktor	**Sistem**
1. Memasukkan kunci enkripsi	
	2. Sistem menerima kunci enkripsi
3. Menekan tombol *browse* pesan	
	4. Sistem memunculkan *open file dialog* untuk memilih pesan yang akan disisipkan
5. Memilih pesan yang akan disisipkan	

	6. Sistem memunculkan pesan yang akan disisipkan
7. Menekan tombol Enkripsi Pesan	
	8. Sistem melakukan pengacakan kunci dan mengenkripsi pesan
9. Menekan tombol *browse* citra	
	10. Sistem memunculkan *open file dialog* untuk memilih citra yang akan digunakan sebagai media penampung
11. Memilih citra yang akan dijadikan media penampung	
	12. Sistem menampilkan citra yang digunakan sebagai media penampung
13. Menekan tombol Penyisipan	
	14. Sistem menyisipkan pesan yang sudah terenkripsi ke dalam citra penampung

6. Skenario *Browse* Citra Stego

Skenario *use case browse* citra stego dapat dilihat pada tabel 3.7.

Tabel 3.7 Skenario *Use Case Browse* Citra

IDENTIFIKASI	
Nama	*Browse* Citra Stego
Deskripsi	Pengguna memilih citra stego yang telah tersisipi pesan.
Actor	Pengguna
Pre-condition	Citra tidak tersedia
Post-condition	Tersedia citra stego

Aktor	**Sistem**
1. Menekan tombol *Browse* citra stego	
	2. Sistem memunculkan *open file dialog* untuk memilih citra stego
3. Memilih citra stego	
	4. Sistem memunculkan citra stego yang dipilih

7. Skenario Pengekstraksian Pesan

Skenario *use case* Pengekstraksian Pesan dapat dilihat pada tabel 3.8.

Tabel 3.8 Skenario *Use Case* Ekstraksi Pesan

IDENTIFIKASI	
Nama	Pengekstraksian Pesan
Deskripsi	Pengguna melakukan ekstraksi file yaitu memisahkan pesan dari dalam citra stego sehingga pesan bisa dibaca oleh pihak yang berhak mengetahuinya. Sebelumnya pesan tersebut harus di dekripsi terlebih dahulu.

Actor	Pengguna
Pre-condition	- Pesan dalam bentuk terenkripsi - Pesan tersembunyi dalam sebuah citra
Post-condition	Pesan terpisah dari citra stego

Aktor	**Sistem**
1. Menekan tombol *browse* citra stego	
	2. Sistem memunculkan *open file dialog* untuk memilih citra stego
3. Memilih citra stego	
	4. Sistem menampilkan citra stego
5. Menekan tombol Ekstrak	
	6. Sistem memunculkan pesan yang terenkripsi yang tersisip di dalam citra stego

8. Skenario Pendekripsian Pesan

Skenario *use case* Pendekripsian Pesan dapat dilihat pada tabel 3.9.

Tabel 3.9 Skenario *Use Case* Dekripsi Pesan

IDENTIFIKASI	
Nama	Pendekripsian Pesan
Deskripsi	Sebelum pesan yang sudah di ekstraksi dari dalam citra bisa dibaca maka pesan tersebut harus di dekripsi terlebih dahulu
Actor	Pengguna
Pre-condition	Pesan tidak dapat terbaca atau dalam bentuk terenkripsi
Post-condition	Pesan dapat dibaca

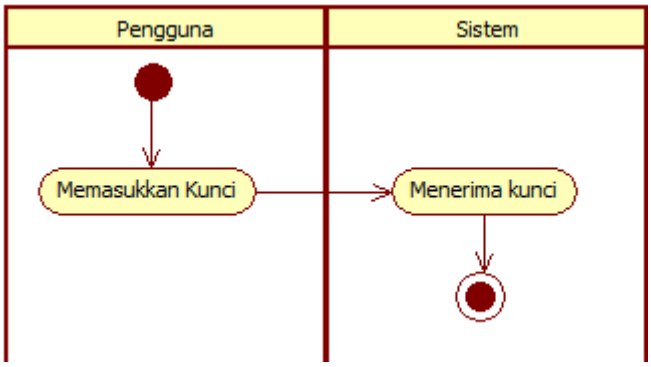
Aktor	**Sistem**
1. Menekan tombol dekripsi pesan	
	2. Sistem mendekripsi pesan yang sudah diambil dari citra stego

3.4.3 Activity Diagram

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing – masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity* diagram yang terdapat pada aplikasi yang dibangun yaitu sebagai berikut:

1. Activity Diagram *Set Key*

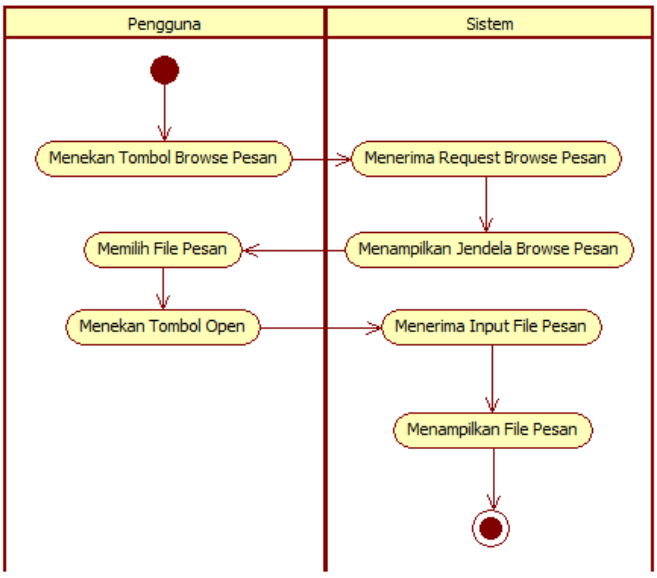
Activity diagram *set key* menggambarkan alir aktivitas memasukkan kunci untuk enkripsi atau dekripsi antara pengguna dangan sistem seperti terlihat pada gambar 3.17.



Gambar 3.17 *Activity* Diagram *Set Key*

2. Activity Diagram *Browse* Pesan

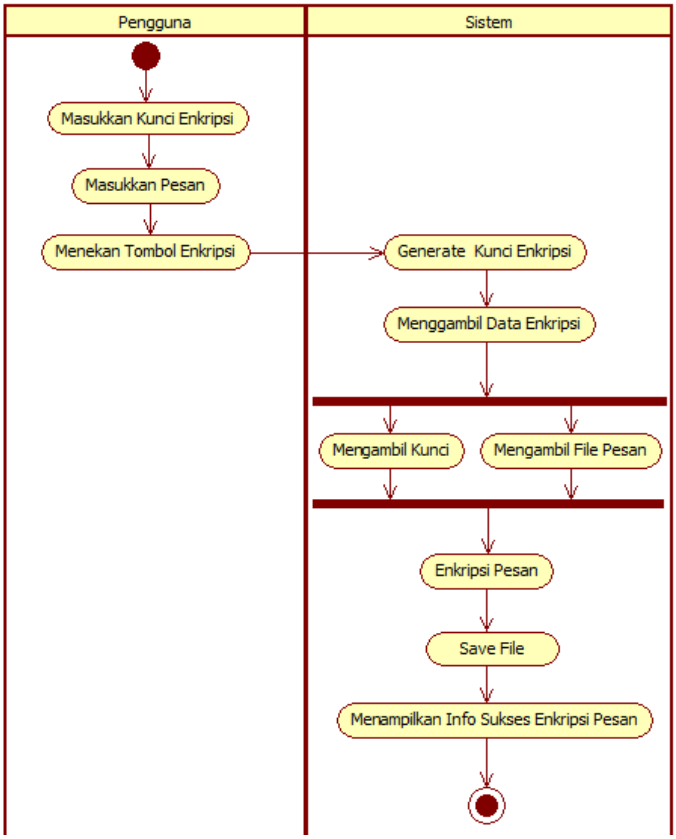
Activity diagram *Browse* Pesan menggambarkan alir aktivitas pengambilan *file* pesan antara pengguna dangan sistem seperti terlihat pada gambar 3.18.



Gambar 3.18 *Activity* Diagram *Browse* Pesan

3. Activity Diagram Pengenkripsian Pesan

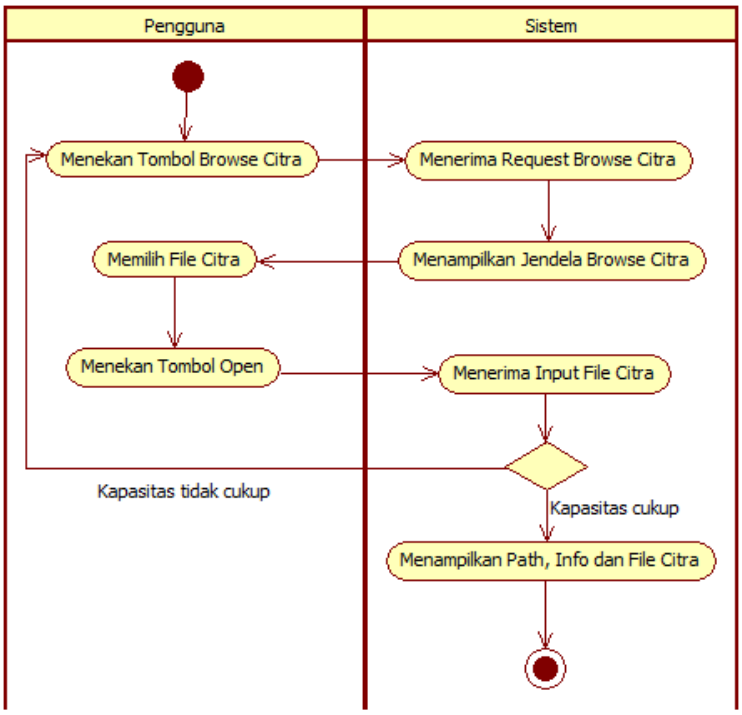
Activity diagram pengenkripsian pesan menggambarkan alir aktivitas enkripsi yang dilakukan antara pengguna dangan sistem seperti terlihat pada gambar 3.19.



Gambar 3.19 *Activity Diagram* Pengenkripsian Pesan

4. Activity Diagram *Browse* Citra

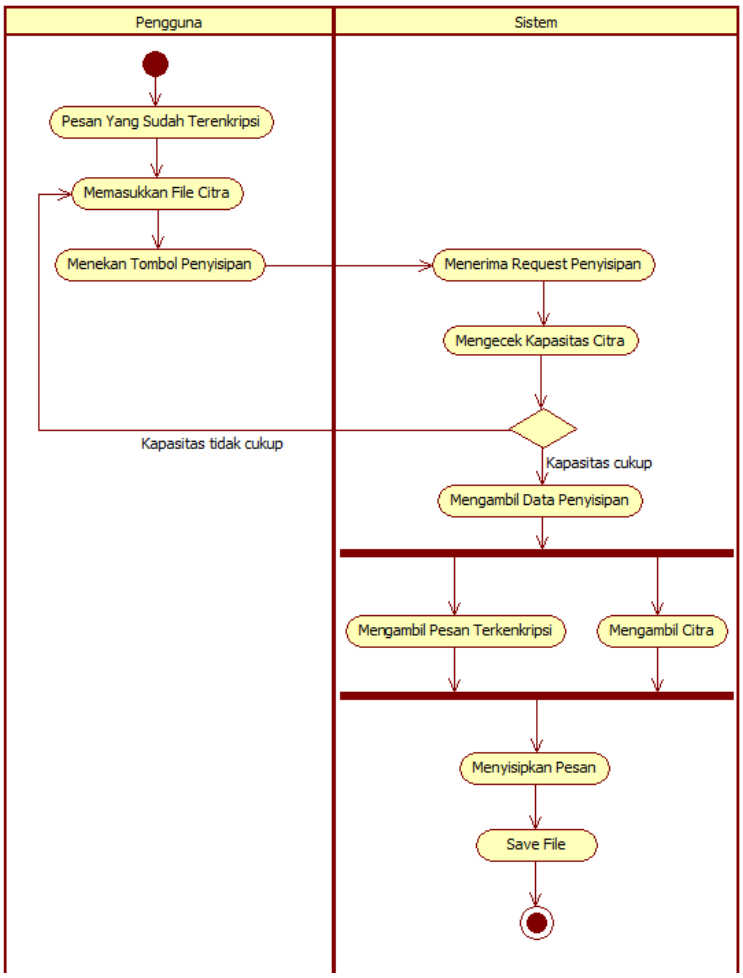
Activity diagram *Browse* Citra menggambarkan alir aktivitas pengambilan *file* citra antara pengirim dangan sistem seperti terlihat pada gambar 3.20.



Gambar 3.20 *Activity* Diagram *Browse* Citra

5. Activity Diagram Penyisipan Pesan

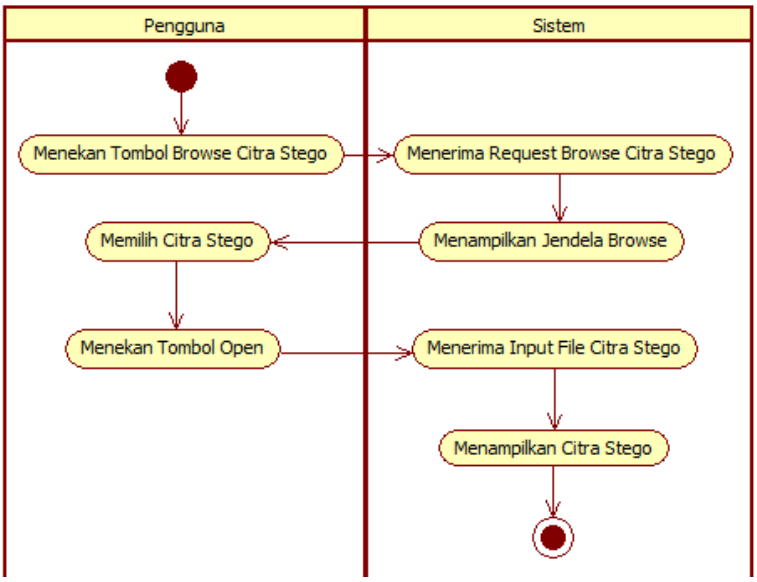
Activity diagram Penyisipan Pesan menggambarkan alir aktivitas penyisipan pesan ke dalam citra antara pengguna dangan sistem seperti terlihat pada gambar 3.21.



Gambar 3.21 *Activity* Diagram Penyisipan Pesan

6. Activity Diagram *Browse* Citra Stego

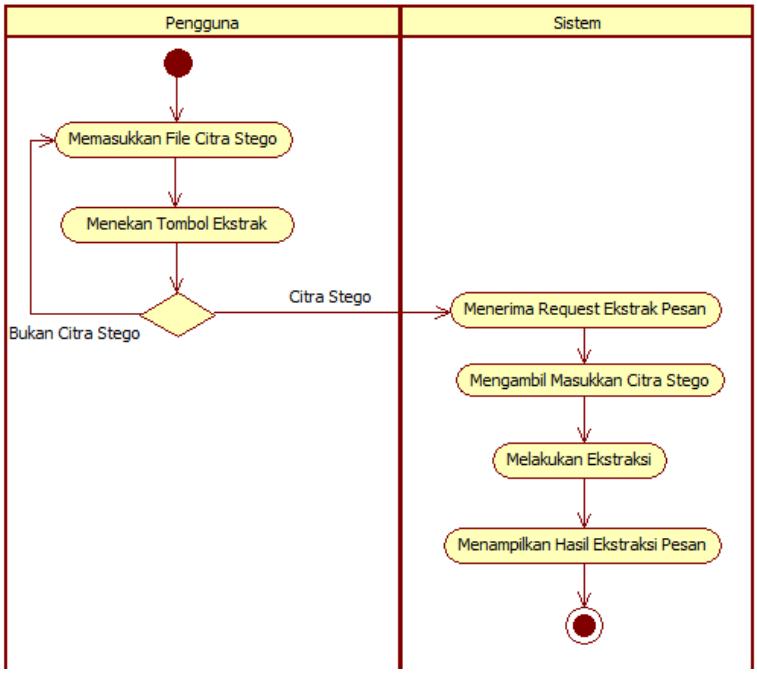
Activity diagram *Browse* Citra menggambarkan alir aktivitas pengambilan *file* citra stego antara pengirim dangan sistem seperti terlihat pada gambar 3.22.



Gambar 3.22 *Activity* Diagram *Browse* Citra Stego

7. Activity Diagram Pengekstraksian Pesan

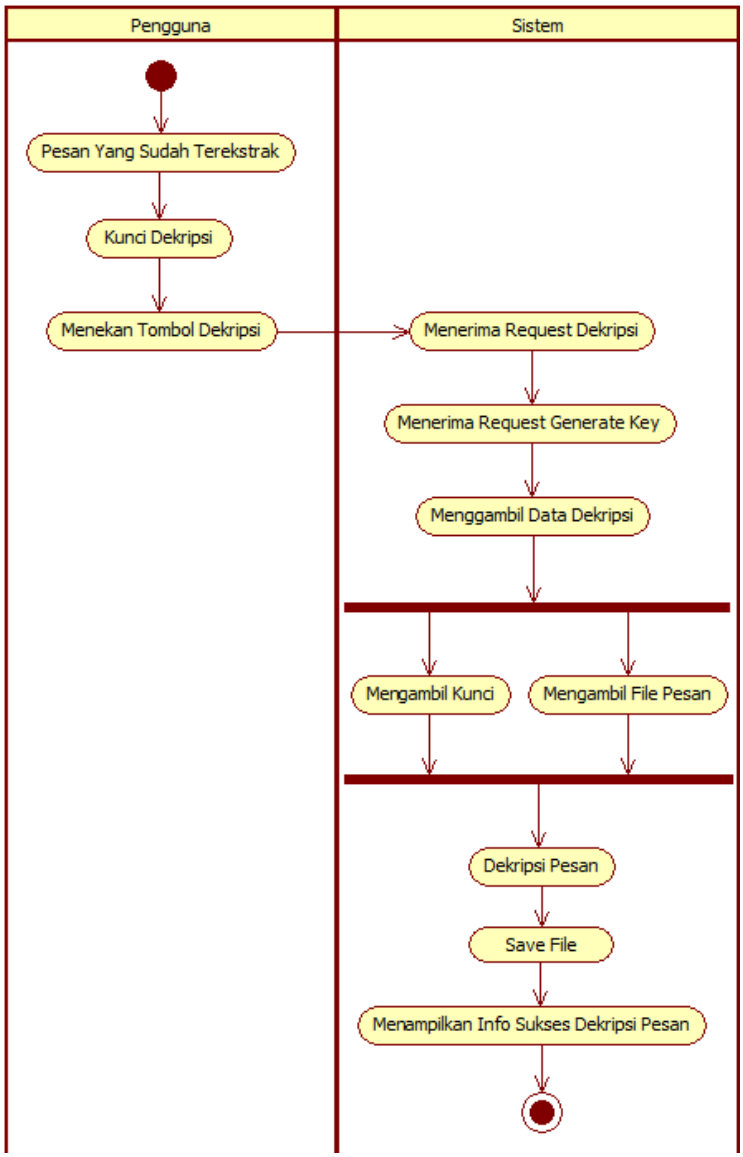
Activity diagram Ekstraksi Pesan menggambarkan alir aktivitas pemisahan pesan yang ada di dalam cira stego antara pengirim dangan sistem seperti terlihat pada gambar 3.23.



Gambar 3.23 *Activity Diagram* Pengekstraksian Pesan

8. Activity Diagram Pendekripsian Pesan

Activity diagram Pendekripsian Pesan menggambarkan alir aktivitas dekripsi pesan antara pengguna dangan sistem seperti terlihat pada gambar 3.24



Gambar 3.24 *Activity Diagram* Pendekripsian Pesan

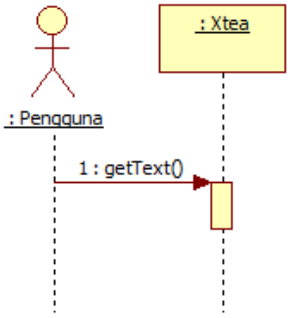
3.4.4 Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) yang

digambarkan terhadap waktu. *Sequence* diagram yang terdapat pada aplikasi steganografi yaitu sebagai berikut:

1. *Sequence Diagram Set Key*

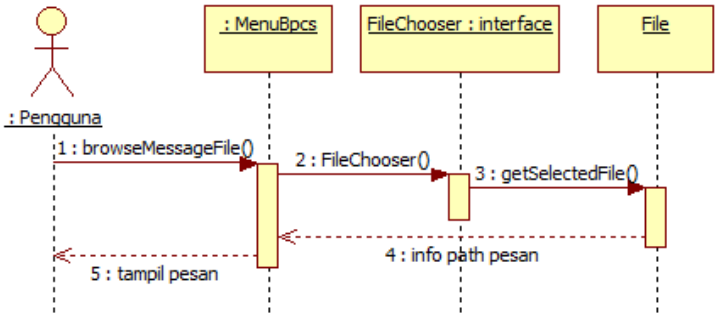
Sequence diagram *set key* merupakan diagram yang mengambarkan interaksi yang terjadi didalam sistem antara pengguna dengan sistem dalam memasukkan kunci. *Sequence* diagram *set key* aplikasi steganografi terlihat seperti pada gambar 3.25.



Gambar 3.25. *Sequence* Diagram *Set Key*

2. *Sequence Diagram Browse* Pesan

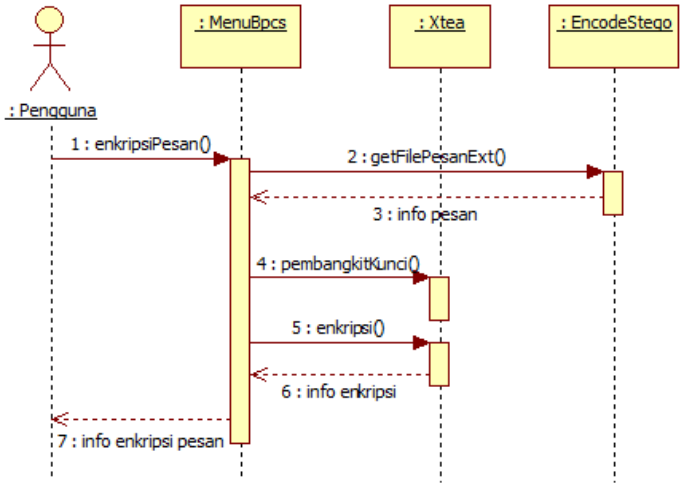
Sequence diagram *browse* pesan merupakan diagram yang mengambarkan interaksi yang terjadi didalam sistem antara pengirim dengan sistem dalam pengambilan pesan. *Sequence* diagram *browse* pesan aplikasi steganografi terlihat seperti pada gambar 3.26.



Gambar 3.26. *Sequence* Diagram *Browse* Pesan

3. *Sequence Diagram* Pengenkripsian Pesan

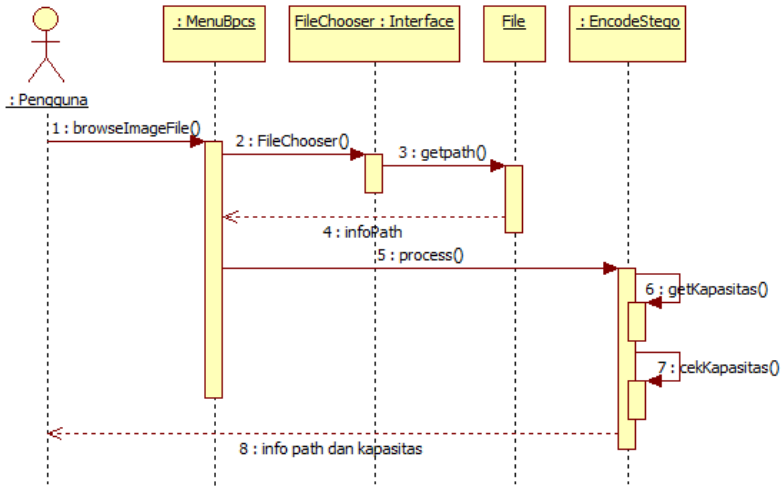
Sequence diagram pengenkripsian pesan merupakan diagram yang mengambarkan interaksi yang terjadi didalam sistem antara pengirim dengan sistem dalam mengenkripsi pesan. *Sequence* diagram enkripsi aplikasi steganografi terlihat seperti pada gambar 3.27.



Gambar 3.27. *Sequence* Diagram Pengenkripsian Pesan

4. *Sequence Diagram Browse* Citra

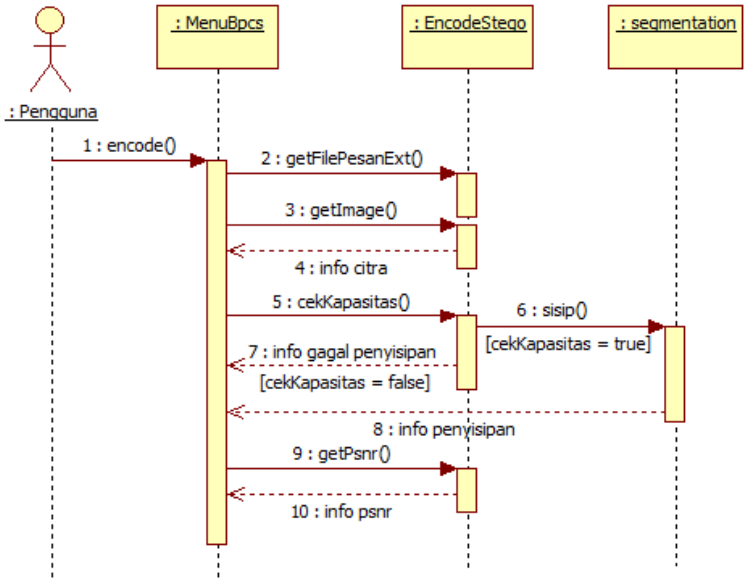
Sequence diagram *browse* citra merupakan diagram yang mengambarkan interaksi yang terjadi didalam sistem antara pengirim dengan sistem dalam pengambilan citra. *Sequence* diagram *browse* citra aplikasi steganografi terlihat seperti pada gambar 3.28.



Gambar 3.28. *Sequence* Diagram *Browse* Citra

5. *Sequence Diagram* Penyisipan Pesan

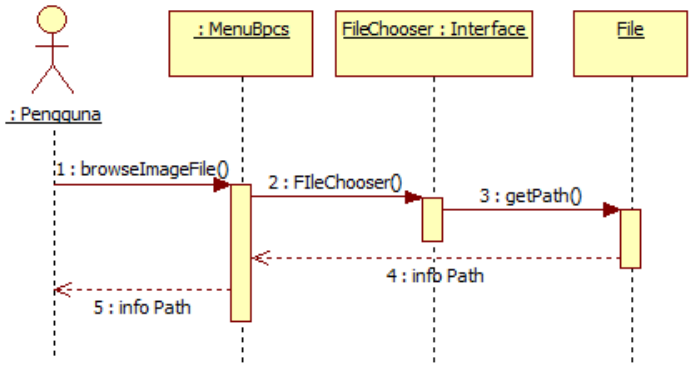
Sequence diagram penyisipan merupakan diagram yang mengambarkan interaksi yang terjadi didalam sistem antara pengirim dengan sistem dalam penyisipan pesan kedalam citra. *Sequence* diagram penyisipan aplikasi steganografi terlihat seperti pada gambar 3.29.



Gambar 3.29. *Sequence* Diagram Penyisipan Pesan.

6. *Sequence Diagram Browse* Citra Stego

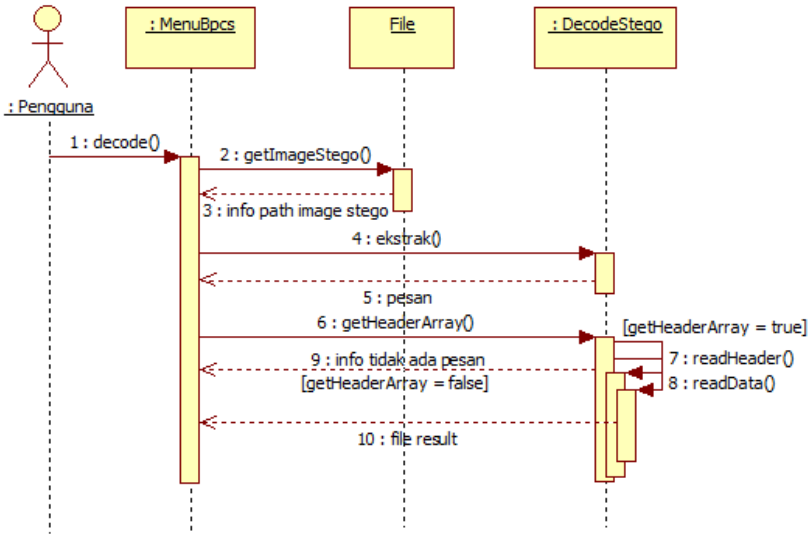
Sequence diagram *browse* citra stego merupakan diagram yang mengambarkan interaksi yang terjadi didalam sistem antara pengirim dengan sistem dalam pengambilan citra stego. *Sequence* diagram *browse* citra stego aplikasi steganografi terlihat seperti pada gambar 3.30.



Gambar 3.30. *Sequence* Diagram *Browse* Citra Stego

7. *Sequence Diagram* Pengekstraksian Pesan

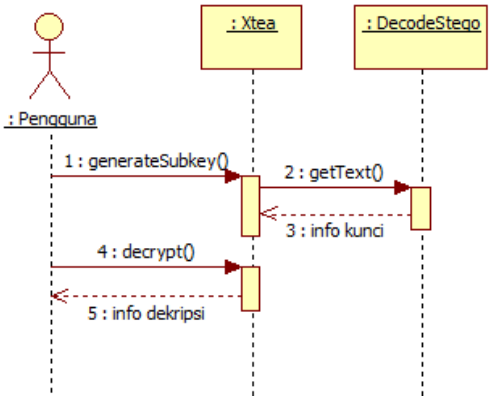
Sequence diagram pengekstraksian pesan merupakan diagram yang mengambarkan interaksi yang terjadi didalam sistem antara penerima dengan sistem dalam ekstraksi pesan dari citra. *Sequence* diagram ekstraksi pesan aplikasi steganografi terlihat seperti pada gambar 3.31.



Gambar 3.31. *Sequence* Diagram Pengekstraksian Pesan.

8. *Sequence Diagram* Pendekripsian Pesan

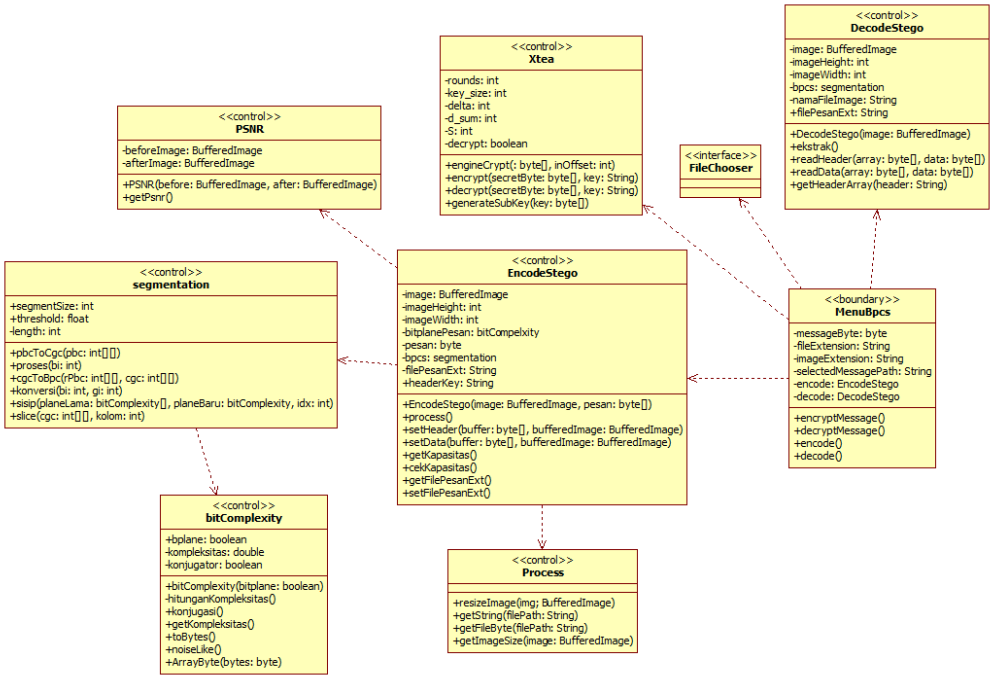
Sequence diagram pendekripsian pesan merupakan diagram yang mengambarkan interaksi yang terjadi didalam sistem antara penerima dengan sistem dalam dekripsi pesan. *Sequence* diagram dekripsi pesan aplikasi steganografi terlihat seperti pada gambar 3.32.



Gambar 3.32. *Sequence* Diagram Pendekripsian Pesan.

3.4.5 Class Diagram

Class diagram adalah dekripsi kelompok objek-objek dengan properti, perilaku (operasi) dan relasi yang sama. Sehingga dengan adanya *class diagram* dapat memberikan pandangan global atas sebuah sistem. Hal tersebut tercermin dari *class-class* yang ada dan relasinya satu dengan yang lainnya. Sebuah sistem biasanya mempunyai beberapa *class diagram*. *Class diagram* sangat membantu dalam visualisasi struktur kelas dari suatu sistem. *Class diagram* dari program aplikasi steganografi digambarkan seperti gambar 3.33 :



Gambar 3.33 *Class Diagram* Aplikasi Steganografi

3.5 Perancangan Sistem

Perancangan merupakan tahap lanjutan dari analisis sistem dimana pada perancangan sistem digambarkan rancangan sistem yang akan dibangun sebelum dilakukan pengkodean kedalam suatu bahasa pemograman. Dalam perancangan sistem tidak lepas dari hasil analisis, karena dari hasil analisis sistem baru dapat dibuat suatu perancangan sistem.

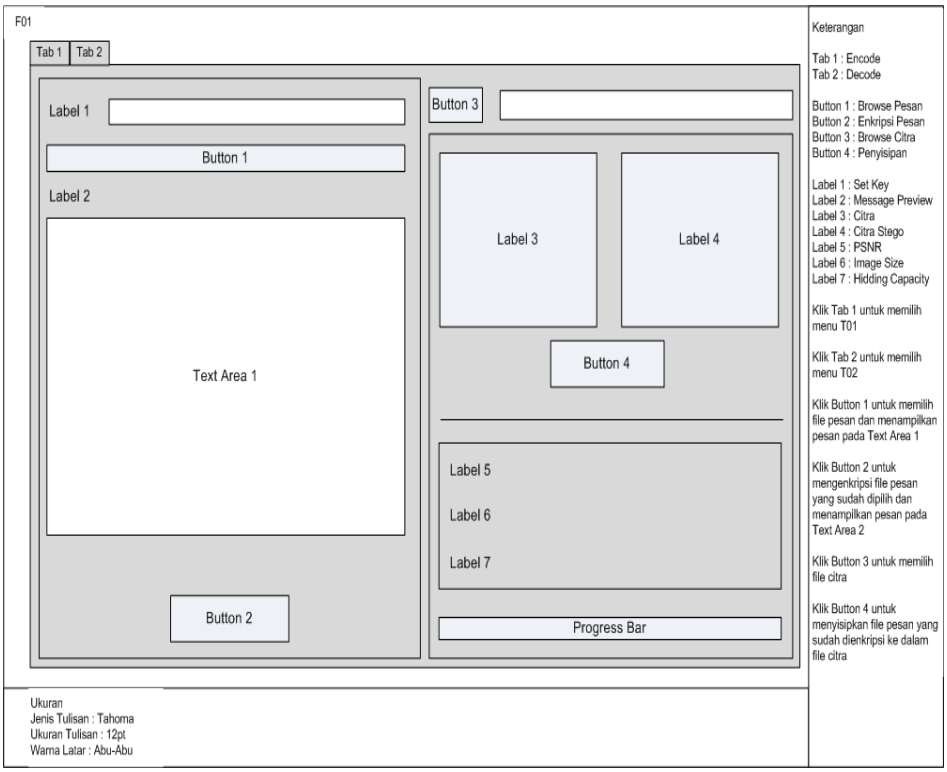
Desain umum yang akan diaplikasikan bertujuan untuk memberikan gambaran secara umum kepada penggunaan tentang sistem yang akan dibangun.

3.5.1 Perancangan Antarmuka

Perancangan antarmuka adalah tahapan pembuatan rancangan antarmuka untuk digunakan pada pembangunan aplikasi steganografi yang dibagi menjadi tigabagian yaitu perancangan antarmuka penyisipan, perancangan antarmuka ekstraksi, dan Perancangan Pesan.

1. Perancangan Antarmuka Penyisipan (F01)

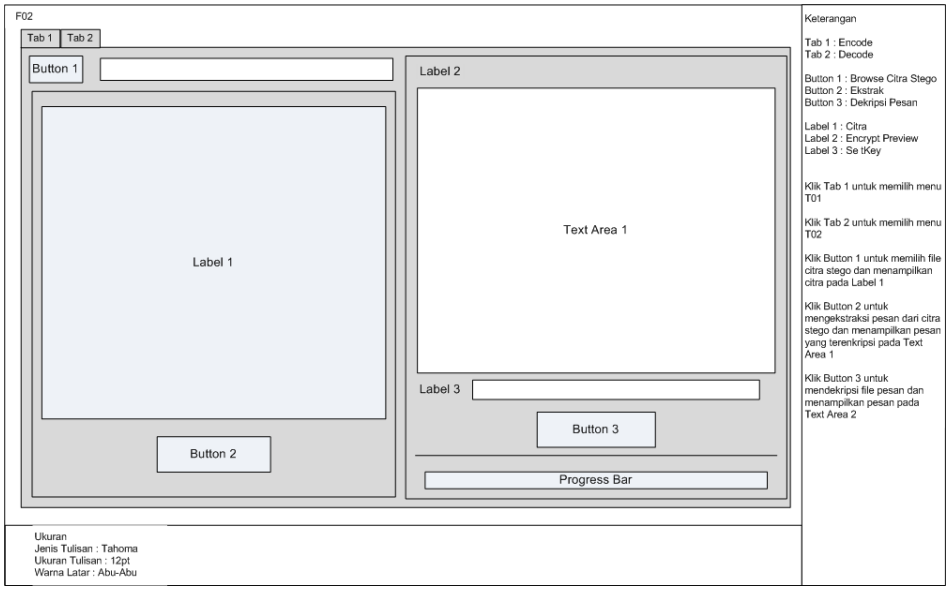
Perancangan antarmuka penyisipan merupakan tahapan rancangan antarmuka untuk menyisipkan pesan kedalam citra. Perancangan antarmuka Penyisipan dari aplikasi steganografi ini terlihat pada gambar 3.34.



Gambar 3.34 Perancangan Antarmuka Penyisipan

2. Perancangan Antarmuka Ekstraksi (F02)

Perancangan antarmuka ekstraksi merupakan tahapan rancangan antarmuka untuk ekstraksi pesan yang ada di dalam citra. Perancangan antarmuka ekstraksi dari aplikasi steganografi ini terlihat pada gambar 3.35



Gambar 3.35 Perancangan Antarmuka Ekstraksi

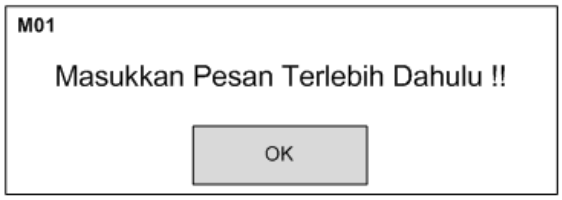
3.5.2 Perancangan Pesan

Pesan merupakan tampilan dari suatu perangkat lunak yang berfungsi untuk menyampaikan notifikasi dan informasi kepada pengguna agar perangkat lunak lebih interaktif. Perancangan pesan pada aplikasi steganografi dibagi menjadi dua bagian yaitu perancangan pesan menu penyisipan dan perancangan pesan menu ekstraksi.

Berikut perancangan pesan yang ada pada perangkat lunak steganografi pada aplikasi ini:

1. Perancangan pesan jika pesan belum dipilih (M01)

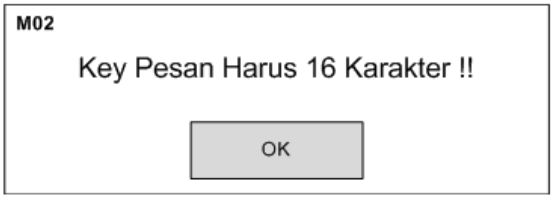
Pesan ini muncul jika pengguna menekan tombol Enkripsi sebelum memilih pesan. Perancangan pesan ini terlihat pada gambar 3.36.



Gambar 3.36. Perancangan Pesan Jika Belum Memilih Pesan.

2. Perancangan pesan informasi memasukkan kunci (M02)

Pesan ini muncul jika pengguna menekan tombol Enkripsi sebelum mengisi kunci atau kunci kurang dari 16 karakter. Perancangan pesan ini terlihat pada gambar 3.37.



Gambar 3.37. Perancangan Pesan Jika Belum Mengisi Kunci

3. Perancangan pesan jika citra belum diisi (M03).

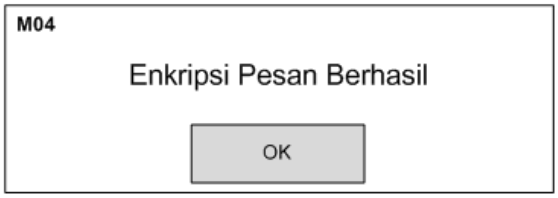
Pesan ini muncul jika citra belum dimasukkan. Perancangan pesan ini terlihat pada gambar 3.38.



Gambar 3.38. Perancangan Pesan Jika Citra Belum Dipilih

4. Perancangan pesan informasi enkripsi berhasil (M04).

Pesan ini muncul jika enkripsi berhasil dilakukan. Perancangan pesan ini terlihat pada gambar 3.39.



Gambar 3.39. Perancangan Pesan Enkripsi Berhasil.

5. Perancangan pesan jika kapasitas citra tidak mencukupi (M05).

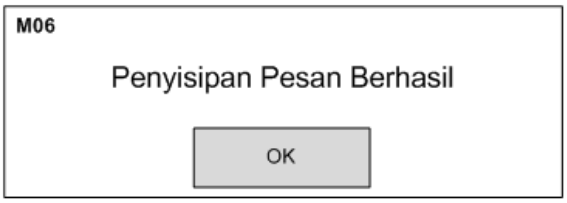
Pesan ini muncul jika kapasitas citra tidak mencukupi untuk menyisipkan pesan pada saat proses penyisipan. Perancangan pesan ini terlihat pada gambar 3.40.



Gambar 3.40. Perancangan Pesan Kapasitas Tidak Cukup.

6. Perancangan pesan informasi penyisipan berhasil (M06).

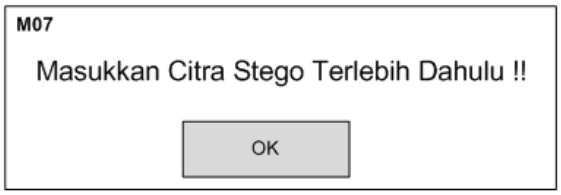
Pesan ini muncul jika penyisipan telah berhasil dilakukan.. Perancangan pesan ini terlihat pada gambar 3.41.



Gambar 3.41. Perancangan Pesan Penyisipan Berhasil.

7. Perancangan pesan jika citra stego belum diisi (M07).

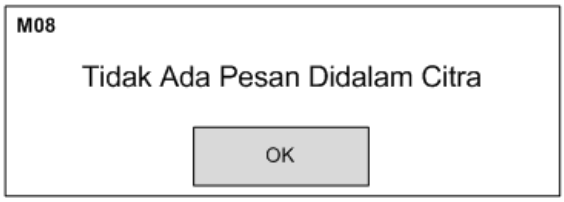
Pesan ini muncul jika citra stego belum dimasukkan. Perancangan pesan ini terlihat pada gambar 3.42.



Gambar 3.42. Perancangan Pesan Jika Citra Belum Dipilih

8. Perancangan pesan jika bukan citra stego (M08).

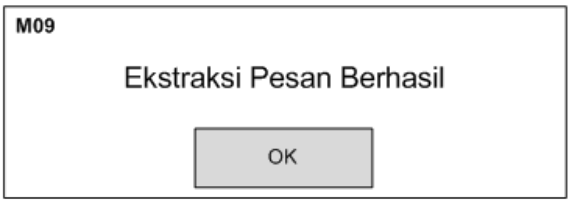
Pesan ini muncul jika citra yang dimasukkan bukan citra stego. Perancangan pesan ini terlihat pada gambar 3.43.



Gambar 3.43. Perancangan Pesan Bukan Citra Stego

9. Perancangan pesan ekstraksi pesan (M09).

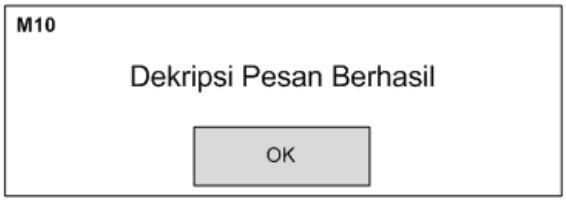
Pesan ini muncul jika ekstraksi pesan berhasil dilakukan. Perancangan pesan ini terlihat pada gambar 3.44.



Gambar 3.44. Perancangan Pesan Ekstraksi Berhasil.

10. Perancangan pesan dekripsi pesan (M10).

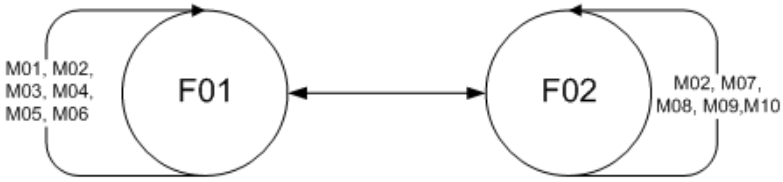
Pesan ini muncul jika dekripsi pesan berhasil dilakukan. Perancangan pesan ini terlihat pada gambar 3.45.



Gambar 3.45. Perancangan Pesan Dekripsi Berhasil.

3.5.3 Perancangan Jaringan Semantik

Jaringan semantik adalah diagram yang menggambarkan aliran-aliran menu dan pesan dalam sebuah program. Jaringan semantik dari aplikasi steganografi yang dibangun yaitu seperti pada gambar 3.46.



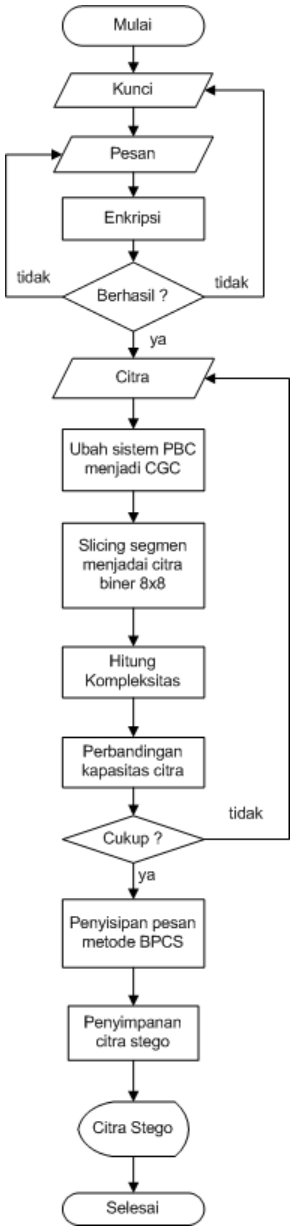
Gambar 3.46 Jaringan Semantik Aplikasi Steganografi

3.6 Perancangan *Method*

Perancangan *method* merupakan tahap akhir dalam perancangan sistem. Perancangan method digunakan untuk memberikan gambaran algoritma untuk diimplementasikan kedalam aplikasi yang dibangun. Perancangan *method* pada aplikasi ini dibagi menjadi dua bagian, yaitu perancangan *method* penyisipan, dan perancangan *method* ekstraksi.

3.6.1 Perancangan Method Penyisipan

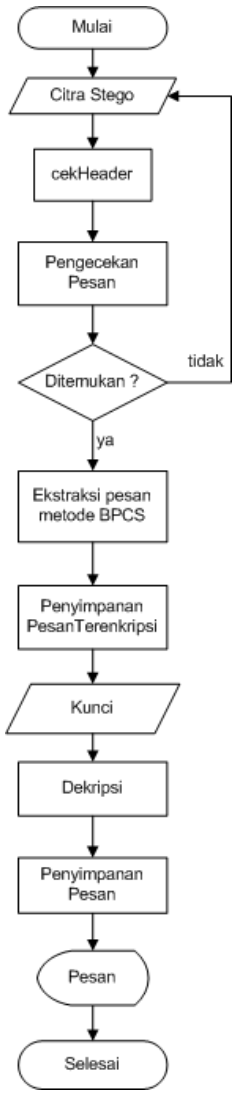
Perancangan *method* penyisipan merupakan gambaran umum dari langkah – langkah proses penyisipan di aplikasi steganografi yang dibangun. Perancangan method penyisipan pada aplikasi steganografi digambarkan dalam *flowchart method* penyisipan seperti pada gambar 3.47.



Gambar 3.47. *Flowchart Method* Penyisipan

3.6.2 Perancangan Method Ekstraksi

Perancangan *method* ekstraksi merupakan gambaran umum dari langkah – langkah proses ekstraksi di aplikasi steganografi yang dibangun. Perancangan *method* ekstraksi pada aplikasi steganografi digambarkan dalam *flowchart method* ekstraksi seperti pada gambar 3.48.



Gambar 3.48. *Flowchart Method* Ekstraksi