BAB III

ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Program aplikasi ini dapat dibuat dengan adanya suatu analisis terlebih dahulu. Analisis sistem dilakukan dengan mendefinisikan masalah yang ada, menganalisis dan mengamati secara keseluruhan bagaimana program aplikasi ini terbentuk. Dalam pembuatan program aplikasi ini diperlukan suatu analisis yang benar agar dihasilkan suatu program aplikasi yang sesuai dengan yang diharapkan.

Secara umum analisis yang akan dilakukan adalah mengenai lingkungan dari sistem dimana program aplikasi ini akan digunakan dan siapa yang menggunakannya. Analisis lingkungan dari sistem menguraikan sejumlah entitas yang ada hubungannya dengan sistem dan berpengaruh terhadap sistem.

3.2 Analisis Masalah

Steganografi bertujuan untuk menyisipkan suatu data/pesan rahasia kedalam suatu media penampung agar data/pesan rahasia tersebut tidak dapat dibaca atau diketahui oleh orang/pihak yang tidak berhak mengetahuinya.

Kebanyakan steganografi diterapkan pada media citra yang memiliki format .bmp atau bitmap. Masalah yang timbul adalah bagaimana jika data/pesan rahasia tersebut disisipkan pada citra yang berformat .bmp atau bitmap yang memiliki ukuran file yang relative besar sehingga dapat menjadi suatu hambatan

apabila kita menggunakan *electronic mail (e-mail)* untuk mendistribusikan citra yang berisi data/pesan rahasia tersebut.

Pada sistem yang akan direalisasikan ini akan mengatasi masalah tersebut, yaitu menyisipkan data/pesan rahasia pada media penampung yang berupa citra digital yang memiliki format gif,jpeg,png dan bmp.

3.2.1 Analisis Pengolahan Citra Digital

Citra merupakan fungsi intensitas dalam bidang dua dimensi. Intensitas yang dimaksud berasala dari sumber cahaya. Pada hakekatnya citra yang dilihat oleh mata manusia terdiri atas berkas-berkas cahaya yang dipantulkan oleh benda-benda disekitar kita.

Suatu citra digital adalah suatu gambar kontinu yang diubah dalam bentuk diskrit, baik koordinat maupun intensitas cahayanya. Kita dapat menganggap suatu citra digital sebagai suatu matriks dimana indeks baris dan kolomnya menyatakan koordinat sebuah titik pada citra tersebut dan masing-masing elemennya menyatakan intensitas cahaya pada titik tersebut. Suatu titik pada citra digital sering disebut sebagai elemen citra, elemen gambar ataupun *pixel.*

Untuk mengubah suatu citra kontinu ke dalam suatu representasi numeric dilakukan dengan proses digitalisasi oleh suatu *digitizer*, misalnya *scanner*, sehingga citra ini dapat diproses oleh sebuah komputer.

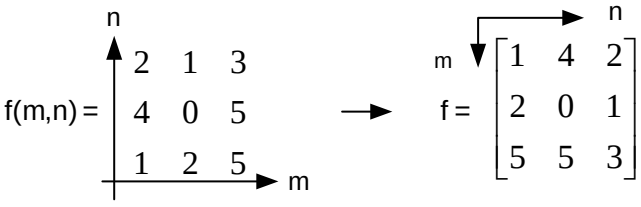
Digitalisasi sebuah citra dilakukan baik terhadap ruang (koordinasi (x,y)), maupun terhadap skala keabuannya (f(x,y)). Proses digitalisasi koordinat (x,y)

dikenal sebagai “pencuplikan citra” *(image sampling),* sedangkan proses digitalisasi keabuan (f(x,y) dikenal sebagai “kuantisasi derajat keabuan” *(grey-level quantization).*

Sebuah citra kontinu f(x,y) akan didekati oleh cuplikan-cuplikan yang seragam jaraknya dalam bentuk matriks MxN, dimana M adalah baris dan N adalah kolom. Nilai elemen-elemen matriks menyatakan derajat keabuan citra, sedangkan posisi elemen tersebut (dalam baris dan kolom) menyatakan koordinat titik-titik (x,y) dari citra. Bentuk matriks di bawah ini sebagai suatu citra digital :

$$F(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(0,1) & f(1,1) & \dots & f(1,N-1) \\ . & . & \dots & . \\ . & . & \dots & . \\ . & . & \dots & . \\ f(M-1,0) & . & \dots & f(M-1,N-1) \end{bmatrix}$$

Matriks di atas dapat disajikan dalam bentuk 2 dimensi dalam system koordinat *Cartesius* dengan memutar posisi matriks di atas sejauh 90^0 derajat searah jarum jam.



Sedangkan derajat keabuan [0,L] dibagi kedalam G selang dengan panjang selang yang sama, yaitu : $\mathbf{G=2^m}$ dimana m adalah kedalaman *bit* dan m bilangan bulat positif, bila hal ini diterapkan pada penyimpanan maka sebuah citra digital membutuhkan sejumlah b *bit*, dengan :

b = M x N x m .....(3.1)

Dalam suatu proses pencuplikan dan kuantisasi sering terjadi permasalahan, yaitu jumlah cuplikan dan derajat keabuan yang diperlukan untuk memperoleh suatu citra yang “baik”, makin tinggi nilai MxN dan m, maka citra kontinu f(x,y) akan makin didekati oleh citra digital yang dihasilkan. Tapi hal ini seringakali dbatasi oleh kemampuan hardware dari suatu komputer.

Agar dapat diolah dengan komputer digital, maka suatu citra harus direpresentasikan secara numeric dengan nilai-nilai diskrit. Representasi citra dari fungsi malar (kontinu) menjadi nilai-nilai diskrit disebut *digitalisasi*. Citra yang dihasilkan inilah yang disebut citra digital *(digital image)*. Pada umumnya citra digital berbentuk empat persegi panjang, dan dimensi ukurannya dinyatakan sebagai tinggi x lebar (atau lebar x panjang).

Citra digital yang tingginya *N*, dan lebarnya *M* dan memiliki *L* derajat keabuannya dapat dianggap sebagai fungsi:

$$f(x,y)\ \begin{cases} 0 \le x \le M \\ 0 \le y \le N \\ 0 \le f \le L \end{cases}$$

Citra digital yang berukuran *NxM* lazim dinyatakan dengan matriks yang berukuran (*N* = baris dan *M* = kolom) sebagai berikut :

$$\mathrm{f(x,y)}= \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,M) \\ f(0,1) & f(1,1) & \cdots & f(1,M) \\ \vdots & \vdots & \vdots & \vdots \\ f(N-1,0) & f(N-1,1) & \cdots & f(N-1,M-1) \end{bmatrix}$$

Indeks baris *(i)* dan indeks kolom *(j)* menyatakan suatu koordinat titik pada citra, sedangkan *f(i,j)* merupakan intensitas (derajat keabuan) pada titik *(i,j).*

Masing-masing elemen pada citra digital (berarti elemen matriks) disebut *image element, picture element, pixel* atau *pel*. Jadi, citra yang berukuran *N x M* mempunyai *NM* buah *pixel*. Sebagai contoh misalnya sebuah matriks berukuran 256 x 256 *pixel* dan direpresentisikan secara numeric dengan matriks yang terdiri dari 256 buah baris (di indeks dari 0 sampai 255) dan 256 buah kolom (di indeks dari 0 sampai 255) seperti contoh berikut :

$$\begin{bmatrix} 0 & 134 & 145 & \cdots & \cdots & 231 \\ 0 & 167 & 201 & \cdots & \cdots & 197 \\ 220 & 187 & 189 & \cdots & \cdots & 120 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 221 & 219 & 210 & \cdots & \cdots & 156 \end{bmatrix}$$

Pixel pertama pada koordinat (0,0) mempunyai nilai intensitas 0 yang berarti warna *pixel* tersebut hitam, *pixel* kedua pada koordinta (0,1) mempunyai intensitas134 yang berarti warnanya antara hitam dan putih, dan seterusnya.

3.2.2 Analisis File Teks

Teks merupakan sekumpulan karakter, huruf-huruf, angka-angka (A-Z, a-z, 0-9) dan symbol-simbol lainnya seperti %, &, ^, =, @, £, $, !, * dan lain-lain dengan menggunakan kode ASCII setiap karakter dari teks berjumlah 8 bit atau 1 *byte.*

3.2.3 Analisis Kriptografi

Pada algoritma RSA terdapat 3 langkah utama yaitu *key generation* (pembangkitan kunci), enkripsi, dan dekripsi. Kunci pada RSA mencakup dua buah kunci, yaitu *public key* dan *private key*. *Public key* digunakan untuk melakukan enkripsi, dan dapat diketahui oleh orang lain. Sedangkan *private key* tetap dirahasiakan dan digunakan untuk melakukan dekripsi. Pembangkitan kunci atau *key generation* dari RSA adalah sebagai berikut :

1. Pilih dua buah bilangan prima sembarang *a* dan *b*. Jaga kerahasiaan *a* dan *b* ini.
2. Hitung $n = a * b$. Besaran *n* ini tidak perlu dirahasiakan.
3. Hitung $m = (a-1) * (b-1)$. Sekali *m* telah dihitung, *a* dan *b* dapat dihapus untuk mencegah diketahuinya oleh pihak lain.
4. Pilih sebuah bilangan bulat untuk kunci publik, sebut namanya *e*, yang relatif prima terhadap *m* (relatif prima berarti GCD(e, m) = 1) dengan syarat $e \neq (p-1)$, $e \neq (q-1)$, dan $e < n$
5. Hitung kunci dekripsi, *d*, dengan kekongruenan $ed \equiv 1 \pmod{m}$.

Proses enkripsi dapat dilakukan dengan : $C_i = P_i^e \bmod n$

Sedangkan proses dekripsi dilakukan dengan cara : $C_i = P_i^e \bmod n$

Contoh penerapan algoritma RSA.

1. Menentukan bilangan acak *a* dan *b*

   a = 13

   b = 5

2. Hitung *n* dan *m*

n = 13 * 5 = 65

m = 12 * 4 = 48

3. Cari nilai *e*

GCD(e, 48) = 1

Misalnya,

e = 2 maka GCD(2, 48) = 2

e = 3 maka GCD(3, 48) = 3

e = 4 maka GCD(4, 48) = 4

e = 5 maka GCD(5, 48) = 1 , jadi digunakan e = 5

4. Lalu cari nilai *d*

$$d = \frac{1 + km}{e}$$

Misalnya

k = 1 maka d = 9,8

k = 2 maka d = 19,4

k = 3 maka d = 29 , jadi digunakan d = 29

Kita coba mengenkripsi pesan menggunakan angka-angka yang telah didapatkan. Misalkan pesan yang dikirim adalah angka 48.

E = 48 ^ 5 mod 65

= 254803968 mod 65

= 3

Setelah dilakukan enkripsi, didapat nilai sekarang adalah 3. Kemudian akan kita coba lakukan dekripsi pada nilai tersebut.

D = 3 ^ 29 mod 65

= 68630377364883 mod 65

= 48

Perhatikan bahwa nilai yang didapat sama dengan nilai awal, yaitu 48. Kekuatan algoritma RSA terletak pada tingkat kesulitan dalam memfaktorkan bilangan menjadi faktor primanya, dalam hal ini memfaktorkan *n* menjadi *a* dan *b*. Karena sekali *n* berhasil difaktorkan, maka menghitung nilai *m* adalah perkara mudah. Selanjutnya, walau nilai *e* diumumkan, perhitungan kunci *d* tidaklah mudah pula karena nilai *m* yang tidak diketahui.

3.2.4 Analisis Steganografi

Steganografi berasal dari bahasa Yunani yaitu Steganós yang berarti menyembunyikan dan Graptos yang artinya tulisan sehingga secara keseluruhan artinya adalah tulisan yang disebunyikan. Secara umum steganografi merupakan seni atau ilmu yang digunakan untuk menyembunyikan pesan rahasia dengan segala cara sehingga selain orang yang dituju, orang lain tidak akan menyadari keberadaan dari pesan rahasia tersebut.

Teknik steganografi sudah dikenal sejak jaman Yunani dan Romawi kuno. Misalnya dengan mencukur kepala budak, lalu pesan rahasia ditulis pada kulit kepalanya. Setelah rambut budak tersebut tumbuh, budak terebut dikirim untuk menyampaikan pesan rahasia tersebut. Untuk format atau media yang berbeda, metode yang digunakan untuk menyembunyikan pesannya juga berbeda-beda. Yang akan dibahas di tugas akhir ini hanya dua metode yaitu :

1. Least Significant Bit (LSB)

Dasar dari metode ini adalah pengetahuan akan bilangan biner atau bilangan basis 2, yang hanya terdiri dari ’1’ dan ’0’. Kedua bilangan yang menjadi dasar dari kerja komputer ini sering disebut dengan istilah *bit*. Susunan dari beberapa *bit* akan membentuk suatu informasi. Istilah yang umum dikenal adalah *byte*, yaitu kumpulan delapan *bit* data.

Dalam satu *byte* data, *bit* yang paling berpengaruh terhadap informasi yang dikandungnya biasanya adalah *bit* paling awal/paling kiri. *Bit* inilah yang dinamakan *Most Significant Bit (MSB)*. Semakin ke kanan, *bit-bit* tersebut semakin kecil pengaruhnya terhadap keutuhan data yang dikandung. *Bit* paling akhir/paling kanan inilah yang dinamakan *Least Significant Bit (LSB)*. Contohnya pada file *image* pesan dapat disembunyikan dengan menggunakan cara menyisipkannya pada *bit* rendah atau *bit* yang paling kanan (lsb) pada data *pixel* (titik) yang menyusun file tersebut. Seperti kita ketahui untuk file bitmap 24 *bit* maka setiap *pixel* (titik) pada gambar tersebut terdiri dari susunan tiga warna merah, hijau dan biru (RGB) yang masing-masing disusun oleh bilangan 8 *bit* (*byte*) dari 0 sampai 255 atau dengan format biner 00000000 sampai 11111111. Dengan demikian pada setiap *pixel* file bitmap 24 *bit* kita dapat menyisipkan 3 *bit* data.

Contohnya huruf A dapat kita sisipkan dalam 3 pixel, misalnya data raster original adalah sebagai berikut :

(00100111 11101001 11001000)

red blue green

(00100111 11001000 11101001)

red green blue

(11001000 00100111 11101001)

green red blue

Sedangkan representasi biner huruf A adalah 100000111. Dengan menyisipkan-nya pada data pixel diatas maka akan dihasilkan :

(00100111 **1110100<u>0</u>** 11001000)

red green green

(**0010011<u>0</u>** 11001000 **1110100<u>0</u>**)

white green green

(**1100100<u>1</u>** 00100111 11101001)

blue red blue

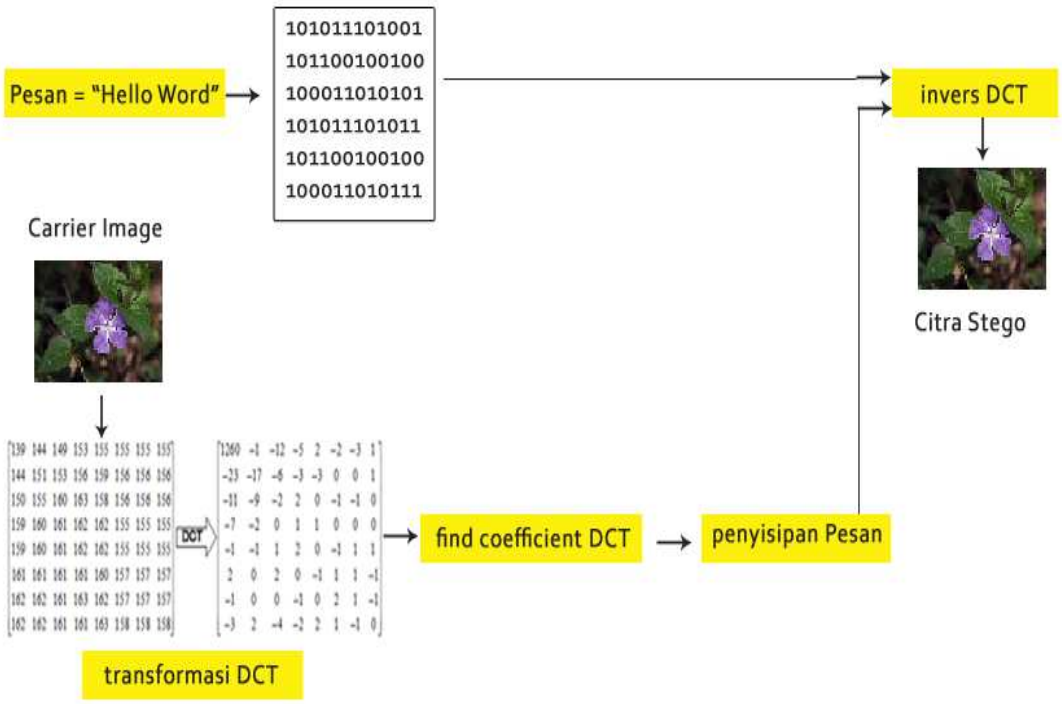
Terlihat hanya empat *bit* rendah yang berubah, untuk mata manusia maka tidak akan tampak perubahannya. Secara rata-rata dengan metoda ini hanya setengah dari data *bit* rendah yang berubah, sehingga bila dibutuhkan dapat digunakan *bit* rendah kedua bahkan ketiga.

2. Discrete Cosine Transformation (DCT)

Fungsi DCT yaitu suatu teknik yang digunakan untuk melakukan konversi sinyal kedalam komponen frekuensi pembentuknya dengan memperhitungkan nilai real dari hasil transformasinya.

DCT akan mentransformasi 8x8 blok nilai menjadi 64 nilai yang disebut koefisien DCT. Proses kebalikan dari DCT disebut IDCT (*Invers* DCT) yang akan mengubah data dari domain frekuensi ke domain spatial. Cara penggunaan metode

DCT pada steganografi dengan media penampung berupa citra digital seperti berikut:



Gambar 3.1 Proses Penyisipan Pesan *DCT*

Pada proses penyisipan pesan terdapat tugas-tugas seperti berikut :

1. *Load image*/citra yang akan disisipkan pesan. Format citra yang digunakan adalah .bmp / .jpg / .png /.gif sedangkan untuk ukuran citranya tidak ada batasannya, semakin besar ukuran citranya maka semakin banyak pula pesan yang dapat disembunyikan.
2. Lakukan Transformasi DCT dengan persamaan sebagai berikut :

$$F(u,v) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \left(\frac{2}{M}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} f(i,j) \cos\left(\frac{\pi.u}{2N}.(2i+1)\right) \cos\left(\frac{\pi.v}{2M}.(2j+1)\right) \quad .....(3.2)$$

Keterangan :

a) Ukuran image M x N

b) f(u,v) : nilai koefisien DCT pada matrik 8x8 kolom ke-u baris ke-v

c) F(i,j) : nilai data yang hendak ditransformasikan pada matrik 8x8 kolom ke-i baris ke-j

Berikut adalah contoh matriks 8x8 pixel yang dikonversi menjadi matriks 8x8 koefisien DCT :

$$\begin{bmatrix} 139 & 144 & 149 & 153 & 155 & 155 & 155 & 155 \\ 144 & 151 & 153 & 156 & 159 & 156 & 156 & 156 \\ 150 & 155 & 160 & 163 & 158 & 156 & 156 & 156 \\ 159 & 160 & 161 & 162 & 162 & 155 & 155 & 155 \\ 159 & 160 & 161 & 162 & 162 & 155 & 155 & 155 \\ 161 & 161 & 161 & 161 & 160 & 157 & 157 & 157 \\ 162 & 162 & 161 & 163 & 162 & 157 & 157 & 157 \\ 162 & 162 & 161 & 161 & 163 & 158 & 158 & 158 \end{bmatrix} \xrightarrow{DCT} \begin{bmatrix} 1260 & -1 & -12 & -5 & 2 & -2 & -3 & 1 \\ -23 & -17 & -6 & -3 & -3 & 0 & 0 & 1 \\ -11 & -9 & -2 & 2 & 0 & -1 & -1 & 0 \\ -7 & -2 & 0 & 1 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 2 & 0 & -1 & 1 & 1 \\ 2 & 0 & 2 & 0 & -1 & 1 & 1 & -1 \\ -1 & 0 & 0 & -1 & 0 & 2 & 1 & -1 \\ -3 & 2 & -4 & -2 & 2 & 1 & -1 & 0 \end{bmatrix}$$

d)

e)

f)

Gambar 3.2 Transformasi *DCT*

3. Dapatkan *coefficient DCT,* koefisien DCT pertama dari tiap blok memiliki nilai paling besar dibanding 63 koefisien lainnya karena merupakan nilai rata-rata dari keseluruhan blok, koefisien pertama ini disebut dengan koefisien DC dan 63 koefisien lainnya disebut dengan koefisien AC.
4. Gantikan bit pada citra tersebut dengan bit pesan yang akan dimasukan agar tidak terlihat (dapat digunakan LSB *insertion).*
5. Lakukan transformasi *invers dct.* Jika blok *DCT* tersbut dikembalikan melalui tahap *IDCT* (*invers DCT*), maka akan didapat kembali 64 nilai pixel yang mengalami sedikit perubahan sehingga tahap ini disebut tahap

lossy (terdapat informasi yang hilang pada saat sampai pada mata manusia). Berikut rumus *IDCT* :

$$f(i,j) = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \left(\frac{2}{N}\right)^{\frac{1}{2}} \left(\frac{2}{M}\right)^{\frac{1}{2}} f(u,v) \cos\left(\frac{\pi(2i+1)u}{2N}\right) \cos\left(\frac{\pi(2j+1)v}{2M}\right) \text{.....(3.3)}$$

6. Setelah selesai melakukan semua langkah diatas maka simpan citra tersebut sebagai citra stego (citra tersisip pesan).

Sebuah algoritma sederhana untuk menyimpan pesan di dalam gambar akan tampak seperti berikut ini:

Input: *message, cover image*

Output: *steganographic image containing message*

while *data left to embed* ***do***

get next DCT coefficient from left image

if *DCT ≠ 0 and DCT ≠ 1* ***then***

get next LSB from message

replace DCT LSB with message bit

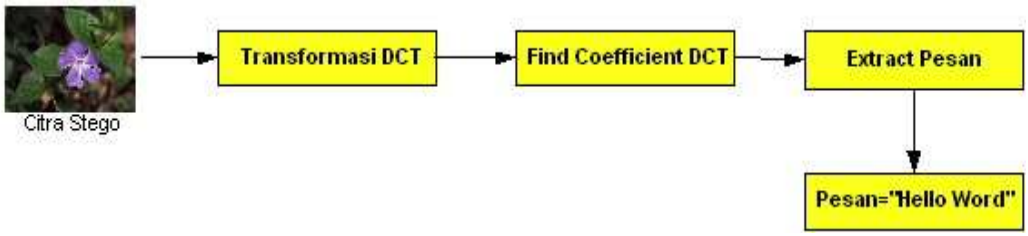
end if

insert DCT into steganographic image

end while

Bila dilihat dari langkah-langkah diatas untuk melakukan steganografi atau penyembunyian pesan dengan menggunakan metoda *Discrete Cosine Transformation (DCT)* adalah lakukan transformasi *DCT* terhadap citra yang akan

dijadikan media penampung tersebut. Transformasi *DCT* mengubah sinyal dari domain spasial ke domain frekuensi. *DCT* akan mentransformasikan 8x8 blok nilai slice F(0,0)…F(7,7) menjadi 64 nilai yang disebut koefisien *DCT* f(0,0)…f(7,7). Setelah mendapatkan koefisien *DCT* maka lakukan penyisipan pesan dengan menggunakan teknik *LSB insertion.* Setelah dilakukan penyisipan blok *DCT* tersbut lalu dilakukan proses *invers DCT* untuk mengembalikan data dari domain frekuensi ke domain spasial. Setelah selesai melakukan semua proses maka output yang dihasilkan adalah citra yang telah disisipi pesan.



Gambar 3.3 Proses Pembacaan Pesan *DCT*

Pada proses pembacaan pesan tugas-tugas yang perlu dilakukan diantaranya :

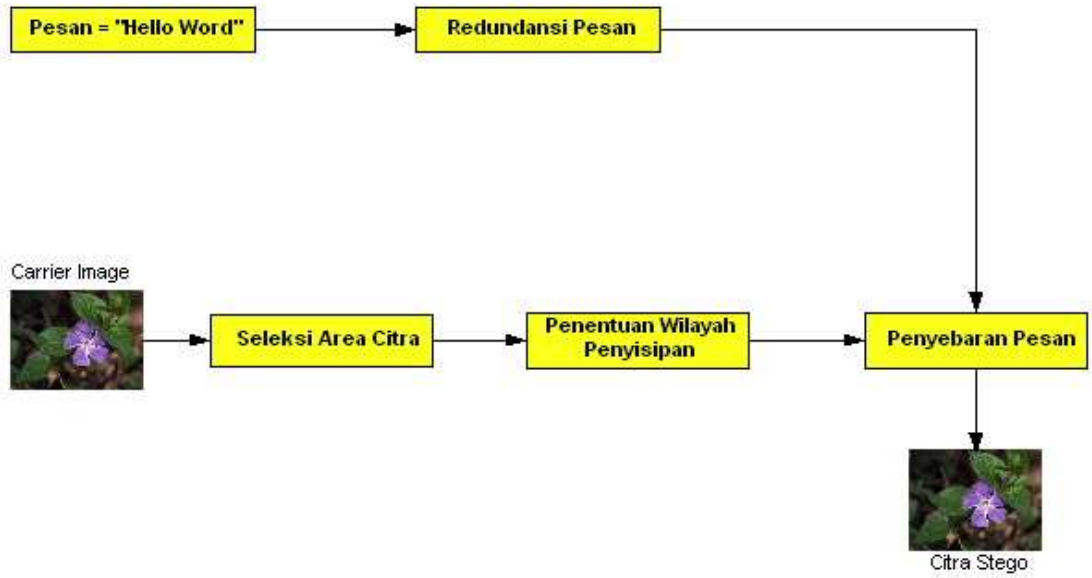
1. *Load image*/citra yang telah disisipi pesan untuk di *extract* dan dibaca pesan yang tersimpan didalamnya.
2. Lakukan transformasi dct dari citra tersisip pesan. Caranya sama dengan transformasi dct pada penyisipan pesan diatas.
3. Dapatkan *coefficient DCT,* koefisien *DCT* pertama dari tiap blok memiliki nilai paling besar dibanding 63 koefisien lainnya karena merupakan nilai rata-rata dari keseluruhan blok, koefisien pertama ini disebut dengan koefisien DC dan 63 koefisien lainnya disebut dengan koefisien AC

4. *Extract* bit data dari koefisien *DCT* terebut.
5. Gabungkan bit-bit tersebut menjadi sebuah pesan aktual

Untuk meng-*extract* pesan yang telah disisipkan pada citra adalah lakukan transfomasi *DCT* lalu cari koefisien *DCT,* setelah mendapatkan koefisien *DCT extract* bit-bit tersebut dan gabungkan bit-bit tersbut menjadi sebuah pesan aktual.

3. Redundant Pattern Encoding

Penerapan steganografi dengan metode ini adalah dengan menggambarkan pesan kecil pada kebanyakan gambar.



Gambar 3.4 Proses Penyisipan Pesan *Redundant*

Algoritma dari *redundant pattern encoding* yaitu memasukkan redundansi (penggandaan) ke dalam informasi/pesan yang hendak disembunyikan dan kemudian menyebarkan pesan itu keseluruhan gambar. Sebuah generator yang bekerja secara *pseudorandom* (suatu algoritma untuk menghasilkan urutan angka

yang mendekati sifat nomor acak) digunakan untuk menyeleksi dua area dari gambar (*patch* A dan *patch* B). *Patch* adalah metoda yang menandai area gambar.

Keseluruhan pixel pada *patch* A akan ditinggikan tingkat cahayanya. Lain halnya dengan *patch* B, yang justru diturunkan tingkat cahayanya (digelapkan). Dengan kata lain, intensitas pada pixel di suatu patch dinaikkan dengan nilai yang konstan, sementara *patch* lainnya diturunkan dengan nilai konstan yang sama. Perubahan kontras pada bagian *patch* akan mengenkripsi tiap satu bit dan perubahannya biasanya sangat kecil dan halus.

Kerugian dari teknik ini adalah hanya satu bit yang dapat dimasukkan pesan rahasia. Suatu gambar dapat digunakan untuk dimasukkan lebih dari satu bit dengan cara membagi-bagi gambar tersebut ke sub-gambar dan menggunakan cara yang sama untuk memasukkan pesan ke dalamnya. Keuntungan untuk menggunakan teknik ini sendiri adalah pesan rahasianya akan disebarkan ke gambar tersebut secara keseluruhan. Sehingga, apabila ada sebuah *patch* yang dihancurkan, yang lain masih tetap bertahan. Namun, tetap saja hal ini bergantung kepada ukuran pesan itu sendiri. Ini disebabkan pesan tersebut hanya bisa diulang dan tersebar ke seluruh bagian gambar apabila ukurannya cukup dan kecil.

Cara ini digunakan tidak bergantung terhadap jenis gambarnya (transfer domain) dan terbukti sebagai cara yang cukup ampuh dan baik untuk memasukkan pesan.



Gambar 3.5 Proses Pembacaan Pesan *Redundant*

Untuk mengextract pesan yang telah disisipkan pada citra adalah *extract* bit-bit tersebut dan gabungkan bit-bit tersbut menjadi sebuah pesan aktual.

3.3 Analisis 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 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 XP SP 3.*
2. *.NET Framework* merupakan suatu komponen *windows* yang terintegrasi dan dibuat agar dapat menjalankan berbagai macam

aplikasi berbasis *.NET* termasuk pembangunan aplikasi *Web Service (XML) dan* yang penulis gunakan adalah .NET *framework* 3.5.

3. *Tools* / bahasa pemrograman, yang penulis gunakan adalah *Microsoft Visual C# 2008.*

3.3.2 Analisis Kebutuhan Perangkat Keras

Sedangkan untuk kebutuhan hardware diantaranya yaitu :

1. *Processor*
2. *Harddisk*
3. *RAM*
4. *CD ROM*
5. *Motherboad*
6. Seperangkat I/O *(mouse,keyboard,monitor*)

3.4 Analisis Fungsional

Tahapan yang diperlukan dalam pembuatan sebuah program adalah menganalisis sebuah sistem. Analisis yang digunakan penulis adalah analisis fungsional. Analisis ini menentukan proses yang melayani kebutuhan sesuai dengan pertimbangan yang ada. Dalam pemodelan sistem ini, penulis menggunakan *Unified Modeling Language (UML)* sebagai bahasa untuk melakukan spesifikasi, visualisasi, konstruksi dan dokumentasi dari komponen perangkat lunak.

3.4.1 *Unified Modeling Language (UML)*

UML *(Unified Modeling Language)* adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun sistem perangkat lunak. UML menggunakan notasi grafis untuk menyatakan suatu desain. Pemodelan dengan UML berarti menggambarkan yang ada dalam dunia nyata ke dalam bentuk yang dapat dipahami dengan menggunakan notasi standart UML.

UML merupakan suatu kumpulan teknik terbaik yang telah terbukti sukses dalam memodelkan sistem yang besar dan kompleks. UML tidak hanya digunakan dalam proses pemodelan perangkat lunak, namun hampir dalam semua bidang yang membutuhkan pemodelan.

Alasan mengapa UML digunakan adalah, pertama, *scalability* dimana objek lebih mudah dipakai untuk menggambarkan sistem yang besar dan komplek. Kedua, *dynamic modeling*, dapat dipakai untuk pemodelan sistem dinamis dan *real time.*

Pemodelan dengan UML terdiri dari 8 tipe diagram yang berbeda untuk memodelkan sistem perangkat lunak. Masing-masing diagram UML didesain untuk menunjukkan satu sisi dari bermacam-macam sudut pandang (perspektif) dan terdiri dari tingkat abstraksi yang berbeda. Namun yang akan digunakan penulis hanya empat tipe diagram yaitu *Use Case Diagram, Sequence Diagram, Class Diagram* dan *Activity Diagram*.

3.4.1.1 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 case* 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 sebagai berikut :

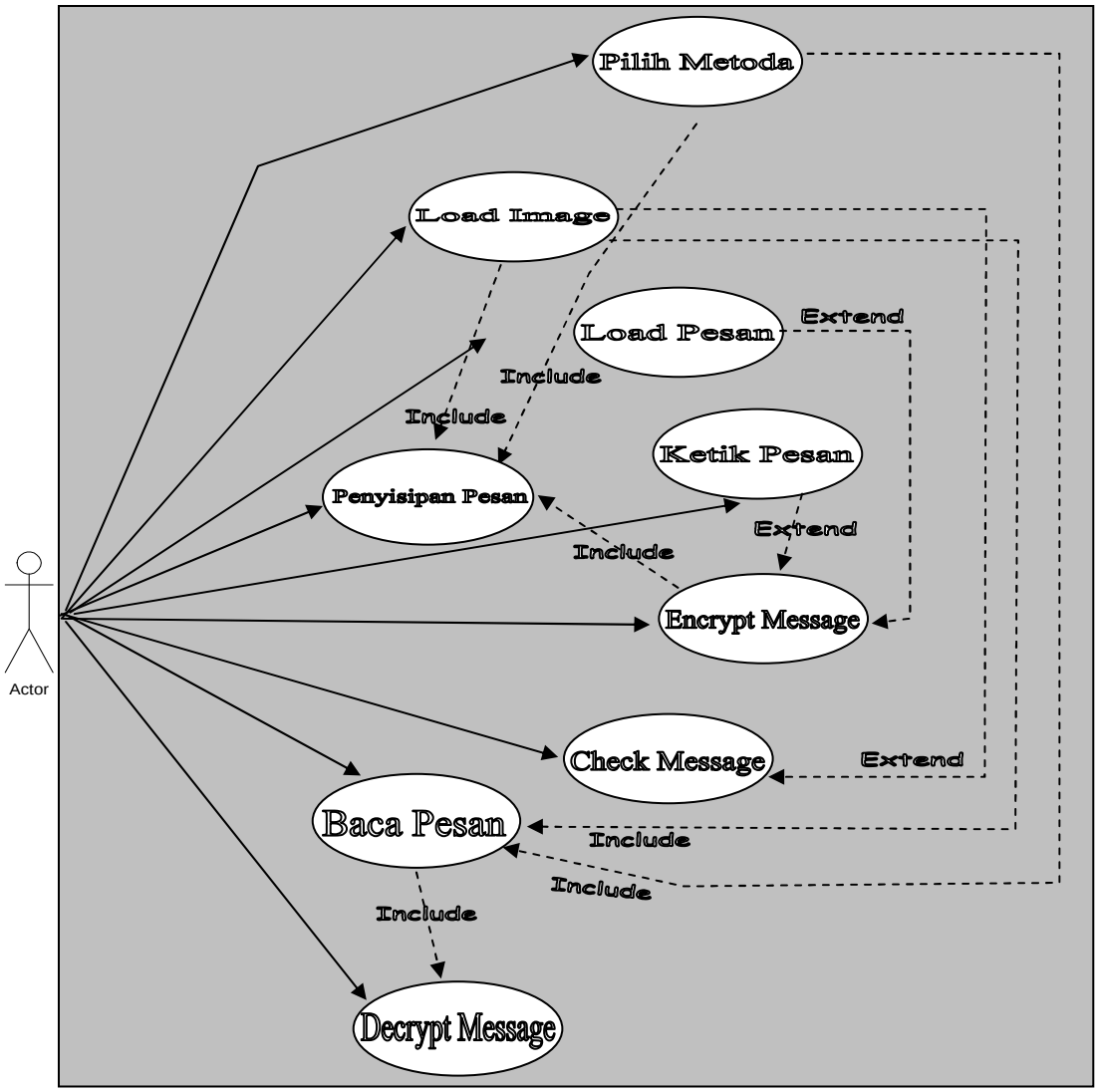


Diagram 3.6 *Use Case Diagram*

3.4.1.2 Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya.

3.4.1.3 Skenario Use Case

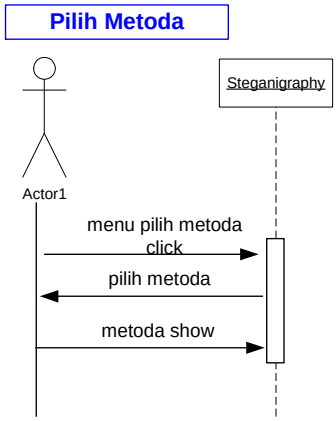
Skenario *use case* dan *sequence diagram* dari program aplikasi steganografi ini adalah sebagai berikut :

Tabel 3.1 *Skenario Use Case* Pilih Metoda

Nama	Pilih Metoda
Deskripsi	Pilih metoda merupakan langkah yang dilakukan oleh *user*. Untuk melakukan proses steganografi, *user* harus memilih salah satu metoda penyisipan pesan.
Aktor	User
Pre-condition	- Metoda steganografi belum ditentukan
Post-condition	- Metoda steganografi sudah dipilih

Aktor	Sistem
1. Mengklik menu *choose method* 3. Mengklik salah satu metoda steganografi	2. Sistem memunculkan metoda steganografi yang tersedia 4. System memunculkan *button-button* sesuai dengan metoda yang diplilih

1. *Sequence Diagram* Pilih Metoda



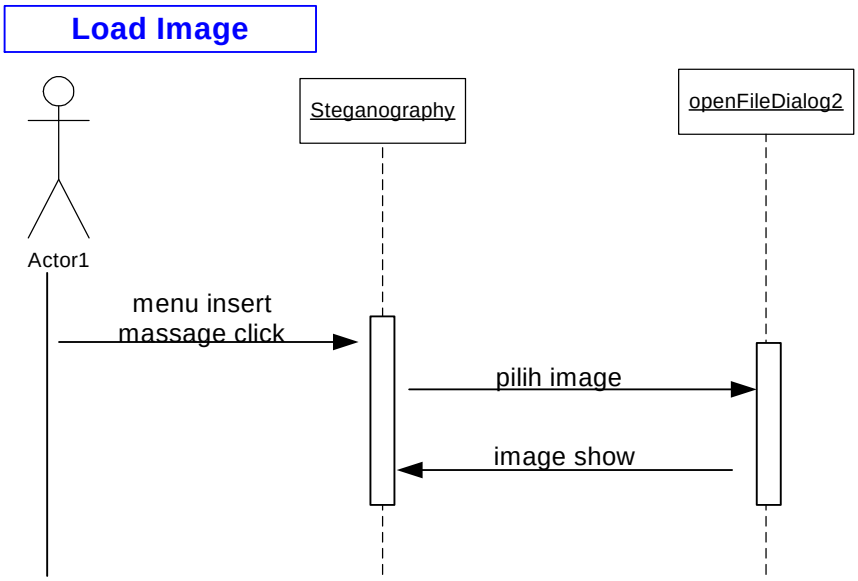
Gambar 3.7 *Sequence Diagram* Pilih Metoda

Tabel 3.2 Skenario Use Case Load Image

Nama	Load Image
Deskripsi	*Load image* merupakan proses yang dilakukan oleh *user*. Untuk melakukan steganografi *user* harus melakukan *load image* untuk dijadikan sebagai media penampung pesan.
Aktor	User
Pre-condition	- Citra tidak tersedia
Post-condition	- Tersedia citra untuk meanmpung pesan

Aktor	Sistem
1. Mengklik menu *insert Message* 3. Memilih citra untuk dijadikan media penampung pesan	2. Sistem memunculkan open file dialog untuk memilih citra digital yang memiliki format .bmp/.png/.jpg/.gif 4. Memunculkan citra yang dipilih

2. Sequence Diagram Load Image



Gambar 3.8 *Sequence Diagram Load Image*

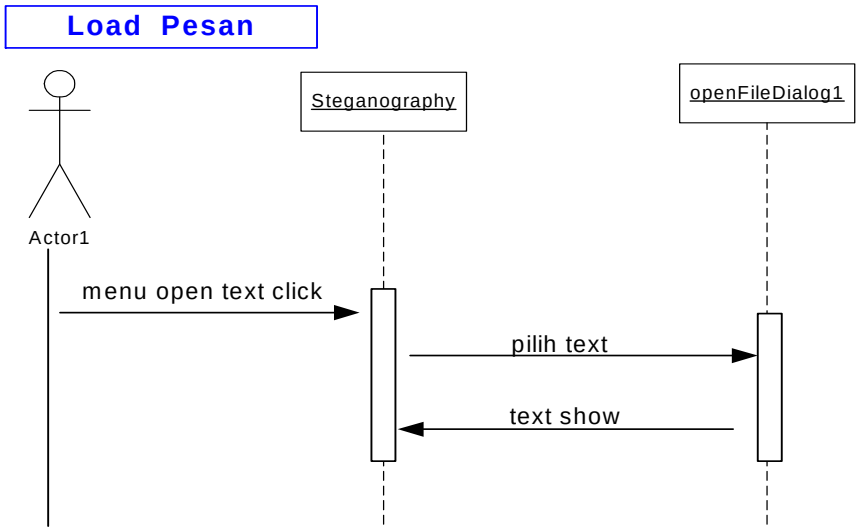
Tabel 3.3 *Skenario Use Case Load* Pesan

Nama	Load Pesan
Deskripsi	*Load* pesan merupakan proses yang dilakukan oleh *user*. Untuk melakukan steganografi *user* harus melakukan *load* pesan/text yang akan dimasukan atau disispkan kedalam media penampung yaitu berupa citra digital.
Aktor	User
Pre-condition	- Pesan tidak tersedia
Post-condition	- Pesan tersedia

Aktor	Sistem
1. Mengklik menu *open text* 3. Membuka file text	2. Sistem memunculkan open file dialog untuk memilih text yang memiliki

	format .txt 4. Memunculkan pesan yang dipilih

3. *Sequence Diagram Load* Pesan



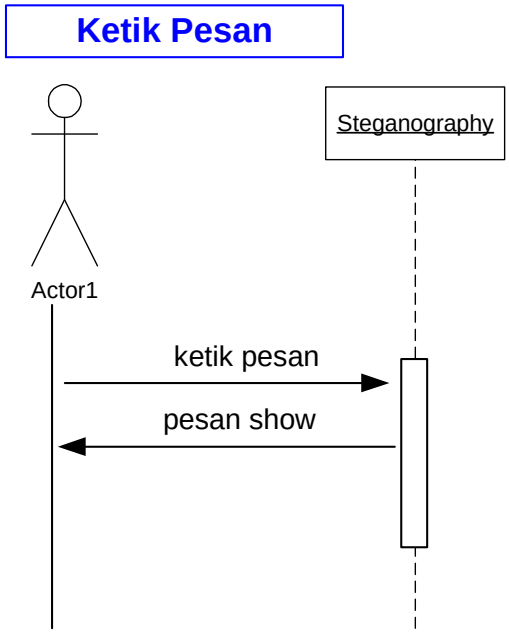
Gambar 3.9 *Sequence Diagram Load* Pesan

Tabel 3.4 *Skenario Use Case* Ketik Pesan

Nama	Ketik Pesan
Deskripsi	*Ketik pesan* merupakan proses yang dilakukan oleh *user*. Untuk melakukan steganografi *user* harus memasukan teks atau pesan yang akan disisipkan kedalam sebuah citra digital.
Aktor	User
Pre-condition	- Pesan tidak tersedia
Post-condition	- Pesan tersedia

Aktor	Sistem
1. Mengetik pesan di text box	2. Sistem memunculkan pesan yang dimasukan

4. *Sequence Diagram* Ketik Pesan



Gambar 3.10 *Sequence Diagram* Ketik Pesan

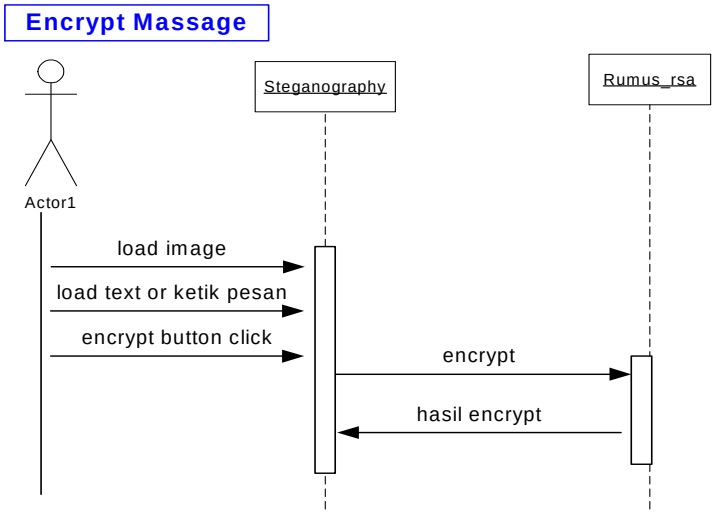
Tabel 3.*5 Skenario Use Case Encrypt Message*

Nama	Encrypt Message
Deskripsi	*Encrypt Message* merupakan proses yang dilakukan oleh *user*. Untuk meningkatkan *security* dalam melakukan steganografi pesan yang akan disispkan harus di *encrypt* terlebih dahulu.
Aktor	User
Pre-condition	- Pesan dapat terbaca atau belum terenkripsi
Post-condition	- Pesan tidak terbaca atau sudah terenkripsi

Aktor	Sistem
1. Memasukan pesan yang akan	2. Sistem memunculkan pesan

dienkripsi 3. Mengklik *button encrypt Message*	4. Sistem memunculkan pesan yang sudah terenkripsi

5. Sequence Diagram Encrypt Message



Gambar 3.11 *Sequence Diagram Encrypt Message*

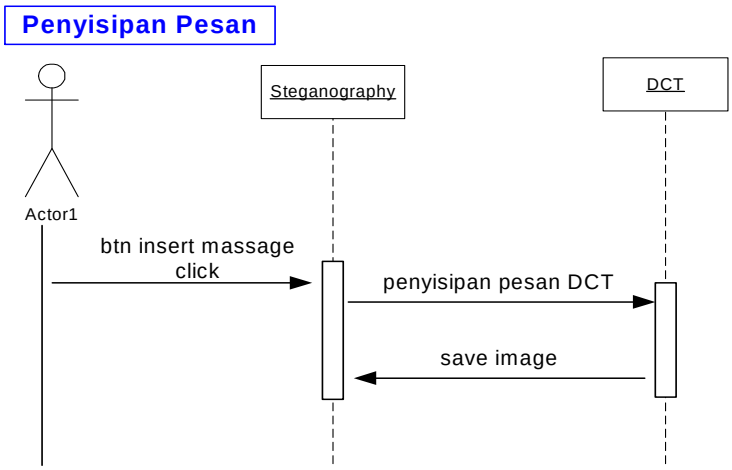
Tabel 3.6 *Skenario Use Case* Penyisipan Pesan

Nama	Penyisipan Pesan
Deskripsi	Penyisipan pesan merupakan proses yang dilakukan oleh user. User dapat menyisipkan pesan pada sebuah citra digital agar pesan tersebut tidak dapat terbaca oleh pihak lain yang tidak berhak mengetahuinya. Untuk lebih meningkatkan keamanan dari pesan yang disisipkan kedalam citra, pesan tersebut dienkripsi terlebih dahulu sebelum disisipkan kedalam citra
Aktor	User
Pre-condition	- Pesan dapat terbaca oleh semua pihak.

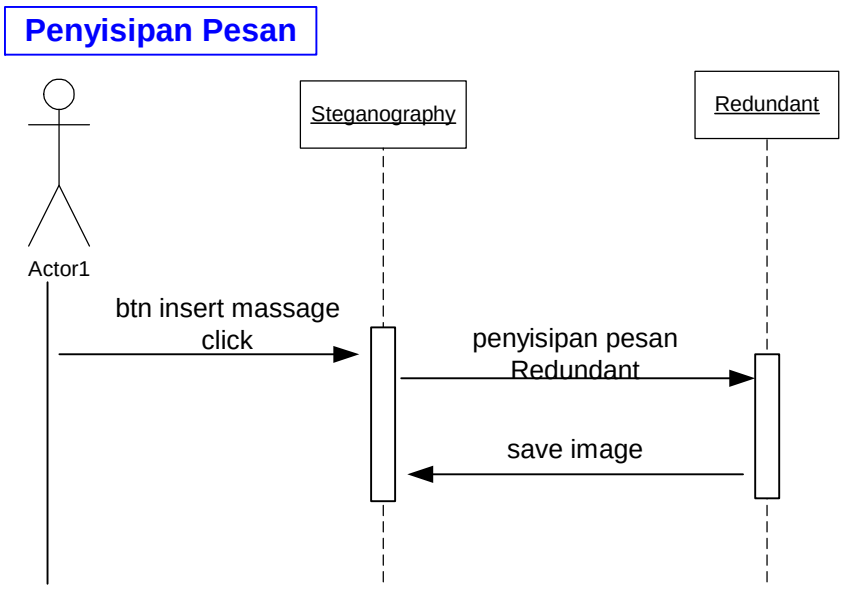
Post-condition	- Pesan sudah dalam bentuk terenkripsi - Pesan tersembunyi di dalam sebuah citra digital

Aktor	Sistem
1. Mengklik menu pilih metoda 3. Mengklik menu *insert Message* 5. Memilih citra untuk dijadikan media penampung pesan 7. Me-*load* atau menulis pesan yang akan dimasukan kedalm citra 9. Mengklik *button encrypt Message* 11. Mengklik tombol insert pesan 13. Menulis nama file dan mengklik *button save*	2. Sistem memunculkan *button-button* sesuai dengan metoda yang diplilih 4. Sistem memunculkan open file dialog citra digital yang memiliki format .bmp/.png/.jpg/.gif 6. Memunculkan citra yang akan disisipi pesan 8. Memunculkan pesan yang akan disisipkan 10. Mengubah pesan menjadi pesan yang ternkripsi 12. Menyisipkan pesan kedalm citra 14. Menyimpan citra stego

6. *Sequence Diagram* Penyisipan Pesan



Gambar 3.12 *Sequence Diagram* Penyisipan Pesan *DCT*



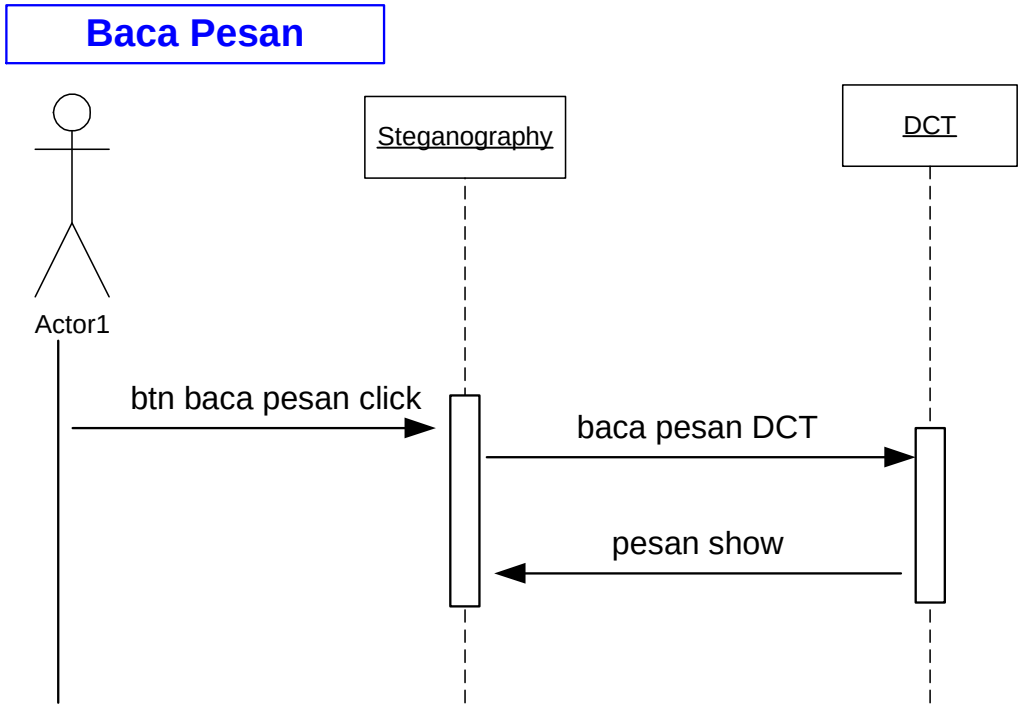
Gambar 3.13 *Sequence Diagram* Penyisipan Pesan *Redundant*

Tabel 3.7 *Skenario Use Case* Baca Pesan

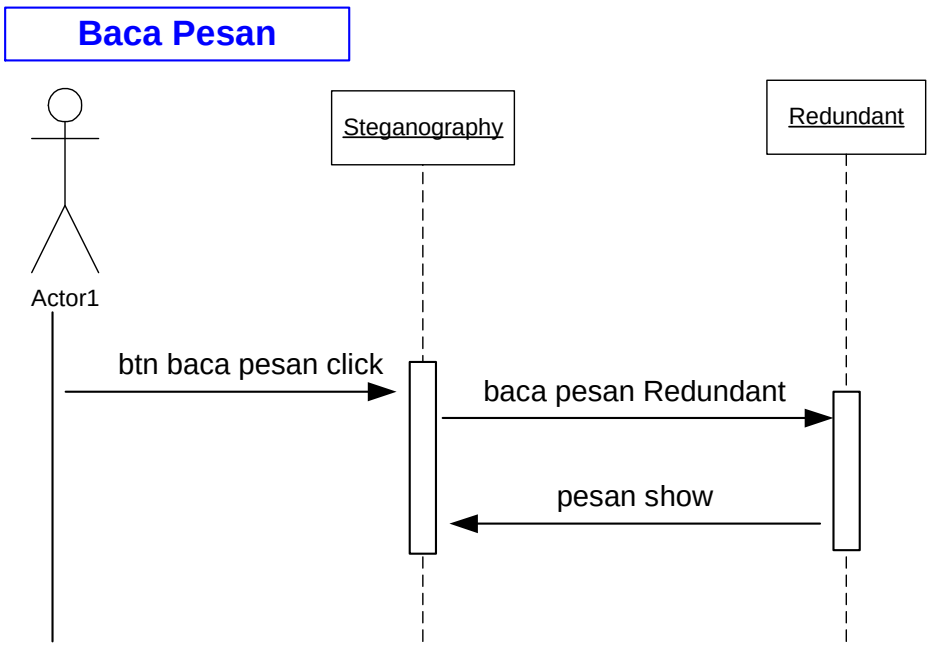
Nama	Baca Pesan
Deskripsi	Baca pesan merupakan proses yang dilakukan oleh *user*. *User* dapat meng-*extract* pesan yang telah disipkan kedalam citra agar pesan tersebut dapat dibaca oleh pihak yang berhak mengetahuinya. Sebelumnya pesan tersebut harus di dekripsi terlebih dahulu.
Aktor	User
Pre-condition	- Pesan dalam bentuk terenkripsi - Pesan tersembunyi dalam sebuah citra
Post-condition	- Pesan dapat terbaca
Aktor	Sistem

1. Mengklik menu *choose method* 3. Mengkilik menu *extract Message* 5. Memilih citra yang telah disisipi pesan 7. Mengklik tombol *extract Message* 9. Mengklik tombol dekripsi pesan	2. Sistem memunculkan *button-button* sesuai dengan metoda yang dipilih 4. Sistem memunculkan open file dialog untuk memilih citra digital yang telah disisipi pesan yang memiliki format .bmp/.jpg/.png/.gif 6. Memunculkan citra yang telah disisipi pesan 8. Memunculkan Pesan yang disembunyikan dalam sebuah citra dalam bentuk pesan masih terenkripsi 10. Mengubah pesan terenkripsi menjadi pesan yang dapat terbaca

7. *Sequence Diagram* Baca Pesan



Gambar 3.14 *Sequence Diagram* Baca Pesan *DCT*



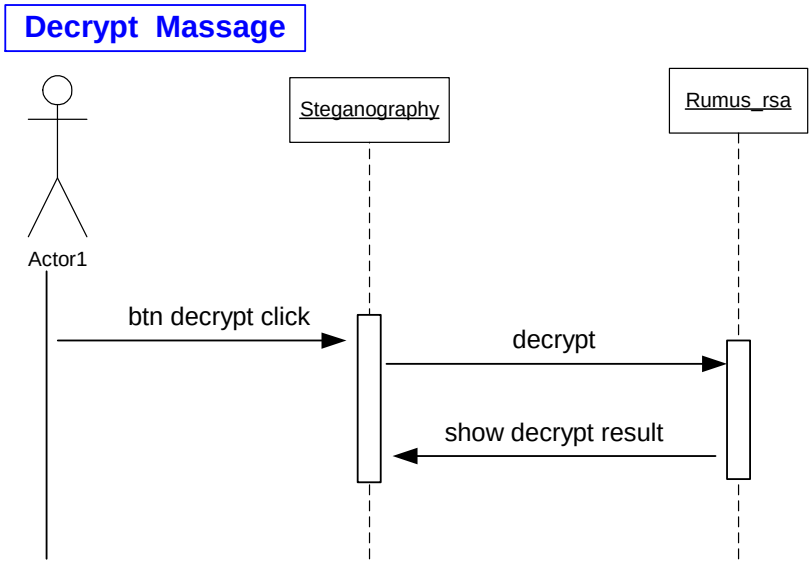
Gambar 3.15 *Sequence Diagram* Baca Pesan *Redundant*

Tabel 3.8 *Skenario Use Case Decrypt Message*

Nama	Decrypt Message
Deskripsi	*Decrypt Message* merupakan proses yang dilakukan oleh *user*. Untuk dapat membaca pesan yang telah di-*extract* dari citra digital *user* harus mendekripsikan pesan tersebut terlebih dahulu.
Aktor	User
Pre-condition	- Pesan tidak dapat terbaca atau dalam bentuk terenkripsi
Post-condition	- Pesan dapat dibaca

Aktor	Sistem
1. Klik *button decrypt Message*	2. Sistem mendekeripsi pesan

8. Sequence Diagram Decrypt Message



Gambar 3.16 *Sequence Diagram Decrypt Message*

3.4.1.4 Activity Diagram

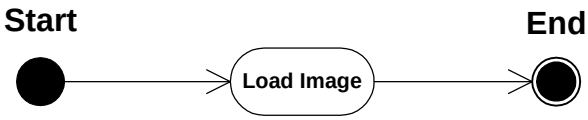
Activity diagram menggambar aliran fungsionalitas dalam suatu sistem. Dapat digunakan dalam analisa kebutuhan untuk menggambarkan aliran kejadian melalui suatu *use case* atau menggambarkan berbagai alir aktivitas dalam sistem yang dirancang , bagaimana masing-masing alir berawal, keputusan yang mungkin terjadi dan bagaimana seluruh alir berakhir. *Activity diagram* dari program aplikasi steganografi digambarkan seperti gambar di bawah ini :

1. *Activity Diagram* Pilih Metoda



Gambar 3.17 *Activity Diagram* Pilih Metoda

2. Activity Diagram Load Image



Gambar 3.18 *Activity Diagram Load Image*

3. *Activity Diagram Load* Pesan



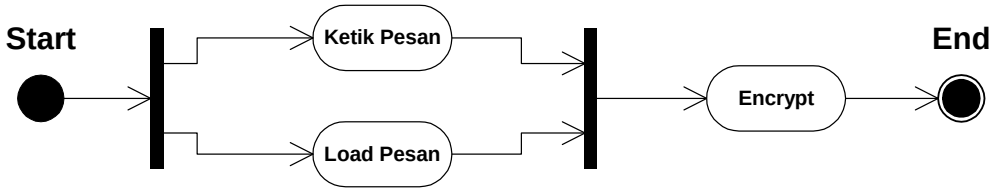
Gambar 3.19 *Activity Diagram Load* Pesan

4. *Activity Diagram* Ketik Pesan



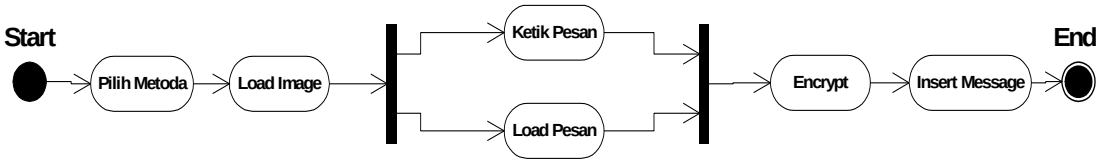
Gambar 3.20 *Activity Diagram* Ketik Pesan

5. Activity Diagram Encrypt Message



Gambar 3.21 *Activity Diagram Encrypt Message*

6. *Activity Diagram* Penyisipan Pesan



Gambar 3.22 *Activity Diagram* Penyisipan Pesan

7. *Activity Diagram* Baca Pesan



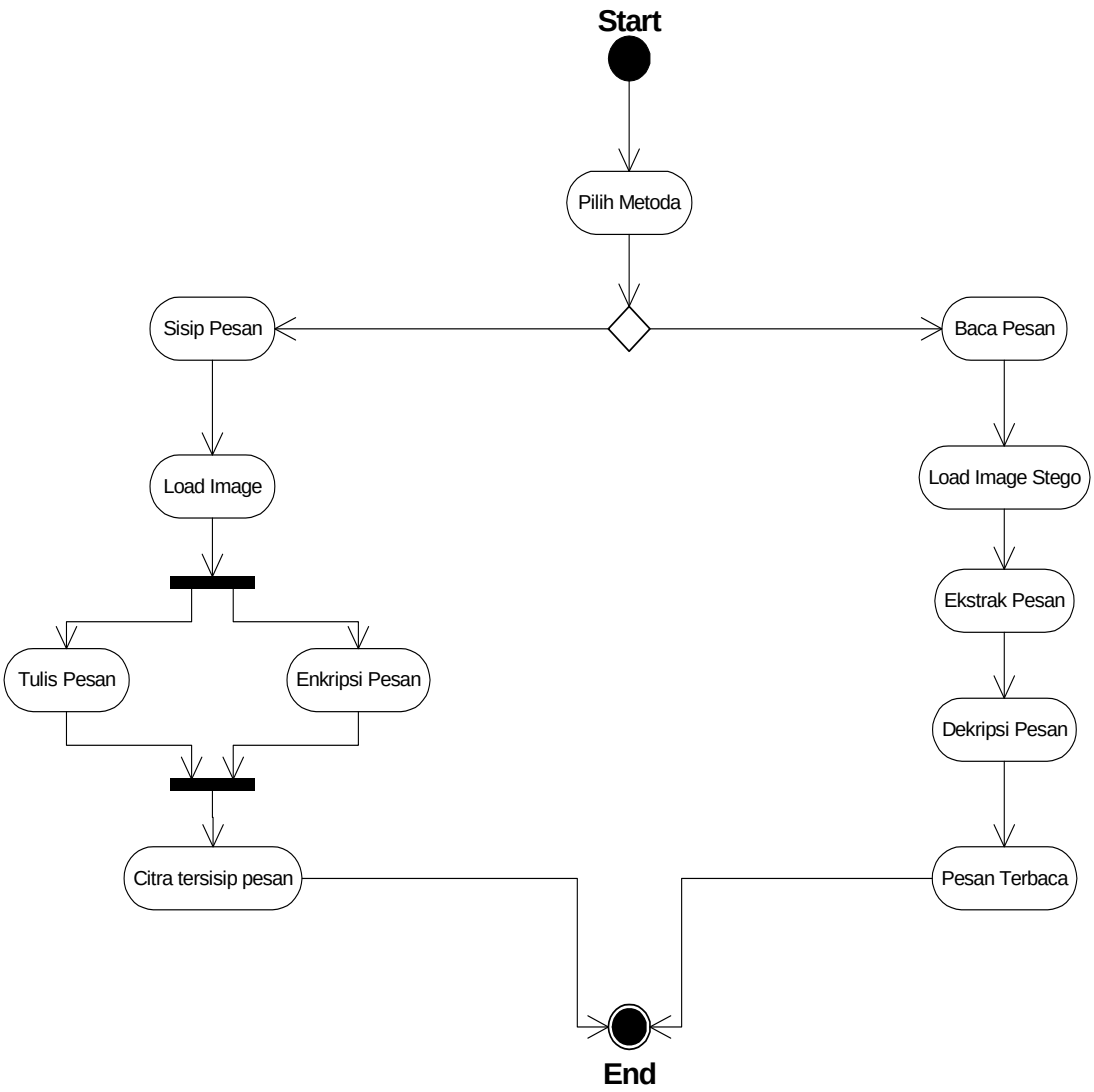
Gambar 3.23 *Activity Diagram* Baca Pesan

8. Activity Diagram Decrypt Message



Gambar 3.24 *Activity Diagram Decrypt Message*

9. *Activity Diagram* Aplikasi Steganografi

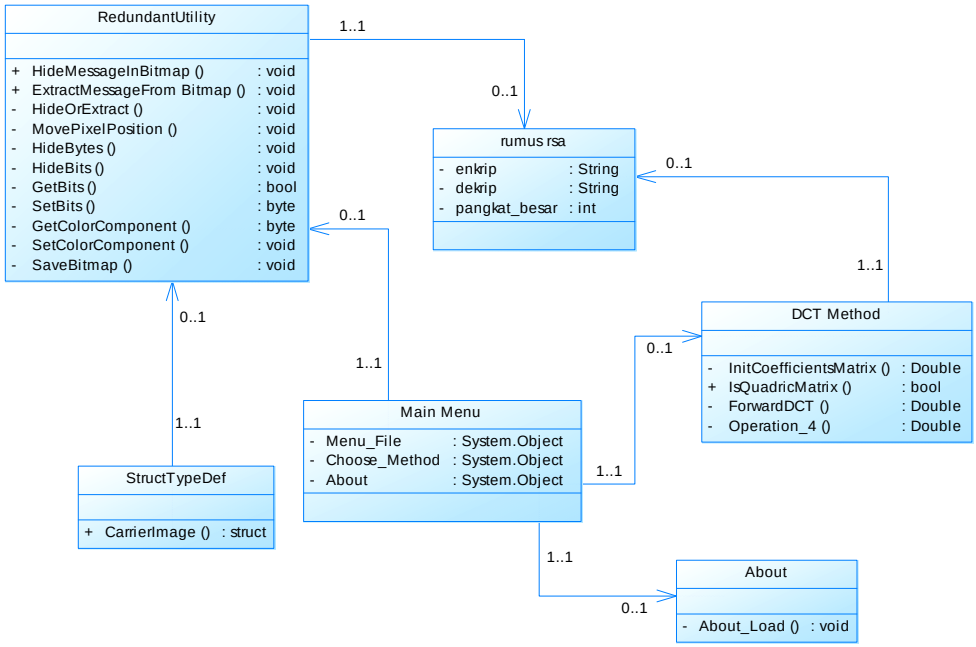


Gambar 3.25 *Activity Diagram* Aplikasi Steganografi

3.4.1.5 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

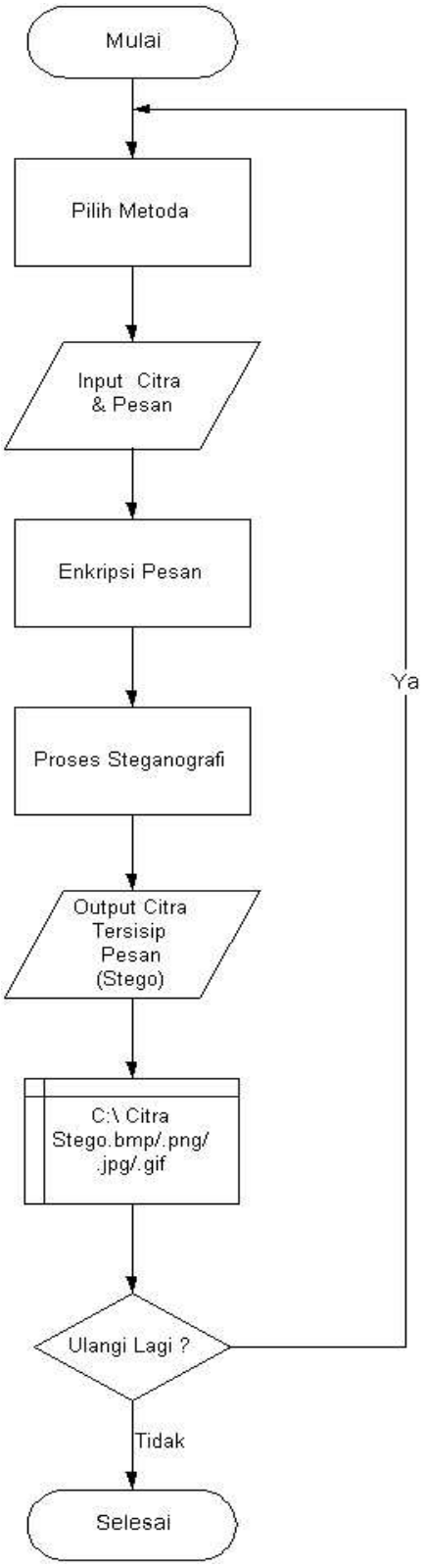
biasanya mempunyai beberapa *class diagram*. *Class diagram* sangat membantu dalam visualisasi struktur kelas dari suatu sistem. *Class diagram* dari program aplikasi steganografi digambarkan seperti gambar di bawah ini :



Gambar 3.26 *Class Diagram*

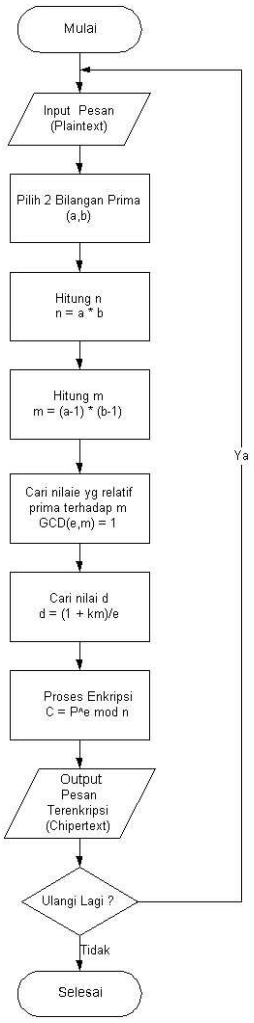
3.5 Perancangan Prosedural

Perancangan prosedural ini berisi tentang flowchart dari aplikasi steganografi ini. Diantaranya adalah flowchart kriptografi RSA dan flowchart steganografi metoda discrete cosine transformation dan redundant pattern encoding. Flowchart adalah salah satu bagan yang menggambarkan arus logika dari data yang akan diproses dalam suatu program dari awal sampai akhir. Flowchart dari program aplikasi steganografi dapat dilihat pada gambar dibawah :



Gambar 3.27 Flowchart Steganografi

3.5.1 Flowchart Kriptografi RSA



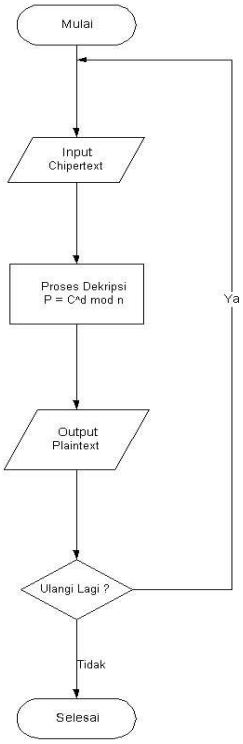
Gambar 3.28 Flowchart Enkripsi Pesan Teknik RSA

Pada flowchart di atas, terlihat data yang dibutuhkan untuk mengenkripsi pesan adalah dua buah bilangan prima sebagai *private key* dan *public key* dan juga file teks(*plaintext*) sebagai pesan yang akan dienkripsi.

Langkah pertama adalah masukan pesan yang akan dienkripsi lalu tentukan dua buah bilangan prima sebagai a dan b, bilangan prima ini harus dijaga kerahasiaannya.

Setelah mendapatkan dua buah bilangan prima maka hitung nilai n yang diperoleh dari perkalian kedua bilangan prima tersebut dan nilai n ini tidak perlu dirahasiakan.

Selanjutnya hitung nilai m dengan rumus m = (a-1) * (b-1), kalau nilai m sudah dihitung maka nilai a dan b dapat dihapus untuk mencegah diketahuinya oleh pihak lain. Pilih sebuah bilangan bulat untuk *public key*, sebut saja namanya e yang relatife prima terhadap m (relatife prima berarti GCD(e,m)=1) dengan syarat : $e \neq (p-1)$, $e \neq (q-1)$, dan $e < n$. Tahap terakhir sebelum melakukan enkripsi terhadap pesan yang dimasukan adalah menghitung nilai d sebagai kunci dekripsi dengan kekongruenan $ed \equiv 1 \pmod{m}$, dan prosen enkripsinya sendiri dapat dilakukan dengan menggunakan nilai atau angka yang sudah dicari sebelumnya dengan rumus $C = P^e \bmod n$, dimana C adalah *Chipertext* dan P adalah *Plaintext.*

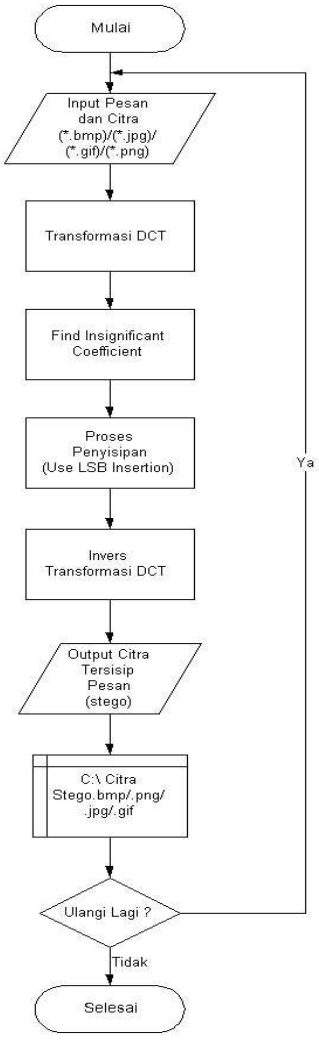


Gambar 3.29 Flowchart Dekripsi Pesan Teknik RSA

Dari flowchart di atas tahapan yang dilakukan untuk mendekripsi pesan yaitu, masukan pesan yang sudah dienkripsi lalu kita ambil *private key* nya yang sudah ditentukan pada saat pembangkitan kunci dengan *private key* adalah (d,n).

Setelah kita ketahui *private key* nya maka kita bias langsung mendekrip pesannya dengan rumus P = C^d mod n, dimana P adalah *Plaintext* dan C adalah *Chipertext.*

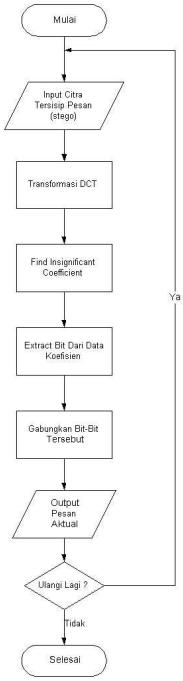
3.5.2 Flowchart DCT (Discrete Cosine Transformation)



Gambar 3.30 Flowchart *Insert* Pesan Metoda *DCT*

Bila dilihat dari flowchart diatas, terlihat data yang dibutuhkan untuk melakukan proses steganografi atau penyembunyian pesan adalah memasukan pesan yang akan disisipkan dan media penampungnya yang berupa citra.

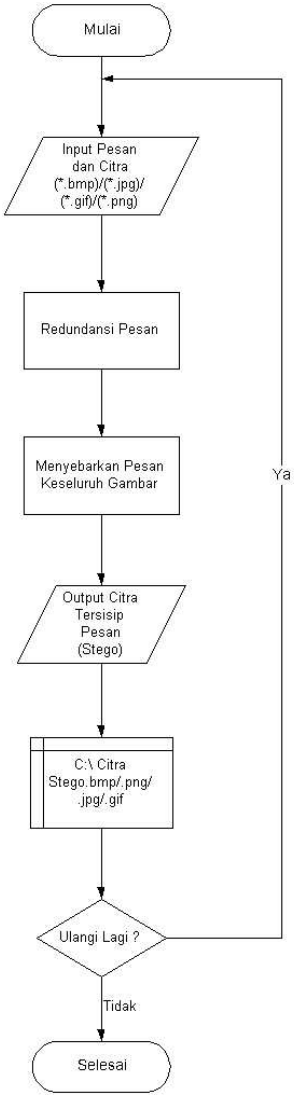
Setelah menuliskan pesan dan memasukan citra maka proses selanjutnya adalah melakukan transformasi DCT terhadap citra yang akan dijadikan media penampung tersebut. Transformasi DCT mengubah sinyal dari domain spasial ke domain frekuensi. DCT akan mentransformasikan 8x8 blok nilai slice F(0,0)…F(7,7) menjadi 64 nilai yang disebut koefisien DCT f(0,0)…f(7,7). Setelah mendapatkan koefisien DCT maka lakukan penyisipan pesan dengan menggunakan teknik *LSB insertion.* Setelah dilakukan penyisipan blok DCT tersbut lalu dilakukan proses *invers* DCT untuk mengembalikan data dari domain frekuensi ke domain spasial. Setelah selesai melakukan semua proses maka output yang dihasilkan adalah citra yang telah disisipi pesan.



Gambar 3.31 Flowchart *Extract* Pesan Metoda *DCT*

Untuk mengextract pesan yang telah disisipkan pada citra, langkah pertama yang harus dilakukan adalah memasukan citra stego atau citra tersisip pesan kedalam aplikasi steganografi. Proses selanjutnya adalah lakukan transfomasi DCT lalu cari koefisien DCT, setelah mendapatkan koefisien DCT *extract* bit-bit tersebut dan gabungkan bit-bit tersbut menjadi sebuah pesan aktual.

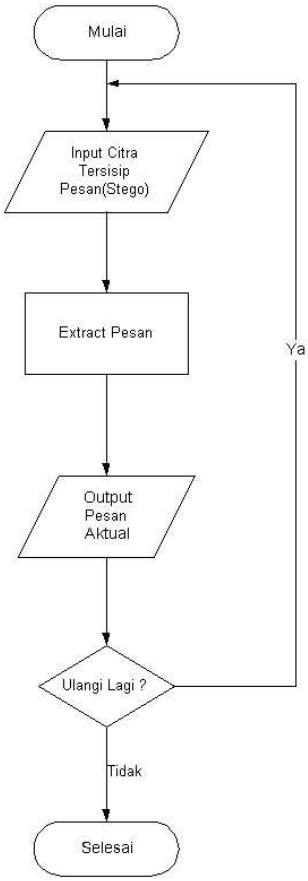
3.5.3 Flowchart Redundant Pattern Encoding



Gambar 3.32 Flowchart *Insert* Pesan *Redundant Pattern Encoding*

Bila kita lihat flowchart diatas langkah pertama untuk melakukan steganografi dengan menggunakan metoda *redundant pattern encoding* adalah menginputkan pesan dan sebuah media penampung. Media penampung yang digunakan dalam aplikasi steganografi ini adalah berupa sebuah citra digital.

Setelah menuliskan pesan dan memasukan citra maka proses selanjutnya adalah melakukan redundansi terhadap pesan yang akan disembunyikan dan kemudian menyebarkan pesan itu keseluruh gambar. Maka keluarlah sebuah citra stego atau citra yang telah disisipi pesan.



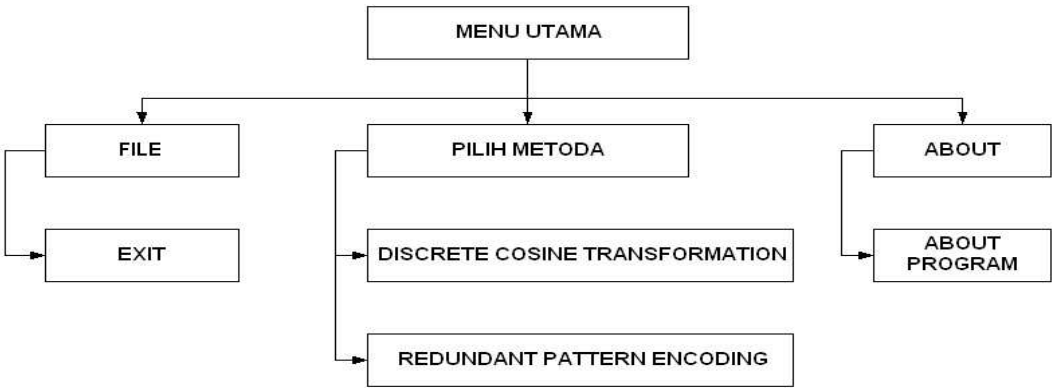
Gambar 3.33 Flowchart *Extract* Pesan *Redundant Pattern Encoding*

Untuk mengextract pesan yang telah disisipkan pada citra, langkah pertama yang harus dilakukan adalah memasukan citra stego atau citra tersisip

pesan kedalam aplikasi steganografi. Proses selanjutnya adalah *extract* bit-bit tersebut dan gabungkan bit-bit tersbut menjadi sebuah pesan aktual.

3.6 Perancangan Struktur Menu

Rancangan struktur menu merupakan prosedur-prosedur program dari sebuah program aplikasi. Menu-menu yang terdapat dalam program aplikasi steganografi ini, baik yang menggunakan metoda *Discrete Cosine Transformation (DCT)* maupun *Redundant Pattern Encoding* memiliki struktur menu yang sama. Untuk lebih jelasnya dapat dilihat dalam bentuk rancangan struktur menu pada gambar berikut ini :



Gambar 3.34 Rancangan Struktur Menu

1. *Exit*

   Exit adalah perintah untuk mengakhiri jalannya program aplikasi.

2. *Discrete Cosine Transformation*

   Menu *Discrite Cosine Transformation* adalah metoda yang akan dipakai untuk menyisipkan dan meng-*extract* pesan.

3. *Redundant Pattern Encoding*

   Menu *Redundant Pattern Encoding* adalah metoda yang akan dipakai untuk menyisipkan dan meng-*extract* pesan.

4. *About Program*

   Menu *about program* merupakan perintah untuk menampilkan *form about* yang berisi tentang nama, versi dan deskripsi program aplikasi.

3.7 Perancangan Antar Muka

Perancangan antar muka dari aplikasi steganografi ini dibuat berdasarkan komponen-komponen visual dasar. Komponen-komponen itu diantaranya :

1. *Form Main Menu*

   Form *main menu* merupakan sebuah bingkai/tempat untuk menyimpan *menu strip* yang berisi fungsionalitas dari program aplikasi steganografi ini.

2. *Form DCT*

   Form yang digunakan untuk menyisipkan dan meng-*extract* pesan dengan menggunakan metoda steganografi *Discrete Cosine Transformation (DCT).*

3. *Form Redundant*

   Form yang digunakan untuk menyisipkan dan meng-*extract* pesan dengan menggunakan metoda *Redundant Pattern Encoding*.

4. *Form About*

   Form ini merupakan form yang berisi informasi yang berkaitan dengan aplikasi steganografi ini. Isinya terdiri dari nama program, versi program, *copyright,* dll.

5. *Menu Strip*

   Komponen ini merupakan komponen yang membentuk sebuah menu pilihan yang disediakan dalam program aplikasi ini.

6. *Text Box*

   Komponen ini merupakan komponen tempat *input* atau *output* bertipe teks.

7. *Button*

   Button merupakan komponen yang dapat difungsikan sesuai dengan kebutuhan program atau untuk melakukan sebuah *event.*

8. *Label*

   Label merupakan komponen untuk menampilkan teks tapi *user* tidak bisa berinteraksi.

9. *Picture Box*

   Komponen ini merupakan komponen yang menampung citra yang dibuka atau di *load.*

3.7.1 Perancangan Form Main Menu

Pada program aplikasi steganografi ini, *form main menu* merupakan form yang pertama kali muncul pada saat program aplikasi steganografi ini dijalankan.

Form main menu ini berisi pilihan fungsionalitas yang dapat dilakukan dalam program aplikasi ini.

Tabel 3.9 Komponen MenuFile

No	Antarmuka	*Properti*	*Value*
1.	*fileToolStripMenuItem*	- *Text* - *Name* - *Collection*	- *&File* - *fileToolStripMenuItem* - *exitToolStripMenuItem*
2.	*exitToolStripMenuItem*	- *Text* - *Name*	- *&Exit* - *exitToolStripMenuItem*

Tabel 3.10 Komponen Label

No	Antarmuka	*Properti*	*Value*
1.	*label1*	- *Name* - *Text*	- *label1* - *Analisis Perbandingan Metoda*
2.	*label2*	- *Name* - *Text*	- *label2* - *Redundant Pattern Encoding dan Discrete Cosine Transformation*

			(DCT)
3.	*label3*	- *Name* - *Text*	- *label3* - *Sebagai Metoda Steganografi Pada Citra Digital*

Tabel 3.11 Komponen *Menu* Pilih Metoda

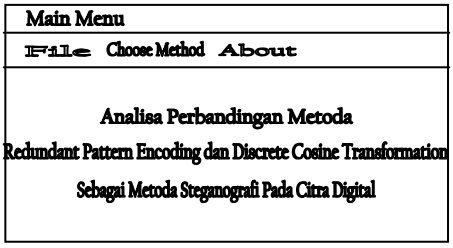
No	**Antarmuka**	***Properti***	***Value***
1.	pilihMetoda*ToolStripMenuItem*	- *Text* - *Name* - *Collection*	- &Pilih Metoda - pilihMetoda*ToolStripMenuItem* - *discreteCosineTransformation* - *redundantPatternEncoding*
2.	*discreteCosineTransformation*	- *Text* - *Name*	- *&Discrete Cosine Transformation* - *discreteCosineTransformation*
3.	*redundantPatternEncoding*	- *Text* - *Name*	- *&Redundant Pattern Encoding* - *redundantPattern*

			Encoding

Tabel 3.12 Komponen *MenuAbout*

No	Antarmuka	*Properti*	*Value*
1.	*aboutToolStripMenuItem*	- *Text* - *Name* - *Collection*	- *&About* - *aboutprogramTool StripMenuItem* - *abuotProgram*
2.	*aboutProgram*	- *Text* - *Name*	- *&About Program* - *aboutProgram*

Untuk Lebih jelasnya perancangan *form main menu* digambarkan seperti berikut ini :



Gambar 3.35 Perancangan *Form Main Menu*

3.7.2 Perancangan *Form DCT*

Pada program aplikasi steganografi ini, *form dct* merupakan *form* yang digunakan untuk menyisipkan dan meng-*extract* pesan dengan menggunakan metoda steganografi *Discrete Cosine Transformation*.

Tabel 3.13 Komponen *Menu Strip*

No	Antarmuka	*Properti*	*Value*
1.	*File*	- *Name* - *Text* - *Collection*	- *fileToolStripMenuItem* - *&File* - *Exit*
2.	*Exit*	- *Name* - *Text* - *Collection*	- *exitToolStripMenuItem* - *&Exit* -
3.	*Open*	- *Name* - *Text* - *Collection*	- *openToolStripMenuItem* - *&Open* - *Open Text*
4.	*Open Text*	- *Name* - *Text* - *Collection*	- *openTextToolStripMenuItem* - *&Open Text* -
5.	*Process*	- *Name* - *Text* - *Collection*	- *processToolStripMenuItem* - *&Process* - *Insert Message* *Extract Message*

Tabel 3.14 Komponen *SplitContainer*

No	Antarmuka	*Properti*	*Value*
1.	*splitContainer1*	- *Panel 1*	- *splitContainer1.Panel1*

		- *Panel 2* - *Name*	- *splitContainer1.Panel2* - *splitContainer1*

Tabel 3.15 Komponen *Text Box*

No	**Antarmuka**	***Properti***	***Value***
1.	*textBox1*	- *Name* - *BackColor* - *ScrollBar* - *MultiLine* - *Text* - *Size*	- *textBox1* - *White* - *Both* - *True* - - *225,200*
2.	*textBox2*	- *Name* - *BackColor* - *ScrollBar* - *MultiLine* - *Text* - *Size*	- *textBox2* - *White* - *Both* - *True* - - *225,200*
3.	*textBox3*	- *Name* - *BackColor* - *ScrollBar* - *MultiLine* - *Text*	- *textBox3* - *White* - *No* - *False* -

		- *Size*	- *120,20*

Tabel 3.16 Komponen *Button*

No	Antarmuka	*Properti*	*Value*
1.	*button1*	- *Name* - *Text* - *Size*	- *btnencryptmsg* - *Encrypt Message* - 155,40
2.	*button2*	- *Name* - *Text* - *Size*	- *btnhidemsg* - *Hide Message* - 95,40
3.	*button3*	- *Name* - *Text* - *Size*	- *saveImage* - *Save* - 70,40
4.	*button4*	- *Name* - *Text* - *Size*	- *btnextractmsg* - *Extract Message* - 155,40
5.	*button5*	- *Name* - *Text* - *Size*	- *btndecryptmsg* - *Decrypt Message* - 155,40
6.	*button6*	- *Name* - *Text* - *Size*	- *psnr* - Hitung PSNR - 90,40

Tabel 3.17 Komponen *Label*

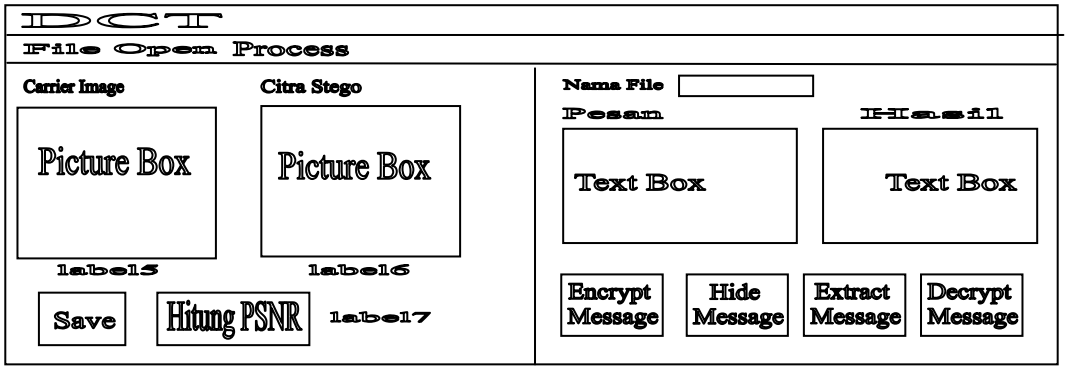
No	Antarmuka	*Properti*	*Value*
1.	*label1*	- *Name* - *Text*	- *label1* - Pesan
2.	*label2*	- *Name* - *Text*	- *label2* - Hasil
3.	*label3*	- *Name* - *Text*	- *label3* - Nama File
4.	*label4*	- *Name* - *Text*	- *label4* - *Carrier Image*
5.	*label5*	- *Name* - *Text*	- *label5* - *label5*
6.	*label6*	- *Name* - *Text*	- *label6* - *label6*
7.	*label7*	- *Name* - *Text*	- *label7* - *label7*
8.	*label8*	- *Name* - *Text*	- *label8* - *Citra Stego*

Tabel 3.18 Komponen Picture Box

No	Antarmuka	*Properti*	*Value*
1.	*pictureBox1*	- *Name*	- *pictureBox1*

2.	*pictureBox2*	- *Name*	- *pictureBox2*

Untuk lebih jelasnya perancangan *form dct* digambarkan seperti berikut ini :



Gambar 3.36 Perancangan *Form DCT*

3.7.3 Perancangan *Form Redundant Pattern Encoding*

Pada program aplikasi steganografi ini, *form redundant* merupakan *form* yang digunakan untuk menyisipkan dan meng-*extract* pesan dengan menggunakan metoda steganografi *Redundant Pattern Encoding.*

Tabel 3.19 Komponen *Menu Strip*

No	Antarmuka	*Properti*	*Value*
1.	*File*	- *Name* - *Text* - *Collection*	- *fileToolStripMenuItem* - *&File* - *Exit*
2.	*Exit*	- *Name* - *Text*	- *exitToolStripMenuItem* - *&Exit*

		- *Collection*	-
3.	*Open*	- *Name* - *Text* - *Collection*	- *openToolStripMenuItem* - *&Open* - *Open Text*
4.	*Open Text*	- *Name* - *Text* - *Collection*	- *openTextToolStripMenuItem* - *&Open Text* -
5.	*Process*	- *Name* - *Text* - *Collection*	- *processToolStripMenuItem* - *&Process* - *Insert Message* *Extract Message*

Tabel 3.20 Komponen *SplitContainer*

No	Antarmuka	*Properti*	*Value*
1.	*splitContainer1*	- *Panel 1* - *Panel 2* - *Name*	- *splitContainer1.Panel1* - *splitContainer1.Panel2* - *splitContainer1*

Tabel 3.21 Komponen *Text Box*

No	Antarmuka	*Properti*	*Value*
1.	*textBox1*	- *Name* - *BackColor*	- *textBox1* - *White*

		- *ScrollBar* - *MultiLine* - *Text* - *Size*	- *Both* - *True* - - *225,200*
2.	*textBox2*	- *Name* - *BackColor* - *ScrollBar* - *MultiLine* - *Text* - *Size*	- *textBox2* - *White* - *Both* - *True* - - *225,200*
3.	*textBox3*	- *Name* - *BackColor* - *ScrollBar* - *MultiLine* - *Text* - *Size*	- *txtfilename* - *White* - *No* - *False* - - *120,20*

Tabel 3.22 Komponen *Button*

No	**Antarmuka**	***Properti***	***Value***
1.	*button1*	- *Name* - *Text* - *Size*	- *btnencryptmsg* - *Encrypt Message* - *155,40*

2.	*button2*	- *Name* - *Text* - *Size*	- *btnhidemsg* - *Hide Message* - *115,40*
3.	*button3*	- *Name* - *Text* - *Size*	- *btnextractmsg* - *Extract Message* - *75,50*
4.	*button4*	- *Name* - *Text* - *Size*	- *btndecryptmsg* - *Decrypt Message* - *115,40*
5.	*button5*	- *Name* - *Text* - *Size*	- *saveImage* - *Save* - *70,40*
6.	*button6*	- *Name* - *Text* - *Size*	- *psnr* - Hitung PSNR - *95,40*

Tabel 3.23 Komponen *Label*

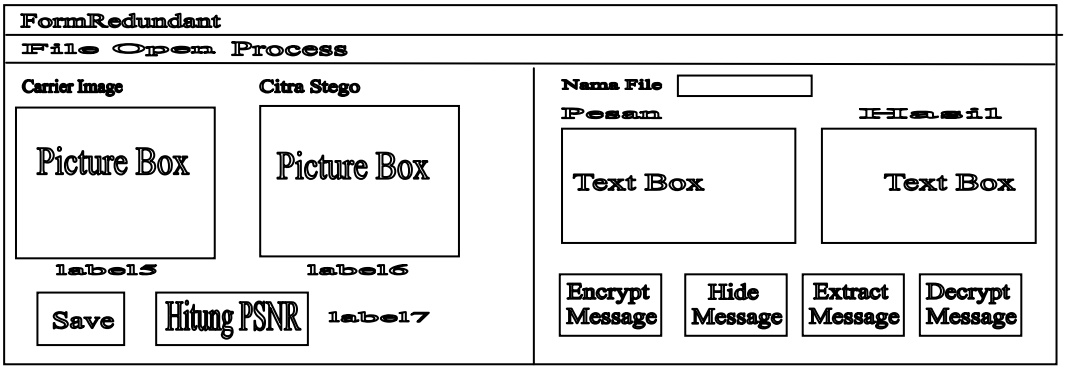
No	**Antarmuka**	***Properti***	***Value***
1.	*label1*	- *Name* - *Text*	- *label1* - *Pesan*
2.	*label2*	- *Name*	- *label2*

		- *Text*	- *Result*
3.	*label3*	- *Name* - *Text*	- *label3* - *label3*
4.	*label4*	- *Name* - *Text*	- *label4* - *Carrier Image*
5.	*label5*	- *Name* - *Text*	- *label5* - *label5*
6.	*label6*	- *Name* - *Text*	- *label6* - *label6*
7.	*label7*	- *Name* - *Text*	- *label7* - *Citra Stego*
8.	*label8*	- *Name* - *Text*	- *label8* - *label8*

Tabel 3.24 Komponen *Picture Box*

No	**Antarmuka**	***Properti***	***Value***
1.	*pictureBox1*	- *Name*	- *pictureBox1*
2.	*pictureBox2*	- *Name*	- *pictureBox2*

Untuk lebih jelasnya perancangan *form redundant* digambarkan seperti berikut ini :



Gambar 3.37 Rancangan Tampilan *Form Redundant*

3.7.4 Perancangan *Form About*

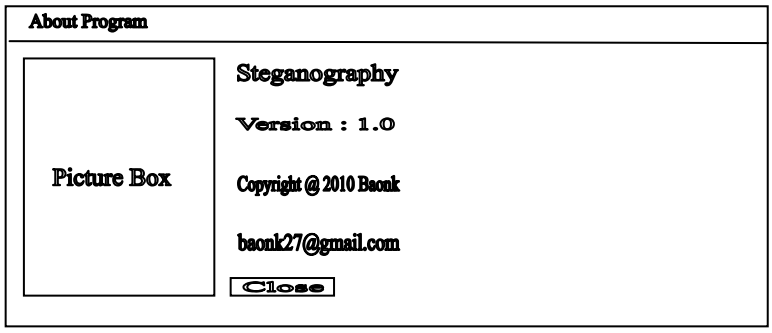
Form about merupakan *form* yang berisi mengenai informasi program aplikasi steganografi ini diantaranya adalah nama program, versi program, deskripsi dan lain-lain. Berikut ini adalah rancangan tampilan dari *form about :*

Tabel 3.25 Komponen *Form About*

No	Antarmuka	*Properti*	*Value*
1.	*pictureBox1*	- *Name* - *Image* - *SizeMode* - *Size*	- *pictureBox1* - *01.jpg* - *StretchImage* - *158;236*
2.	*label1*	- *Name* - *Text* - *Text Align*	- *label1* - *Product Name* - *TopLeft*
3.	*Label2*	- *Name* - *Text* - *Text Align*	- *label2* - *Version* - *TopLeft*

4.	*Label3*	- *Name* - *Text* - *Text Align*	- *label3* - *Copyright* - *TopLeft*
5.	*Label4*	- *Name* - *Text* - *Text Align*	- *label4* - *Company Name* - *TopLeft*
6.	*button1*	- *Name* - *Text* - *Size*	- *label2* - *Close* - *75;23*

Untuk lebih jelasnya perancangan *form about* digambarkan seperti berikut ini :



Gambar 3.38 Rancangan Tampilan *Form About*