BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Analisis sistem merupakan suatu tahapan yang bertujuan untuk mengetahui dan mengamati apa saja yang terlibat dalam suatu *system*. Pembahasan yang ada pada analisis sistem ini yaitu analisis masalah,analisis proses kerja sistem, analisis metode, analisis algoritma, analisis MMS, analisis *manager key*, analisis kebutuhan nonfungsional, analisis spesifikasi dan kebutuhan perangkat lunak, analisis kebutuhan fungsional dan perancangan arsitektur.

3.1.1 Analisis Masalah

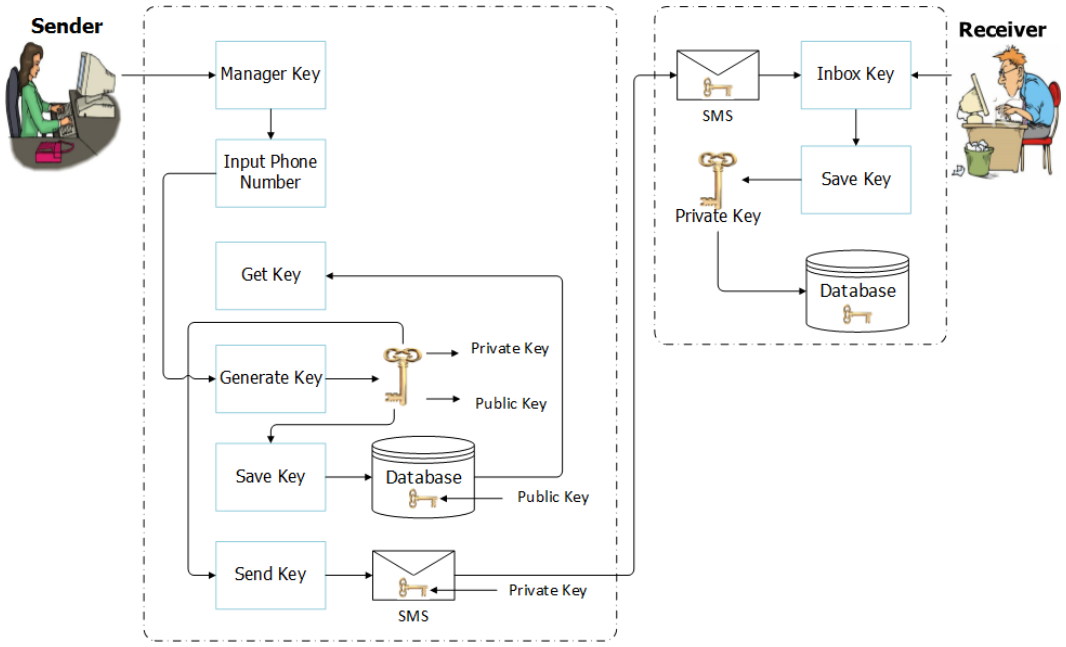
Analisis masalah dilakukan untuk mengetahui masalah-masalah apa saja yang terjadi dalam pembangunan aplikasi *steganography*. Masalah yang terjadi pada pembangunan aplikasi *steganography* pada citra digital berbasis *android* ini yaitu timbulnya dampak negatif pada berkembangnya teknologi informasi, salah satunya pada fasilitasnya *Multimedia Messaging Service*(MMS). Pada proses pengiriman dan penerimaan pesan MMS sangatlah rentan yang memungkinkan terjadinya seperti pencurian, penyadapan, pembajakan dan banyak hal lain. Karena hal tersebut, penerapan *steganography* sangat dibutuhkan dalam menjaga kerahasiaan suatu pesan.

Bedasarakan uraian analisis masalah yang telah dipaparkan, diperlukan aplikasi untuk mengamankan informasi pada citra digital. Oleh karena itu penulis tertarik untuk membuat tugas akhir ini dengan judul "Aplikasi *Steganography* Citra Digital Pada Layanan MMS Berbasis *Android*".

3.1.2 Analisis Proses Kerja *System*

Analisis *system* digunakan untuk mengetahui alur proses sistem yang digunakan untuk dapat diterapkan ke dalam aplikasi yang dibangun. Pembangunan aplikasi ini menggunakan algoritma *RSA* (*RivestShamirAdleman*) sebagai keamanan data dan untuk teknik *embedding* pesannya menggunakan metode *Haar Wevelet* yang disertai proses kirim kunci antara *sender* dan *receiver*.

Tahapan-tahapan *sender* dan *receiver* dalam mengirim dan menerima *key* dapat digambarkan lebih rinci melalui gambar 3.1.



Gambar 3.1 Proses Pada *Sender* dan *Receiver Key*

Tahapan-tahapan yang dilakukan antara *sender* dan *receiver* yaitu sebagai berikut:

1. ***Sender Key***
   a. *Sender* masuk ke menu *manager key*, yang bertujuan untuk membangkitkan sepasang kunci *(Private key* dan *Public key)*.
   b. Sender memasukan *phone number receiver* yang akan digunakan untuk menyimpan pasangan kunci pada *phone number sender(public key)* kedalam database aplikasi.
   c. Sender bisa mengambil kunci yang sudah ada di dalam database aplikasi dengan cara menekan *button get key*, dengan syarat *sender* sudah pernah menyimpan pasangan kunci untuk *phone number* yang dipilih.

d. *Sender* melakukan *generate key,* tujuan *generate key* ialah untuk membangkitkan kunci algoritma, dimana Algoritma yang digunakan yaitu Algoritma *RSA*.

e. *Sender* menyimpan pasangan kunci kedalam *database,* dimana kunci yang di dapat merupakan hasil dari pembangkitan kunci dengan Algoritma *RSA*. Dimana kunci yang akan disimpan disebut juga *public key*.

f. *Sender* mengirimkan pasangan kunci dari *public key* yaitu *private key* ke *receiver,* dimana *privet key* ini akan digunakan untuk proses *decrypt* untuk *receiver*. Proses *sender key* ini dikirim melaui SMS.

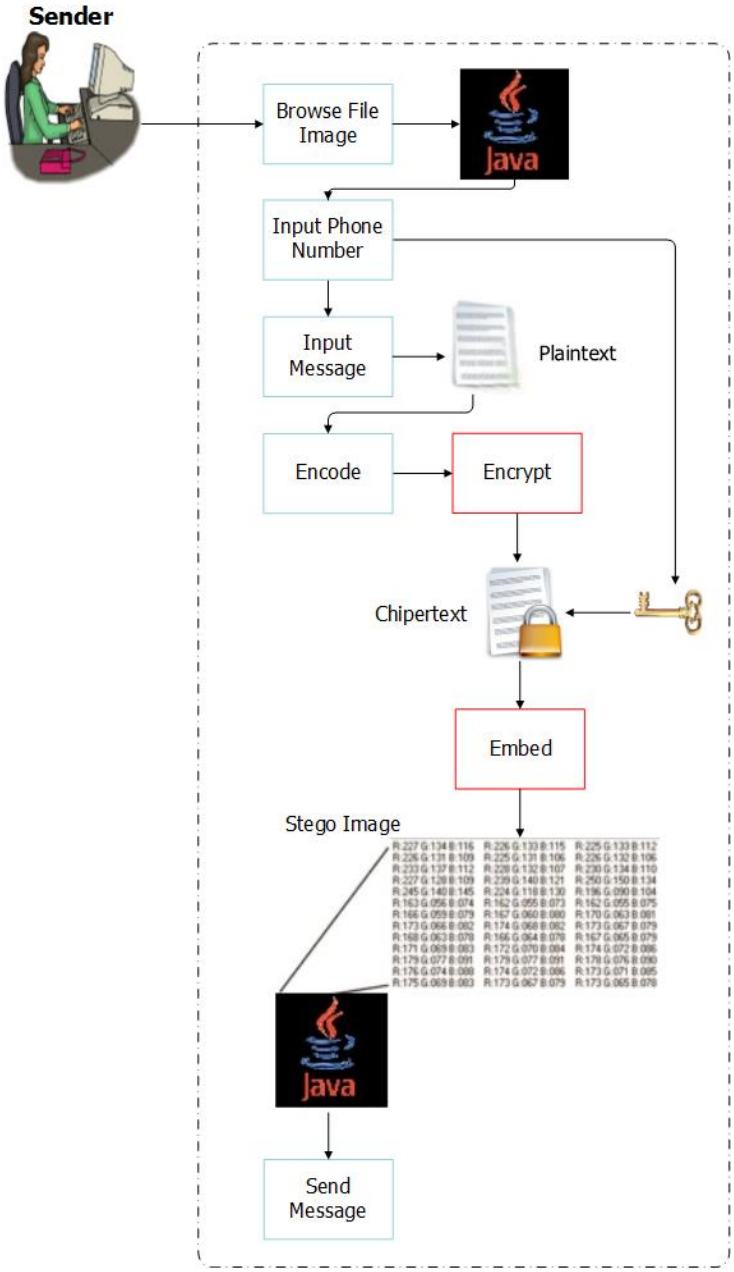
2. *Receiver Key*

a. *Receiver* masuk ke menu *inbox key*yang ada didalam aplikasi *steganography,* yang bertujuan untuk menyimpan pasangan kunci*(private key)* kedalam database aplikasi apabila telah menerima pesan *key* dari *sender*. *Private key* ini akan digunakan pada saat *system* melakukan proses *decrypt* pesan *chipertext* yang telah disisipkan ke dalam gambar*(stego image). Receiver* menyimpan pasangan kunci *(public key)* yang dimiliki *sender* kedalam database, pasangan kunci tersebut yaitu *private key* yang telah dikirimkan oleh *sender* melalui pesan singkat SMS.

b. *Receiver* memilih *file* citra .png yang telah disisipkan pesan*(stego image)* oleh *sender*, proses pada *system receiver* akan secara langsung melakukan proses *extracting* pesan yang ada didalam *stego image* menggunakan metode *Haar Wevelet* serta *system* akan melakukan proses *decrypt* pesan menggunakan Algoritma *RSA*, yang bertujuan untuk pesan*(chipertext)* bisa dikembalikan menjadi *(plaintext)* yang dimengerti oleh *receiver*.

1. **Proses pada *sender* pesan**

Tahapan-tahapan *sender* pesan dapat digambarkan lebih rinci melalui gambar 3.2



Gambar 3.2 Proses Pada *Sender* Pesan

Tahapan-tahapan yang dilakukan antara *sender* dan *receiver* yaitu sebagai berikut:

a. *Sender* memilih gambar type format .png dimana *file* citra gambar ini yang akan dijadikan tempat penampung informasi pesan*(stego image)*.

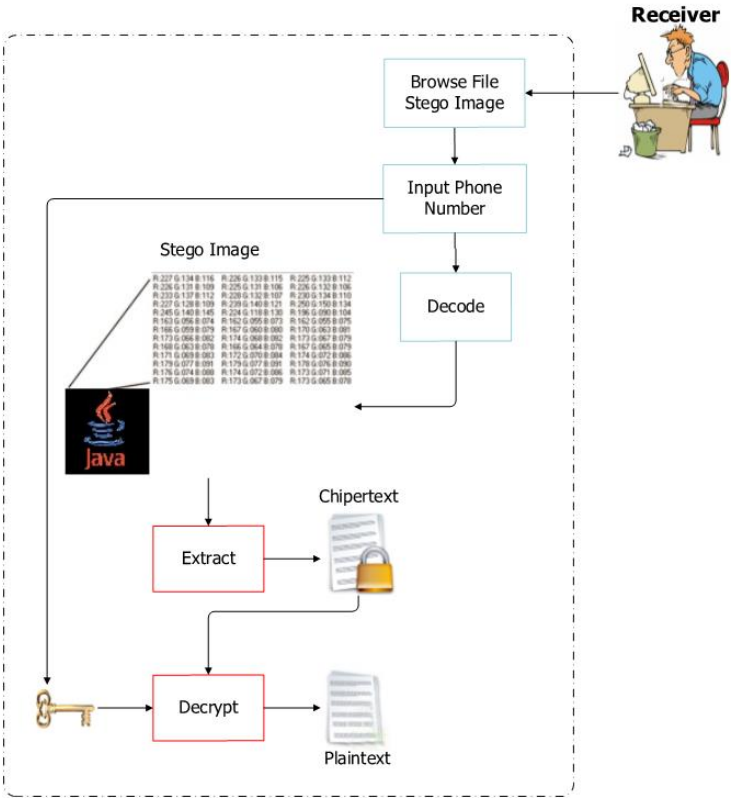
b. *Sender* memasukan *phone number receiver*, jika *phone number* sudah ada di dalam database dalam arti *sender* sudah pernah melakukan

generate key pada *phone number* tersebut, maka proses *encrypt* terhadap pesan*(plaintext)* bisa dilakukan dengan menggunakan pasangan kunci yang didapat dari proses *generate key* tersebut. Akan tetapi apabila *phone number* belum ada di dalam database atau *sender* belum melakukan proses *generate key* pada *phone number* yang akan digunakan dalam proses *sender* pesan, dipastikan proses *encrypt* akan gagal, hal ini dikarenakan tidak adanya pasangan kunci yang di simpan didalam database pada *phone number* tersebut.

c. *Sender* memasukan pesan*(plaintext)* yang akan di *encrypt*.
d. *Sender* memilih *button encode*, pada proses *encode system* akan melakukan proses *encrypt* pesan*(plaintext)* dengan menggunakan Algoritma *RSA* yang bertujuan sebagai keamanan datanya. Setelah proses *encrypt* berhasil *system* akan melakukan proses *embed*, dimana hasil *encrypt* akan disisipkan ke dalam *file* citra gambar menggunakan metode *Haar Wevelet* yang betujuan untuk meghasilkan *stego image*.
e. *Sender* mengirimkan *stego image* melalui mms kepada *receiver*.

2. Proses pada *receiver* pesan

Tahapan-tahapan *receiver* pesan dapat digambarkan lebih rinci melalui gambar 3.3



Gambar 3.3 Proses Pada *Receiver* Pesan

Tahapan-tahapan yang dilakukan antara *sender* dan *receiver* yaitu sebagai berikut:

a. *Receiver* memilih pesan *stego image* yang dikirimkan *sender*.
b. *Receiver* memasukan *phone number sender*, dalam hal ini agar hasil dari proses *cryptography* bisa diterjemahkan atau dikembalikan*(decrypt)* menjadi *paintext* yang sebenarnya.
c. *Receiver* memilih *file* citra gambar .png yang telah disisipkan pesan(*stego image*) oleh *sender*, proses pada *system receiver* secara langsung akan melakukan proses *extracting* pada pesan yang ada didalam *stego image* dengan menggunakan metode *Haar Wevelet*, serta *system* akan melakukan proses *decrypt* terhadap pesan *chipertext* dengan menggunakan Algoritma *RSA* yanag bertujuan untuk mengembalikan

pesan *chipertext* menjadi pesan *plaintext* agar bisa di mengerti oleh *receiver.*

3.1.3 Analisis Metode Haar Wevelet

Analisis metode *haar wavelet* terdiri dari 2 tahapan proses, yaitu proses *embedding*/*embedding* dan proses *extract*si/*extracion.* Tahapan-tahapan proses tersebut akan dijelaskan sebagai baerikut:

3.1.3.1 Contoh Kasus *Embedding* dan *Extracting* (Harr wavelet) dengan rangkaian karakter (*Plaintext*) "ANDY".

1. Analisis Proses *Embedding*/*Embedding*

Proses *embedding* yaitu proses menyembunyikan informasi kedalam media penampung, dalam hal ini media penampung berupa citra digital. Proses ini akan menghasilkan citra yang telah disisipkan pesan *(stego-object)* yang menyerupai dengan citra sebelum disisipkan pesan.

Contoh gambar A memiliki ukuran *pixel* 12x12, setiap blok *pixel* memiliki nilai sebagai berikut :



Gambar 3.4 Contoh Gambar A

Tabel 3.1 Contoh Nilai Pixel Gambar A

2	6	7	9	2	6	7	9	2	6	7	9
3	4	6	10	3	4	6	10	3	4	6	10
7	8	6	1	7	8	6	1	7	8	6	1
2	7	1	4	2	7	1	4	2	7	1	4
2	6	7	9	2	6	7	9	2	6	7	9
3	4	6	10	3	4	6	10	3	4	6	10
7	8	6	1	7	8	6	1	7	8	6	1
2	7	1	4	2	7	1	4	2	7	1	4
2	6	7	9	2	6	7	9	2	6	7	9
3	4	6	10	3	4	6	10	3	4	6	10
7	8	6	1	7	8	6	1	7	8	6	1
2	7	1	4	2	7	1	4	2	7	1	4

Tahapan-tahapan yang dilakukan pada setiap prosesnya yaitu sebagai berikut :

1. Merubah *cover* gambar menjadi 3 channel, yaitu *Red, Green* dan *Blue*
2. Menghitung nilai transfrom *row* dengan operasi penjumlahan dan pengurangan pada *pixel*.

   Rumus penjumlahan : *px1+px2/2*

   Rumus pengurangan : *px3-px4/2*. Hasilnya :

Tabel 3.2 Hasil Perhitungan *Transform Row* Pada *Channel Red*

4	8	4	8	4	8	-2	-1	-2	-1	-2	-1
3	8	3	8	3	8	0	-2	0	-2	0	-2
7	3	7	3	7	3	0	2	0	2	0	2
4	2	4	2	4	2	-2	-1	-2	-1	-2	-1
4	8	4	8	4	8	-2	-1	-2	-1	-2	-1
3	8	3	8	3	8	0	-2	0	-2	0	-2
7	3	7	3	7	3	0	2	0	2	0	2
4	2	4	2	4	2	-2	-1	-2	-1	-2	-1
4	8	4	8	4	8	-2	-1	-2	-1	-2	-1
3	8	3	8	3	8	0	-2	0	-2	0	-2
7	3	7	3	7	3	0	2	0	2	0	2
4	2	4	2	4	2	-2	-1	-2	-1	-2	-1

3. Setelah mendapatkan hasil dari transfrom *row* selanjutnya menghitung nilai transfrom *column* dengan operasi penjumlahan dan pengurangan pada *pixel*.

   Rumus penjumlahan : *px1+px13/2*

   Rumus pengurangan : *px25-px37/2*. Hasilnya :

Tabel 3.3 Hasil Perhitungan *Transform Column* Pada *Channel Red*

3	8	3	8	3	8	-1	-1	-1	-1	-1	-1
5	2	5	2	5	2	-1	0	-1	0	-1	0
3	8	3	8	3	8	-1	-1	-1	-1	-1	-1
5	2	5	2	5	2	-1	0	-1	0	-1	0
3	8	3	8	3	8	-1	-1	-1	-1	-1	-1
5	2	5	2	5	2	-1	0	-1	0	-1	0
0	0	0	0	0	0	-1	-1	-1	-1	-1	-1
1	0	1	0	1	0	-1	0	-1	0	-1	0
0	0	0	0	0	0	-1	-1	-1	-1	-1	-1
1	0	1	0	1	0	-1	0	-1	0	-1	0
0	0	0	0	0	0	-1	-1	-1	-1	-1	-1
1	0	1	0	1	0	-1	0	-1	0	-1	0

4. Menormalisasikan hasil dari pada penjumlahan dan pengurangan pada proses diatas dengan *range* (0 – 255).

   Rumus Normalisasi (0 - 255) : *round*($\frac{255}{\max - min} * (\mathrm{Px} - \min)$) Hasilnya :

Tabel 3.4 Hasil Perhitungan Normalisai Pada *Channel Red*

113	255	113	255	113	255	0	0	0	0	0	0
170	85	170	85	170	85	0	28	0	28	0	28
113	255	113	255	113	255	0	0	0	0	0	0
170	85	170	85	170	85	0	28	0	28	0	28
113	255	113	255	113	255	0	0	0	0	0	0
170	85	170	85	170	85	0	28	0	28	0	28
28	28	28	28	28	28	0	0	0	0	0	0
56	28	56	28	56	28	0	28	0	28	0	28
28	28	28	28	28	28	0	0	0	0	0	0
56	28	56	28	56	28	0	28	0	28	0	28
28	28	28	28	28	28	0	0	0	0	0	0
56	28	56	28	56	28	0	28	0	28	0	28

5. Mencari nilai koefisien yang nilainya <=1. Karena pada metode ini tempat untuk *embedding* informasi berada pada nilai tersebut. Pada proses ini terdapat 5 subproses lagi. yaitu :

a. Memasukan informasi pesan, misal pesan rahasia yang akan di masukan adalah = “ANDY” dan hasil *encrypt*nya C = *Ciphertext* (182091204072) dan M = *Plaintext* (65786889), ubah nilai *plaintext* ke dalam nilai *binary* 8 bit seperti tabel 3.5

Tabel 3.5 Konversi *Char* Ke *Binary*

Char	M	C	*Binary*
A	65	182	10110110
N	78	091	01011011
D	68	204	11001100
Y	89	072	01001000

b. Lakukan proses *generate header* pada *binary* yang telah didapatkan diatas, fungsi *generate header* yaitu untuk menampung informasi tentang total atau banyaknya *pixel*yang dibutuhkan untuk menyimpan *plaintext*.

   Dari hasil diatas 8 digit *binary* * 4(banyaknya char)=32 digit, dan nilai *header* yang digunakan dalam aplikasi ini adalah 7 digit (0000000) yang artinya informasi yang bisa ditampung hanya sebanyak 9999999 *pixel*. Pada contoh ini nilai *header*-nya menjadi =0000032, nilai header ini akan diubah ke *binary* pada setiap digitnya

   0 = 00000000

   0 = 00000000

   0 = 00000000

   0 = 00000000

   0 = 00000000

   3 = 00110011

   2 = 00110010

c. Gabungkan *Header* dengan *Chipertext*.

   00000000|00000000|00000000|00000000|00000000|00110011|00110010|10110110|01011011|11001100|01001000

   Sehingga banyaknya digit yang akan disisipkan ke dalam *pixel* yaitu 8*11 = 88 *pixel*.

d. Setelah mendapatkan nilai sisipkan setiap biner ke nilai batas koefisien yang <=1 pada channel yang sudah di normalisasi.

Tabel 3.6 Hasil Penyisipan Nilai Batas Koefisien Pada *Channel Red*

113	255	113	255	113	255	0	0	0	0	0	0
170	85	170	85	170	85	0	28	0	28	0	28
113	255	113	255	113	255	0	0	0	0	0	0
170	85	170	85	170	85	0	28	0	28	0	28
113	255	113	255	113	255	0	0	0	0	0	0
170	85	170	85	170	85	0	28	0	28	0	28
28	28	28	28	28	28	0	0	0	0	0	0
56	28	56	28	56	28	0	28	0	28	0	28
28	28	28	28	28	28	0	0	0	0	0	0
56	28	56	28	56	28	1	28	1	28	0	28
28	28	28	28	28	28	0	1	1	0	0	1
56	28	56	28	56	28	1	28	0	28	0	28

e. Dikarenakan pada channel *red* sudah tidak bisa menampung untuk tempat penyisipan nilai biner 8-bit yang nilai batas koefisien <=1, maka lakukan *embedding* pada channel *Green* dan *blue*.

Tabel 3.7 Hasil Penyisipan Nilai Batas Koefisien Pada *Channel Green*

113	255	113	255	113	255	1	0	1	0	1	1
170	85	170	85	170	85	0	28	1	28	1	28
113	255	113	255	113	255	0	0	1	0	1	1
170	85	170	85	170	85	0	28	1	28	1	28
113	255	113	255	113	255	1	1	0	0	1	1
170	85	170	85	170	85	0	28	0	28	0	28
28	28	28	28	28	28	1	0	0	1	0	0
56	28	56	28	56	28	0	28	0	28	0	28
28	28	28	28	28	28	0	0	0	0	0	0
56	28	56	28	56	28	0	28	0	28	0	28
28	28	28	28	28	28	0	0	0	0	0	0
56	28	56	28	56	28	0	28	0	28	0	28

Keterangan warna:

Biru : nilai *header*.

Merah : nilai *chipertext*.

6. Denormalisasi hasil dari penysisipan dengan *range* 0 - 255. Dikarenakan nilai min = 0, maka hasilnya pun tidak berubah jika di denormalisasikan.

Rumus denormalisasi (0 - 255): *round* $\left(\left(\frac{\max - min}{255} * px\right) + min\right)$ Hasilnya :

Tabel 3.8 Hasil Denormalisai Dari Penyisipan Pada *Channel Red*

3	8	3	8	3	8	-1	-1	-1	-1	-1	-1
5	2	5	2	5	2	-1	0	-1	0	-1	0
3	8	3	8	3	8	-1	-1	-1	-1	-1	-1
5	2	5	2	5	2	-1	0	-1	0	-1	0
3	8	3	8	3	8	-1	-1	-1	-1	-1	-1
5	2	5	2	5	2	-1	0	-1	0	-1	0
0	0	0	0	0	0	-1	-1	-1	-1	-1	-1
1	0	1	0	1	0	-1	0	-1	0	-1	0
0	0	0	0	0	0	-1	-1	-1	-1	-1	-1
1	0	1	0	1	0	-1	0	-1	0	-1	0
0	0	0	0	0	0	-1	-1	-1	-1	-1	-1
1	0	1	0	1	0	-1	0	-1	0	-1	0

Tabel 3.9 Hasil Denormalisai Dari Penyisipan Pada *Channel Green*

3	8	3	8	3	8	-1	-1	-1	-1	-1	-1
5	2	5	2	5	2	-1	0	-1	0	-1	0
3	8	3	8	3	8	-1	-1	-1	-1	-1	-1
5	2	5	2	5	2	-1	0	-1	0	-1	0
3	8	3	8	3	8	-1	-1	-1	-1	-1	-1
5	2	5	2	5	2	-1	0	-1	0	-1	0
0	0	0	0	0	0	-1	-1	-1	-1	-1	-1
1	0	1	0	1	0	-1	0	-1	0	-1	0
0	0	0	0	0	0	-1	-1	-1	-1	-1	-1
1	0	1	0	1	0	-1	0	-1	0	-1	0
0	0	0	0	0	0	-1	-1	-1	-1	-1	-1
1	0	1	0	1	0	-1	0	-1	0	-1	0

7. Hitung nilai *invers row*.

   Cara menghitung nilai *invers*i yaitu dengan membagi menjadi 2 bagian, setelah itu lakukan penjumlahan dan pengurangan secara menyilang. Pada *pixel* ganjil lakukan penjumlahan (+) dan untuk *pixel* genap lakukan pengurangan (-). Hasilnya :

Tabel 3.10 Hasil Perhitungan Invers Row Pada *Channel Red*

2	7	2	7	2	7	4	9	4	9	4	9
4	2	4	2	4	2	6	2	6	2	6	2
2	7	2	7	2	7	4	9	4	9	4	9
4	2	4	2	4	2	6	2	6	2	6	2
2	7	2	7	2	7	4	9	4	9	4	9
4	2	4	2	4	2	6	2	6	2	6	2
-1	-1	-1	-1	-1	-1	1	1	1	1	1	1
0	0	0	0	0	0	2	0	2	0	2	0
-1	-1	-1	-1	-1	-1	1	1	1	1	1	1
0	0	0	0	0	0	2	0	2	0	2	0
-1	-1	-1	-1	-1	-1	1	1	1	1	1	1
0	0	0	0	0	0	2	0	2	0	2	0

8. Hitung nilai *invers column.* Hasilnya :

Tabel 3.11 Hasil Perhitungan Invers Column Pada *Channel Red*

1	6	1	6	1	6	5	10	5	10	5	10
4	2	4	2	4	2	8	2	8	2	8	2
1	6	1	6	1	6	5	10	5	10	5	10
4	2	4	2	4	2	8	2	8	2	8	2
1	6	1	6	1	6	5	10	5	10	5	10
4	2	4	2	4	2	8	2	8	2	8	2
3	8	3	8	3	8	3	8	3	8	3	8
4	2	4	2	4	2	4	2	4	2	4	2
3	8	3	8	3	8	3	8	3	8	3	8
4	2	4	2	4	2	4	2	4	2	4	2
3	8	3	8	3	8	3	8	3	8	3	8
4	2	4	2	4	2	4	2	4	2	4	2

Dibawah ini adalah hasil dari pada perhitungan nilai *invers row* channel green. Hasilnya :

Tabel 3.12 Hasil Perhitungan Invers Row Pada *Channel Green*

2	7	2	7	2	7	4	9	4	9	4	9
4	2	4	2	4	2	6	2	6	2	6	2
2	7	2	7	2	7	4	9	4	9	4	9
4	2	4	2	4	2	6	2	6	2	6	2
2	7	2	7	2	7	4	9	4	9	4	9
4	2	4	2	4	2	6	2	6	2	6	2
-1	-1	-1	-1	-1	-1	1	1	1	1	1	1
0	0	0	0	0	0	2	0	2	0	2	0
-1	-1	-1	-1	-1	-1	1	1	1	1	1	1
0	0	0	0	0	0	2	0	2	0	2	0
-1	-1	-1	-1	-1	-1	1	1	1	1	1	1
0	0	0	0	0	0	2	0	2	0	2	0

Dibawah ini adalah hasil dari pada perhitungan nilai *invers column* channel green. Hasilnya :

Tabel 3.13 Hasil Perhitungan Invers Column Pada *Channel Green*

1	6	1	6	1	6	5	10	5	10	5	10
4	2	4	2	4	2	8	2	8	2	8	2
1	6	1	6	1	6	5	10	5	10	5	10
4	2	4	2	4	2	8	2	8	2	8	2
1	6	1	6	1	6	5	10	5	10	5	10
4	2	4	2	4	2	8	2	8	2	8	2
3	8	3	8	3	8	3	8	3	8	3	8
4	2	4	2	4	2	4	2	4	2	4	2
3	8	3	8	3	8	3	8	3	8	3	8
4	2	4	2	4	2	4	2	4	2	4	2
3	8	3	8	3	8	3	8	3	8	3	8
4	2	4	2	4	2	4	2	4	2	4	2

9. Setelah mendapatkan nilai *invers*i, ubah dari 3 *channel* RGB menjadi *cover* gambar.

2. Analisis Proses Pengungkapan / *Extracting*.

Proses *extract*si yaitu proses pengambilan informasi yang tersembunyi pada citra digital. Proses ini akan menghasilkan *file* informasi yang disembunyikan, dengan masukan berupa citra *stego-object. Extract*si data dari koefisien DWT mungkin dilakukan karena *steganography* menyisipkan data setelah melalui semua proses *(lossles compression)* PNG, sehingga apapun proses yang dilalui sesudahnya dapat dikembalikan kepada bentuk semula.

Tahapan-tahapan yang dilakukan pada setiap prosesnya yaitu sebagai berikut :

1. Merubah *cover* gambar menjadi 3 channel, yaitu *Red, Green* dan *Blue*

Tabel 3.14 Pixel *Channel Red*

1	6	1	6	1	6	5	10	5	10	5	10
4	2	4	2	4	2	8	2	8	2	8	2
1	6	1	6	1	6	5	10	5	10	5	10
4	2	4	2	4	2	8	2	8	2	8	2
1	6	1	6	1	6	5	10	5	10	5	10
4	2	4	2	4	2	8	2	8	2	8	2
3	8	3	8	3	8	3	8	3	8	3	8
4	2	4	2	4	2	4	2	4	2	4	2
3	8	3	8	3	8	3	8	3	8	3	8
4	2	4	2	4	2	4	2	4	2	4	2
3	8	3	8	3	8	3	8	3	8	3	8
4	2	4	2	4	2	4	2	4	2	4	2

Tabel 3.15 Pixel *Channel Green*

1	6	1	6	1	6	5	10	5	10	5	10
4	2	4	2	4	2	8	2	8	2	8	2
1	6	1	6	1	6	5	10	5	10	5	10
4	2	4	2	4	2	8	2	8	2	8	2
1	6	1	6	1	6	5	10	5	10	5	10
4	2	4	2	4	2	8	2	8	2	8	2
3	8	3	8	3	8	3	8	3	8	3	8
4	2	4	2	4	2	4	2	4	2	4	2
3	8	3	8	3	8	3	8	3	8	3	8
4	2	4	2	4	2	4	2	4	2	4	2
3	8	3	8	3	8	3	8	3	8	3	8
4	2	4	2	4	2	4	2	4	2	4	2

2. Menghitung nilai transfrom *row* dengan operasi penjumlahan dan pengurangan pada *pixel*.
   Rumus penjumlahan :*px1+px2/2*
   Rumus pengurangan :*p31-p4/2x.* Hasilnya :

Tabel 3.16 Hasil Perhitungan *Transform Row* Pada *Channel Red*

3	3	3	7	7	7	-2	-2	-2	-2	-2	-2
3	3	3	5	5	5	1	1	1	3	3	3
3	3	3	7	7	7	-2	-2	-2	-2	-2	-2
3	3	3	5	5	5	1	1	1	3	3	3
3	3	3	7	7	7	-2	-2	-2	-2	-2	-2
3	3	3	5	5	5	1	1	1	3	3	3
5	5	5	5	5	5	-2	-2	-2	-2	-2	-2
3	3	3	3	3	3	1	1	1	1	1	1
5	5	5	5	5	5	-2	-2	-2	-2	-2	-2
3	3	3	3	3	3	1	1	1	1	1	1
5	5	5	5	5	5	-2	-2	-2	-2	-2	-2
3	3	3	3	3	3	1	1	1	1	1	1

Tabel 3.17 Hasil Perhitungan *Transform Row* Pada *Channel Green*

3	3	3	7	7	7	-2	-2	-2	-2	-2	-2
3	3	3	5	5	5	1	1	1	3	3	3
3	3	3	7	7	7	-2	-2	-2	-2	-2	-2
3	3	3	5	5	5	1	1	1	3	3	3
3	3	3	7	7	7	-2	-2	-2	-2	-2	-2
3	3	3	5	5	5	1	1	1	3	3	3
5	5	5	5	5	5	-2	-2	-2	-2	-2	-2
3	3	3	3	3	3	1	1	1	1	1	1
5	5	5	5	5	5	-2	-2	-2	-2	-2	-2
3	3	3	3	3	3	1	1	1	1	1	1
5	5	5	5	5	5	-2	-2	-2	-2	-2	-2
3	3	3	3	3	3	1	1	1	1	1	1

3. Setelah mendapatkan hasil dari transfrom *row* selanjutnya menghitung nilai transfrom *column* dengan operasi penjumlahan dan pengurangan pada *pixel*.
   Rumus penjumlahan : *px1+px13/2*
   Rumus pengurangan :*px25-px37/*2. Hasilnya :

Tabel 3.18 Hasil Perhitungan *Transform Column* Pada *Channel Red*

3	3	3	6	6	6	0	0	0	0	0	0
3	3	3	6	6	6	0	0	0	0	0	0
3	3	3	6	6	6	0	0	0	0	0	0
4	4	4	4	4	4	0	0	0	0	0	0
4	4	4	4	4	4	0	0	0	0	0	0
4	4	4	4	4	4	0	0	0	0	0	0
0	0	0	1	1	1	-1	-1	-1	-1	-1	-1
0	0	0	1	1	1	-1	-1	-1	-1	-1	-1
0	0	0	1	1	1	-1	-1	-1	-1	-1	-1
1	1	1	1	1	1	-1	-1	-1	-1	-1	-1
1	1	1	1	1	1	-1	-1	-1	-1	-1	-1
1	1	1	1	1	1	-1	-1	-1	-1	-1	-1

Tabel 3.19 Hasil Perhitungan *Transform Column* Pada *Channel Red*

3	3	3	6	6	6	0	0	0	0	0	0
3	3	3	6	6	6	0	0	0	0	0	0
3	3	3	6	6	6	0	0	0	0	0	0
4	4	4	4	4	4	0	0	0	0	0	0
4	4	4	4	4	4	0	0	0	0	0	0
4	4	4	4	4	4	0	0	0	0	0	0
0	0	0	1	1	1	-1	-1	-1	-1	-1	-1
0	0	0	1	1	1	-1	-1	-1	-1	-1	-1
0	0	0	1	1	1	-1	-1	-1	-1	-1	-1
1	1	1	1	1	1	-1	-1	-1	-1	-1	-1
1	1	1	1	1	1	-1	-1	-1	-1	-1	-1
1	1	1	1	1	1	-1	-1	-1	-1	-1	-1

4. Menormalisasikan hasil dari pada penjumlahan dan pengurangan pada proses diatas dengan *range* (0 – 255).

   Rumus Normalisasi (0 - 255) : *round*($\frac{255}{max - min} * (Px - min)$) Hasilnya :

Tabel 3.20 Hasil Normalisai Dari Penyisipan Pada *Channel Red*

113	255	113	255	113	255	0	0	0	0	0	0
170	85	170	85	170	85	0	28	0	28	0	28
113	255	113	255	113	255	0	0	0	0	0	0
170	85	170	85	170	85	0	28	0	28	0	28
113	255	113	255	113	255	0	0	0	0	0	0
170	85	170	85	170	85	0	28	0	28	0	28
28	28	28	28	28	28	0	0	0	0	0	0
56	28	56	28	56	28	0	28	0	28	0	28
28	28	28	28	28	28	0	0	0	0	0	0
56	28	56	28	56	28	1	28	1	28	0	28
28	28	28	28	28	28	0	1	1	0	0	1
56	28	56	28	56	28	1	28	0	28	0	28

Tabel 3.21 Hasil Normalisai Dari Penyisipan Pada *Channel Green*

113	255	113	255	113	255	1	0	1	0	1	1
170	85	170	85	170	85	0	28	1	28	1	28
113	255	113	255	113	255	0	0	1	0	1	1
170	85	170	85	170	85	0	28	1	28	1	28
113	255	113	255	113	255	1	1	0	0	1	1
170	85	170	85	170	85	0	28	0	28	0	28
28	28	28	28	28	28	1	0	0	1	0	0
56	28	56	28	56	28	0	28	0	28	0	28
28	28	28	28	28	28	0	0	0	0	0	0
56	28	56	28	56	28	0	28	0	28	0	28
28	28	28	28	28	28	0	0	0	0	0	0
56	28	56	28	56	28	0	28	0	28	0	28

5. Ambil 56 *pixel* pertama yang nilai koefisiennya <=1, ini adalah ***Header***. Karena 56 ini di dapat dari perhitungan 7 digit header * 8 digit (banyaknya biner).

   00000000|**00000000**|**00000000**|**00000000**|**00000000**|**00110011**|**00110010**|

   00000000 = 0
   00000000 = 0
   00000000 = 0
   00000000 = 0
   00000000 = 0
   00110011 = 3
   00110010 = 2

6. Ambil *pixel* ke 57 sampai dengan ke 89 yang nilai koefisiennya <=1, ini adalah ***Chipertext***. Karena 89 ini di dapat dari perhitungan 4 char * 8digit (banyaknya biner) = 32 digtit
   10110110|**01011011**|**11001100**|**01001000** , ubah nilai *binar 8 bit* ke dalam nilai *plaintext* seperti tabel 3.22 :

Tabel 3.22 Konversi *Binary* ke *Char*

Binary	**C**	**M**	**Char**
10110110	182	65	A
01011011	091	78	N
11001100	204	68	D
01001000	072	89	Y

3.1.4 Analisis Algoritma RSA

Skema Algoritma *cryptography public key* (asimetris) *RSA* terdiri dari tiga proses yaitu proses *encrypt*, dan proses *decrypt*.

3.1.4.1 Contoh Kasus *Encrypt* dan *Decrypt* (RSA) dengan rangkaian karakter (*Plaintext*) "ANDY".

1. *Encrypt* (*RSA*) Modulus 3 Digit

Contoh kasus kali ini menggunakan (*RSA*) modulus yang berjumlah 3 digit, langkah yang dilakukan Algoritma *RSA*, berikut contoh pengerjaannya :

1. Pembentukan Kunci
   a. Menentukan 2 bilangan prima sembarang dan acak yang diwakilkan oleh variable *p* dan variabel *q*.
      Dimisalkan $p = 13$ dan $q = 17$
   b. Menghitung (*RSA*) modulus *(n)* dengan formula $n = p * q$, dimana nilai p=13 dan nilai q = 17
      $n = p * q$
      $n = 13 * 17$
      $n = 221$
   c. Menghitung nilai *m* yang akan digunakan untuk mencari (*RSA*) *enciphering exponet (e).*
      $m = (p - 1) * (q - 1)$

*m = (13 – 1) * (17 – 1)*

*m = (12) * (16)*

m = 192

d. Menghitung nilai *e* dengan formula : *gcd(e,m) = 1*, dengan syarat

e = bilangan prima dan *1 < e < m*.

dimisalkan e = 5, maka formula gcd(5,192) = 1 bernilai true.

e. Menghitung nilai (*RSA*) deciphering exponent (d) dengan menggunakan formula e * d = 1 mod (m), dimana nilai e dan m didapatkan dari langkah sebelumnya e = 5 dan m = 192.

*e * d = 1 mod (m)*

*d = 1 + (k * m) / e*

*d = 1 (k * 192) /5*

dengan nilai k = integer sembarang, maka dimisalkan nilai d yang akan diambil adalah d yang bernilai integer. Nilai d yang diambil kali ini adalah d = 77.

f. Dari langkah diatas, nilai n, e, dan d telah ditemukan yang berarti juga pasangan kunci telah terbentuk.

Pasangan *public key* (n,e) = (221, 5)

Pasangan kunci rahasia (n,d) = (221, 77)

2. Dimisalkan terdapat himpunan karakter: “ANDY” yang akan dienkrip menggunakan kunci yang telah dibentuk pada langkah sebelumnya, maka terlebih dahulu karakter tersebut dikonversi kedalam bentuk numerik, proses konversi bisa menggunakan teknik tersendiri dari *user* atau menggunakan tabel ASCII seperti contoh 3.23 :

Tabel 3.23 Konversi *Char* ke ASCII

Char	ASCI Dec
A	65
N	78
D	68
Y	89

Dari tabel 3.3 hasil konversi setiap baris dijadikan saru deret, akan menghasilkan *plaintext* ASCII ***(M)*** dari karakter yang akan di enkripsi sebagai berikut : **M = 65786889**

3. Untuk menjalankan proses *encrypt*, digunakan *public key* yang telah dibentuk sebelumnya yaitu *public key* (221,5), dengan formula $C = M^e \ mod \ n$. Namun sebelum melakukan perhitungan, terlebih dahulu dilakukan pemecahan deret *plaintext* ASCII menjadi blok yang panjang digit setiap bloknya kurang dari panjang digit n. Pada contoh kali ini, (*RSA*) modulus yang digunakan adalah = 221 (3 digit), jadi untuk setiap blok dibatasi maksimal 2 digit per blok.

   Hasil perhitungan *encrypt* dengan formula $C = M^e \ mod \ n$ sebagai 3.24 :

Tabel 3.24 Hasil Perhitungan *Encrypt*

Blok ke	M	e	N	C
1	65	5	221	182
2	78	5	221	91
3	68	5	221	204
4	89	5	221	72

   Dari hasil nilai C pada tabel 3.24, dilakukan penyamaan panjang digit setiap blok dengan menambahkan 0 sebanyak kekurangan panjang digit blok tersebut, kemudian jadikan suatu deret, maka akan dihasilkan deret ciphertext : **C =182091204072**

4. Hasil *encrypt* telah didapatkan yaitu **C** = 182 091 204 072, deret ini akan dipenggal 3 karakter menjadi satu blok. Setelah dipenggal per 3 karakter, lalu ada nilai yang kurang dari 3 digit maka akan ditambahkan nilai nol (0) di depannya agar menjaga konsistensi panjang setiap blok sehingga tidak terjadi kesalahan pada proses *decrypt*.

2. *Decrypt* (*RSA*) Modulus 3 Digit

Tabel 3.25 Hasil Deret *Chipertext*

Block Ke	C
1	182
2	091
3	204
4	072

Dari tabel 3.25 dihasilkan deret C =182091204072 Untuk membuktikan pasangan *public key* dan kunci rahasia berjalan sempurna, dicoba untuk dekripsi *ciphertext* yang telah dienkrip sebelumnya. Proses *decrypt* menggunakan kunci rahasia (221, 77), dengan formula $M = C^d \, mod \, n$. Hasil perhitungan untuk proses *decrypt* ditunjukkan pada tabel 3.26

Tabel 3.26 Hasil Perhitungan *Decrypt*

Blok ke	C	d	n	M
1	182	77	221	65
2	91	77	221	78
3	204	77	221	68
4	72	77	221	89

Jika hasil perhitungan *decrypt* (nilai M) dijadikan deret akan terbentuk kembali deret *plaintext* ASCII yang sama antara deret sebelum di *encrpt* dan setelah di dekrip yaitu : **M = 65786889**

Langkah terakhir adalah mengubah deret plaintext ASCII menjadi *plaintext* menggunakan tabel ASCII sebagai pedoman konversi.

Tabel 3.27 Konversi ASCII ke *Char*

ASCII Dec	Char
65	A
78	N
68	D
89	Y

Pembentukan nilai *e* dengan formula *gcd(e,m) = 1* , yang berarti nilai FPB antara kedua variabel *e* dan *m* harus bernilai 1 atau kedua bilangan tersebut harus relatif prima. Syarat *e* adalah *1 < e < m*, sehingga untuk mendapatkan nilai *e* dimiliki kemungkinan lebih dari 1, karena kemungkinan bilangan prima dari 1 hingga m bisa lebih dari 1.

Sedangkan dalam pembentukan nilai d dengan formula akhir *"d = 1 + (k * m) / e"* dimungkinkan ditambahkan aturan lain untuk mendapatkan nilai *d*. Variabel *k* adalah nilai integer sembarang, yang jika dimasukkan kedalam formula akan menghasilkan nilai *d*, namun nilai *d* yang memenuhi syarat adalah nilai *d* yang integer. Sehingga dalam proses penentuan nilai *d* dengan menggunakan nilai *k* sebagai integer sembarang, dapat ditambahkan aturan misalnya dalam penentuan *range* percobaan nilai *k*, atau nilai *d* yang akan diambil adalah *d* yang bernilai integer yang setelah berapa kali ditemukan dan lain sebagainya.

Salah satu syarat wajib yang menjadi perhatian dalam proses *encrypt* dan *decrypt* adalah nilai *M* atau *Message* atau *Number* harus lebih kecil jika dibandingkan nilai Modulus, yang berarti untuk membuatnya berada dibatas aman (selalu lebih kecil) ditetapkan jumlah digit maksimal nilai *M* atau *Message* atau *Number* adalah jumlah digit nilai modulus dikurang 1.

Untuk mengubah *string* menjadi bilangan tidak diharuskan menggunakan tabel konversi tertentu sehingga *user* dapat menerapkan tabel konversi sendiri sesuai dengan kebutuhan *user*.

Pada akhirnya, untuk mengimplementasi Algoritma *cryptography asimetris RSA* dimungkinkan untuk diterapkan banyak kondisi dan batasan yang menyesuaikan dengan kebutuhan *user*, sehingga memungkinkan dalam penerapan *RSA* terbentuk Algoritma yang berbeda dari tiap penyusun sintak diluar Algoritma *RSA* yang utama. Hal ini memungkinkan untuk menjadi alasan tidak dapat dibentuk *function* khusus untuk Algoritma *cryptography asimetris RSA* karena banyaknya sisi Algoritma yang tidak mutlak.

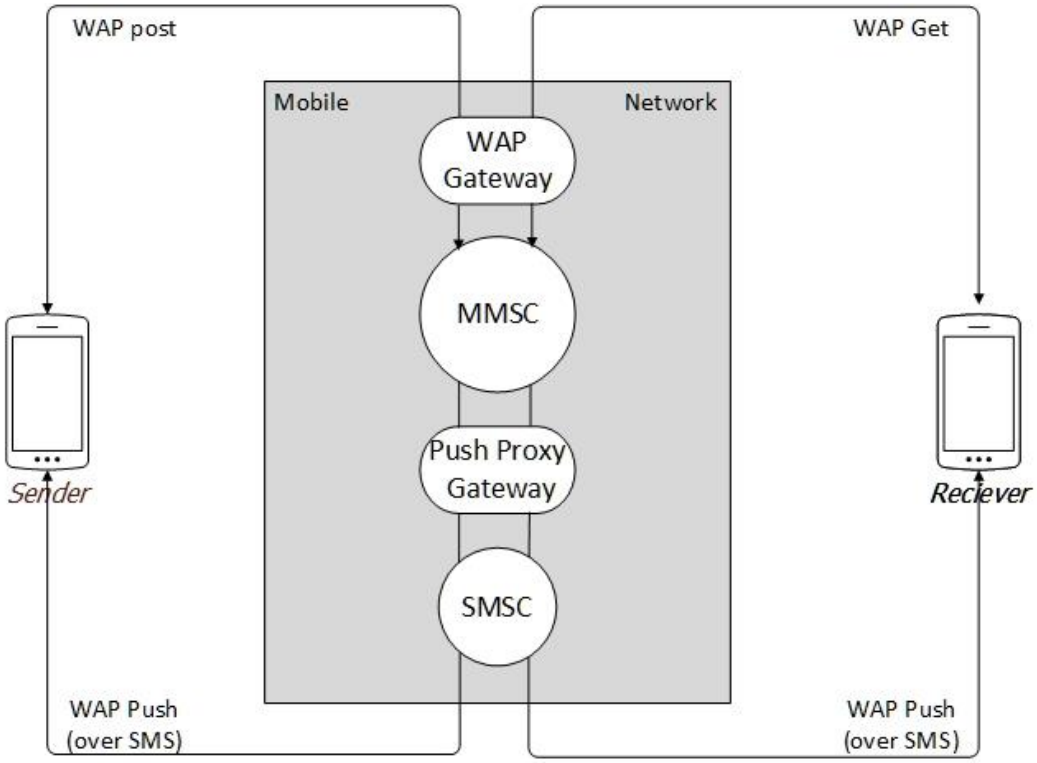
Penerapan Algoritma *cryptography* pada basis data salah satunya menguntungkan dalam sisi monitoring program aplikasi. Jika implementasi Algoritma *cryptography* berada di dalam program aplikasi, maka tingkat kecepatan proses perhitungan yang terjadi akan bergantung salah satunya pada perangkat keras di mana program aplikasi tersebut dijalankan. Sementara itu, jika Algoritma *cryptography* berada pada sisi basis data, yang menjadi sorotan dalam proses pemantauan kinerja hanya terpusat pada perangkat keras dimana basis data

tersebut dijalankan, bukan pada dimana masing-masing program aplikasi dijalankan.

3.1.5 Analisis MMS di aplikasi

Sender pesan MMS di Aplikasi *steganography* ini sama seperti *sender* MMS biasa hanya saja gambar yang dikirmkan telah disisipkan informasi didalamnya. Dalam proses *sender* pesan di aplikasi ini sebenarnya hampir mirip dengan *sender* pesan MMS. Dalam proses *receiver*, pesan hanya akan masuk ke dalam pesan masuk di aplikasi bawaan ponsel saja. Pesan masuk masuk di aplikasi bawaan ponsel yaitu pesan yang masih ter*encode*, maka dari itu dibutuhkannya aplikasi untuk me*decode* pesan(*stego image*) yang masuk. Proses *sender* dan *receiver* pesan terbagi menjadi beberapa tahapan sebagai berikut :

1. *Sender* melakukan *send stego image* ke *receiver* melalui MMSC menggunakan WAP Post, dan pesan tersimpan dam MMSC(*store*).
2. MMSC memberitahukan(*notification*) ke *reciever* via SMS dengan WAP Push.
3. Setelah menerima notifikasi untuk mengambil pesan MMS, *receiver* melakukan inisialisasi hubungan WAP Get untuk mengambil(*fetch*) MMS di MMSC.
4. Setelah semua proses berjalan maka *report* via SMS dikirim ke *sender*.
5. *Reciever* menerima pesan yang dikirimkan oleh *sender* berupa *stego image*.

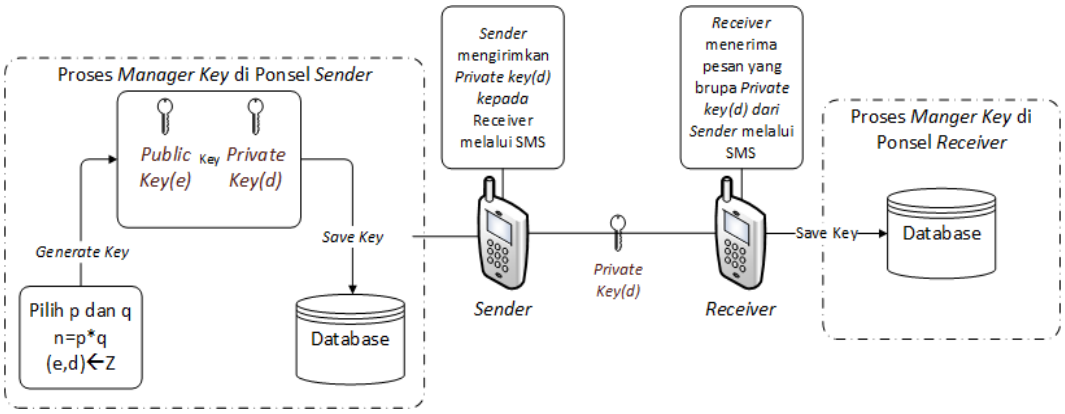


Gambar 3.5 Proses *Sender* Dan *Receiver* Pesan

3.1.6 Analisis *Manager Key*.

Manager Key atau *Key Management* adalah sebuah manajemen kunci di dalam sebuah *system cryptography*. *Manager Key* ini berurusan dengan *generate*, pertukaran, penyimpanan, dan *user* kunci. *Manager Key* ini berkaitan dengan *user*, baik antara *user* dengan *user* atau *user* dengan *system*. *Manager Key* ini berbeda dengan penjadwalan kunci, karena penjadwalan kunci biasanya mengacu pada penanganan material kunci dalam operasi ciphernya. *Manager Key* penting dalam sebuah *system cryptography* karena kesuksesan sebuah *system cryptography* tergantung pada pengelolaan *Manager Key*. Pada aplikasi kemanan MMS ini juga terdapat *Manager Key*, dimana *Manager Key* ini dibuat untuk mengamankan kunci yang akan dikirimkan *sender* pesan kepada *receiver* pesan. *Manager Key* di aplikasi *steganography* ini dilakukan dengan cara autentifikasi nomor telepon, maksudnya setiap *receiver* pesan mendapatkan *private key* yang

berbeda-beda. Proses pada *manager key* terbagi menjadi beberapa tahapan seperti digambarakan pada gambar 3.6



Gambar 3.6 Gambar *Manager Key* atau *Management Key*

3.1.6.1 Contoh Dalam Membangkitkan Sepasang Kunci.

3.2.2.2 Berikut contoh pengerjaanya:

a. Menentukan 2 bilangan prima sembarang dan acak yang diwakilkan oleh variable *p* dan variabel *q*.

   Dimisalkan *p* = 13 dan *q* = 17

b. Menghitung (*RSA*) modulus *(n)* dengan formula *n = p * q*, dimana nilai p=13 dan nilai q = 17

   *n = p * q*

   *n = 13 * 17*

   n = 221

c. Menghitung nilai *m* yang akan digunakan untuk mencari (*RSA*) *enciphering exponet (e).*

   *m = (p – 1) * (q – 1)*

   *m = (13 – 1) * (17 – 1)*

   *m = (12) * (16)*

   m = 192

d. Menghitung nilai *e* dengan formula : *gcd(e,m) = 1*, dengan syarat

   e = bilangan prima dan *1 < e < m*. dimisalkan e = 5, maka formula gcd(5,192) = 1 bernilai true.

e. Menghitung nilai (*RSA*) deciphering exponent (d) dengan menggunakan formula e * d = 1 mod (m), dimana nilai e dan m didapatkan dari langkah sebelumnya e = 5 dan m = 192.

   *e * d = 1 mod (m)*

   *d = 1 + (k * m) / e*

   *d = 1 (k * 192) /5*

   dengan nilai k = integer sembarang, maka dimisalkan nilai d yang akan diambil adalah d yang bernilai integer. Nilai d yang diambil kali ini adalah d = 77.

f. Dari langkah diatas, nilai n, e, dan d telah ditemukan yang berarti juga pasangan kunci telah terbentuk.

   Pasangan *public key* (n,e) = (221, 5)

   Pasangan kunci rahasia (n,d) = (221, 77)

g. *Sender* menyimpan kunci kedalam *database*.

h. *Private key* dikirimkan kepada *receiver* pesan melalui sms.

i. *Receiver* menerima pesan sms yang berupa kunci lalu menyimpannya ke dalam database berdasarkan nomor ponsel *sender*.

3.1.7 Analisis Kebutuhan non-Fungsional

Analisis kebutuhan nonfungsional adalah sebuah langkah dimana seorang pembangun aplikasi menganalisis sumber daya yang dibutuhkan untuk menggunakan aplikasi yang akan dibangun. Analisis kebutuhan nonfungsional yang dilakukan dibagi dalam tiga tahap, yaitu analisis *user*, analisis kebutuhan perangkat keras, dan analisis perangkat lunak.

3.1.7.1 Analisis User

User yang akan menggunakan aplikasi *steganography* dengan kebutuhan spesifikasi seperti pada tabel 3.28

Tabel 3.28 Kebutuhan *User* Aplikasi

Tipe	Hak Akses	Tingkat Keterampilan	Jenis pelatihan
User	Dapat melakukan operasi *embedding* pesan kedalam citra dan dapat melakukan operasi *extracting* pesan dari citra .	Dapat mengoperasikan ponsel dengan *system* operasi *android* dan memahami konsep *Multimedia Messaging Service* (MMS).	Tidak diperlukan pelatihan khusus.
	Dapat melakukan operasi *extracting* pesan dari citra dan dapat melakukan operasi *embedding* pesan dari citra.	Dapat mengoperasikan ponsel dengan *system* operasi *android* dan memahami konsep *Multimedia Messaging Service* (MMS).	Tidak diperlukan pelatihan khusus.

3.1.7.2 Analisis Kebutuhan Perangkat Keras

Perangkat keras merupakan salah satu kebutuhan yang sangat penting bagi pembuatan *steganography*. Perangkat keras akan mempengaruhi kinerja dari pembuatan *steganography*, semakin tinggi spesifikasi dari perangkat keras yang digunakan maka akan semakin cepat pula pembuatan *steganography*nya.

Perangkat keras yang digunakan pada pembangunan aplikasi *steganography* ini yaitu seperti pada tabel 3.29

Tabel 3.29 Spesifikasi Perangkat Keras

No	Nama Perangkat	Spesifikasi
1	Prosessor	AMD FX(tm)-4100 (4CPUs) 3.6 GHz
2	Monitor	Monitor 14 inch (1366x768)
3	*Memory*	RAM 8 GB DDR3
4	Harddisk	500 GB SATA 7200rpm

Sedangkan kebutuhan perangkat keras untuk menjalankan aplikasi yang dibangun, yang harus dipenuhi yaitu seperti pada tabel 3.30

Tabel 3.30 Spesifikasi Perangkat Keras

No	Nama Perangkat	Spesifikasi
1	Prosessor	AMD FX(tm)-4100 (4CPUs) 3.6 GHz
2	Monitor	Monitor 14 inch (1366x768)
3	*Memory*	RAM 2 GB DDR3
4	Harddisk	250 GB SATA 7200rpm

3.1.7.3 Analisis Perangkat Lunak

Perangkat lunak yang digunakan untuk membangun aplikasi *steganography* ini yaitu seperti pada tabel 3.31

Tabel 3.31 Spesifikasi Perangkat Lunak

No.	Nama Perangkat	Spesifikasi
1	*System* Operasi	Windows 8 Profesional 32 bit
2	JDK (*Java*Development Kit)	JDK 1.7.0
3	JRE (*Java*Runtime Environment)	JRE 7
4	*Eclipse*	Juno
5	ADT	22.2.1

Sedangkan kebutuhan Perangkat keras minimal untuk menggunakan aplikasi *steganography* ini yaitu seperti pada tabel 3.32

Tabel 3.32 Kebutuhan Perangkat Lunak

No.	Nama Perangkat	Spesifikasi
1	*System* Operasi	*Android* 4.0 Ice Cream Sandwich
2	*Memory*	1GB
3	Monitor	240 x 320 *pixel*s
4	*Memory* Card	1 GB

3.1.8 Analisis Spesifikasi dan Kebutuhan Perangkat Lunak

Spesifikasi perangkat lunak adalah analisis spesifikasi prangkat lunak yang akan dibangun. Spesifikasi perangkat lunak dibutuhkan untuk menetapkan layanan apa yang akan dituntut pada operasi dan pengembangan perangkat lunak. Berikut spesifikasi perangkat lunak yang akan dibangun:

1. Aplikasi dapat melakukan proses *encode/embedding* dan *decode/extracting* pesan.
2. Aplikasi dapat melakukan proses *encrypt* dan *decrypt* pesan.
3. Aplikasi dapat me*generate private key* dan *public key* untuk *encrypt* dan *decrypt* pesan.

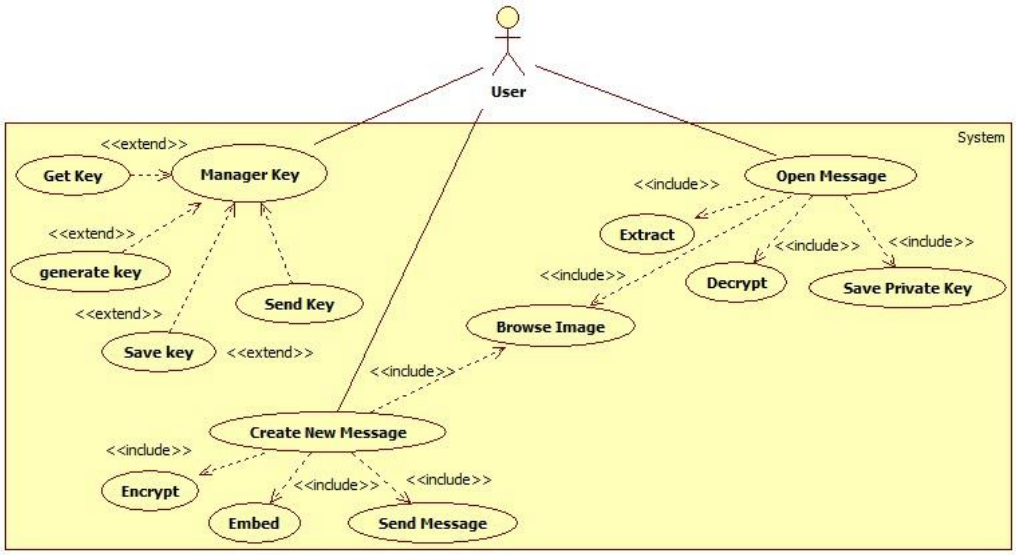
4. Aplikasi dapat melakukan pengiriman pesan.
5. Pesan yang dikirmkan berupa *stego image*, karena pesan yag dikirimkan berupa *image* maka layanan yang digunakan untuk mengirimkan pesan yaitu layanan MMS.
6. Dalam aplikasi terdapat *inbox key*.
7. Ketika *extracting* pesan, *user* tidak harus memasukan *private key* karena *private key* sudah otomatis masuk menurut nomor telepon *user*.

3.1.9 Analisis Kebutuhan Fungsional

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

3.1.9.1 *Use Case* Diagram

Use case diagram digunakan untuk mengetahui apa saja yang dapat dilakukan oleh *user*/aktor terhadap fungsionalitas yang terdapat pada aplikasi yang dibangun. *Use case* diagram pada aplikasi *steganography* terlihat pada gambar 3.7



Gambar 3.7 Use Case Diagram Aplikasi Steganography

3.1.9.2 Scenario *Use Case*

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

1. *Scenario Use Case Manager Key*

Scenario use case manager key dapat dilihat pada tabel 3.33

Tabel 3.33 *Scenario Use Case Manager Key*

Use case name	*Manager Key.*
Actor	*User*
Use case description	Digunakan untuk membangkitkan kunci, mengirim kunci yang telah dibangkitkan kepada *user*, dan menyimpan kunci yang telah dibangkitkan ke dalam *database*.
Precondition	*System* menampilkan *form Manager Key*
Trigger	*User* menekan *button* Ambil Kunci. *User* menakan *button* Bangkitkan Kunci. *User* menekan *button* Simpan Kunci. *User* menekan *button* kirim kunci.
Scenario :	
1. *User* memasukan nomor telepon *receiver*. 2. *System* menampilkan *list* kontak telepon. 3. *User* menekan *button* ambil kunci jika nomor *sender* sudah memiliki kunci yang telah dibangkitkan. 4. *User* menekan *button* bangkitkan kunci jika nomor *receiver* belum	

memiliki kunci yang telah dibangkitkan. 5. *User* menekan *button* simpan kunci. 6. *System* menyimpan kunci kedalam *database*. 7. *User* menekan *button* kirim kunci. 8. *System* mengirimkan kunci pada *receiver*.
Exception :

2. *Scenario Use Case Get Key*

Scenario use case get key dapat dilihat pada tabel 3.34

Tabel 3.34 *Scenario Use Case Get Key*

Use case name	*Get Key.*
Actor	*User.*
Use case description	Digunakan untuk mengambil kunci yang sudah tersimpan di *database*.
Precondition	*System* menampilkan *form manager key.*
Trigger	*User* menekan *button Get Key*.
Scenario :	
1. *User* menekan *button manager key* di menu utama. 2. *System* menampilkan *form manager key*. 3. *User* memasukan nomor telepon *receiver*. 4. *User* menekan *button Get Key*.	
Exception :	
1. *User* menekan *button back, Get Key* dibatalkan.	

3. *Scenario Use Case Generate Key*

Scenario use case generate key dapat dilihat pada tabel 3.35

Tabel 3.35 *Scenario Use Case Generate Key*

Use case name	*Generate Key.*
Actor	*User.*
Use case description	Digunakan untuk membangkitkan pasangan kunci.
Precondition	*System* menampilkan *form manager key.*
Trigger	*User* menekan *button Generate Key*.
Scenario :	
1. *User* menekan *button manager key* di menu utama. 2. *System* menampilkan *form manager key*. 3. *User* memasukan nomor telepon *receiver*. 4. *User* menekan *button Generate Key.*	
Exception :	
1. *User* menekan *button back, Generate Key* dibatalkan.	

4. *Scenario Use Case Save Key*

Scenario use case save key dapat dilihat pada tabel 3.36

Tabel 3.36 *Scenario Use Case Save Key*

Use case name	*Save Key.*
Actor	*User.*
Use case description	Digunakan untuk menyimpan kunci yang telah dibangkitkan ke dalam *database*.
Precondition	*System* menampilkan *form manager key.*
Trigger	*User* menekan *button Save Key*.
Scenario :	
1. *User* menekan *button manager key* di menu utama. 2. *System* menampilkan *form manager key*. 3. *User* memasukan nomor telepon *receiver*. 4. *User* menekan *button Save Key*.	
Exception :	
1. *User* menekan *button back, Save Key* dibatalkan.	

5. *Scenario Use Case Send Key*

Scenario use case send key dapat dilihat pada tabel 3.37

Tabel 3.37 *Scenario Use Case Send Key*

Use case name	*Send Key.*
Actor	*User.*
Use case description	Digunakan untuk mengirimkan *private key* kepada *receiver* yang akan disimpan kedalam *database*.
Precondition	*System* menampilkan *form manager key.*
Trigger	*User* menekan *button Send Key*.
Scenario :	
1. *User* menekan *button manager key* di menu utama. 2. *System* menampilkan *form manager key*. 3. *User* memasukan nomor telepon *receiver*. 4. *User* menekan *button Send Key*.	
Exception :	
1. *User* menekan *button back, Send Key* dibatalkan.	

6. *Scenario Use Case Create New Message*

Scenario use case create new message dapat dilihat pada tabel 3.38

Tabel 3.38 *Scenario Use Case Create New Message*

Use case name	*Create New Message.*
Actor	*User.*
Use case description	Digunakan untuk membuat pesan rahasia yang akan dikirimkan.
Precondition	*System* menampilkan *form Create New Message.*
Trigger	*User* menekan *button write secret message*.
Scenario :	
1. *User* menekan *button Write Secret Message.* 2. *System* menampilkan *form Create New Message.* 3. *User* menekan *button Browse Image.* 4. *System* menampilkan *galeri image.* 5. *User* memilih *image* yang akan digunakan. 6. *User* memasukan nomor telepon *receiver.* 7. *User* mengetikan pesan. 8. *User* menekan *button encode.* 9. *System* menampilkan info *encode*.	

Exception :
1. *User* menekan *button back, Write Secret Message* dibatalkan.

7. *Scenario Use Case Browse Image*

Scenario use case browse image dapat dilihat pada tabel 3.39

Tabel 3.39 *Scenario Use Case Browse Image*

Use case name	*Browse Image.*
Actor	*user.*
Use case description	Digunakan untuk membuka galeri dan memilih gambar.
Precondition	*System* menampilkan galeri.
Trigger	*user* menekan *button browse image.*
Scenario :	
1. *User* menekan *button Write Secret Message.* 2. *System* menampilkan *form Create New Message.* 3. *User* menekan *button Browse Image*. 4. *System* menampilkan *galeri image.* 5. *User* memilih *image* yang akan digunakan.	
Exception :	
1. *User* menekan *button back, browse image* dibatalkan.	

8. *Scenario Use Case Encrypt*

Scenario use case encrypt dapat dilihat pada tabel 3.40

Tabel 3.40 *Scenario Use Case Encrypt*

Use case name	*Encrypt.*
Actor	*User.*
Use case description	Digunakan untuk menyadikan pesan.
Precondition	*System* menampilkan form *Create New Message.*
Trigger	*User* menekan *button encode.*
Scenario :	
1. *User* menekan *button Write Secret Message.* 2. *System* menampilkan *form Create New Message.* 3. *User* menekan *button Browse Image*. 4. *System* menampilkan *galeri image.* 5. *User* memilih *image* yang akan digunakan. 6. *User* memasukan nomor telepon *receiver.* 7. *User* mengetikan pesan. 8. *User* menekan *button encode.*	
Exception :	
2. *User* menekan *button back, encode* dibatalkan.	

9. *Scenario Use Case Embed*

Scenario use case embed/*embedding* dapat dilihat pada tabel 3.41

Tabel 3.41 *Scenario Use Case Embed*

Use case name	*Embed.*
Actor	*User.*
Use case description	Digunakan untuk menyisipkan pesan ke dalam citra.
Precondition	*System* menampilkan menu *Write Secret Message*, *sender* telah memilih citra, *sender* memasukan informasi pesan, dan pesan telah ter-*encrypt*.
Trigger	*User* menekan *button encode*.
Scenario :	
1. *User* menekan *button Write Secret Message.* 2. *System* menampilkan *form Create New Message.* 3. *User* menekan *button Browse Image*. 4. *System* menampilkan *galeri image.*	

5. *User* memilih *image* yang akan digunakan. 6. *User* memasukan nomor telepon *receiver.* *7. User* mengetikan pesan. *8. User* menekan *button encode.* 9. *System* menampilkan info *encode*.
Exception :
1. *User* menekan *button back, encode* dibatalkan.

10. *Scenario Use Case Send Message*

Scenario use case send message dapat dilihat pada tabel 3.42

Tabel 3.42 *Scenario Use Case Send Message*

Use case name	*Send Message.*
Actor	*User.*
Use case description	Digunakan untuk mengirim pesan yang telah disisipkan ke dalam image.
Precondition	*System* menampilkan *form Write Secret Message.*
Trigger	*User* menekan *button Send It Now.*
Scenario :	
1. *User* menekan *button Write Secret Message.* 2. *System* menampilkan *form Create New Message.* 3. *User* menekan *button Browse Image*. 4. *System* menampilkan *galeri image.* 5. *User* memilih *image* yang akan digunakan. 6. *User* memasukan nomor telepon *receiver.* *7. User* mengetikan pesan. *8. User* menekan *button encode.* 9. *System* menampilkan info *encode*. 10. *User* menekan *button Send It Now.*	
Exception :	
1. *User* menekan *button back, Send Message* dibatalkan.	

11. *Scenario Use Case Open Message*

Scenario use case open message dapat dilihat pada tabel 3.43

Tabel 3.43 *Scenario Use Case Open Message*

Use case name	*Open Message.*
Actor	*User.*
Use case description	Digunakan untuk memilih pesan rahasia yang akan dibuka.
Precondition	*System* menampilkan *form Open Message.*
Trigger	*User* menekan *button Read Secret Message*.
Scenario :	
1. *User* menekan *button Read Secret Message.* 2. *System* menampilkan *form Open Message.* 3. *User* menekan *button Browse Image*. 4. *System* menampilkan *galeri image.* 5. *User* memilih *stego image* yang akan diungkap. 6. *User* memasukan nomor telepon *receiver.* *7. User* menekan *button decode.* 8. *System* menampilkan info *decode*.	
Exception :	
1. *User* menekan *button back, Open Message* dibatalkan.	

12. *Scenario Use Case Save Private Key*

Scenario use case save private key dapat dilihat pada tabel 3.44

Tabel 3.44 *Scenario Use Case Save Private Key*

Use case name	*Save Private Key.*
Actor	*User.*
Use case description	Digunakan untuk menyimpan *private key* yang dikirimkan oleh *sender* kepada *receiver* dan akan disimpan ke database.
Precondition	*System* menampilkan *form inbox key.*
Trigger	*User* menekan *inbox key*.
Scenario :	
1. *User* menekan *button Inbox Key*. 2. *System* menampilkan *form list Inbox.* 3. *User* memilih pesan yang berisikan *private key* untuk disimpan. 4. *System* menampilkan *form detail inbox key*.	

5. *User* menekan *button Save Key.*
Exception :
1. *Receiver* menekan *button back, Save Key* dibatalkan.

13. *Scenario Use Case Extract*

*Scenario use case extract/*pengungkapan dapat dilihat pada tabel 3.45

Tabel 3.45 *Scenario Use Case Extract*

Use case name	*Extract*.
Actor	*User.*
Use case description	Digunakan untuk *extracting* pesan *stego image*.
Precondition	*System* menampilkan *form Open Message*.
Trigger	*User* menekan *button decode.*
Scenario :	
1. *User* menekan *button Read Secret Message.* 2. *System* menampilkan *form Open Message.* 3. *User* menekan *button Browse Image*. 4. *System* menampilkan *galeri image.* 5. *User* memilih *image* yang akan di*extract*. 6. *User* memasukan nomor telepon *sender.* *7. User* menekan *button decode.*	
Exception :	
1. *User* menekan *button back, extract* dibatalkan.	

14. *Scenario Use Case Decrypt*

Scenario use case Decrypt dapat dilihat pada tabel 3.46

Tabel 3.46 Scenario *Use Case Decrypt*

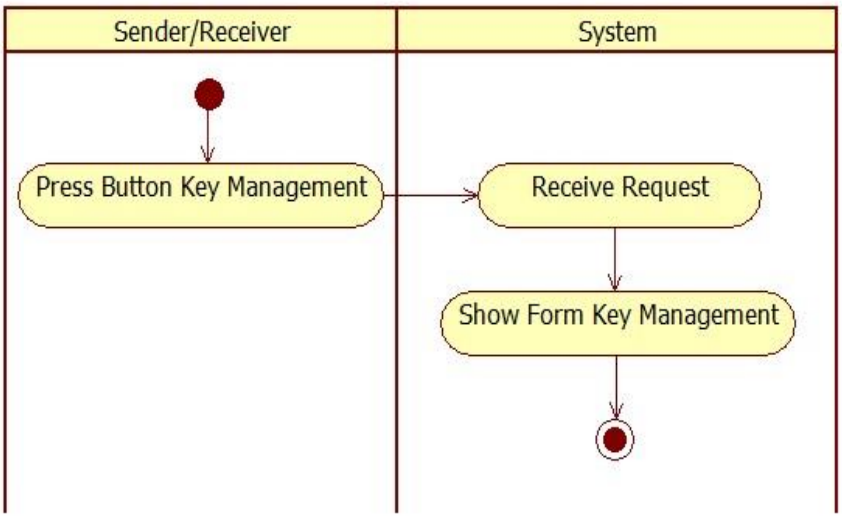
Use case name	*Decrypt.*
Actor	*User.*
Use case description	Digunakan untuk men*decrypt* pesan yang terdapat pada *stego image.*
Precondition	*System* menampilkan pesan yang akan di *decrypt*.
Trigger	*User* menekan *button decode.*
Scenario :	
1. *User* menekan *button Read Secret Message.* 2. *System* menampilkan *form Open Message.* 3. *User* menekan *button Browse Image*. 4. *System* menampilkan *galeri image.* 5. *User* memilih *image* yang akan di*extract*. 6. *User* memasukan nomor telepon *sender.* *7. User* menekan *button decode.* *8. System* menampilkan info *decode*.	
Exception :	
1. *User* menekan *button back, decrypt* dibatalkan.	

3.1.9.3 Activity Diagram

Activity Diagram menggambarkan berbagai alir aktivitas dalam *system* 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 *Manager Key***

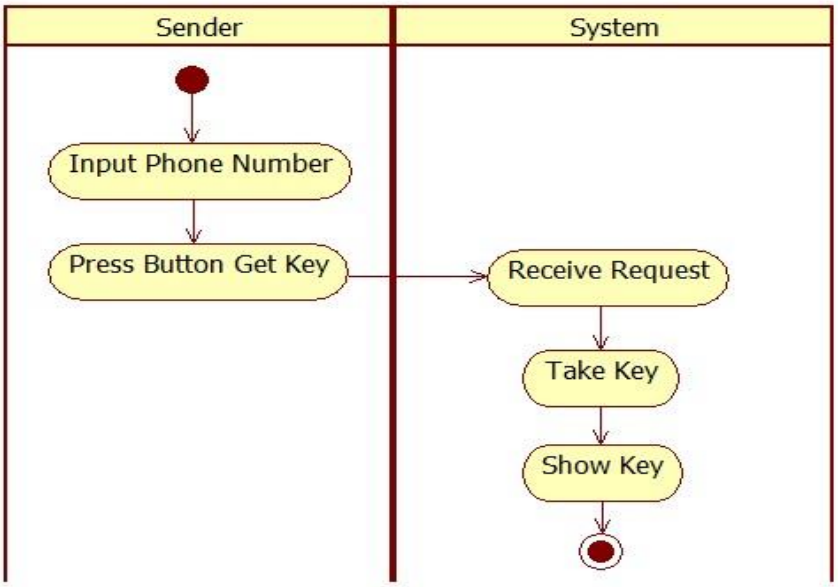
Activity diagram *key management* atau *manager key* menggambarkan alur aktivitas membangkitkan sepasang kunci antara *sender* dengan *system* seperti terlihat pada gambar 3.8



Gambar 3.8 Activity Diagram *Manager Key*

2. Activity Diagram *Get Key*

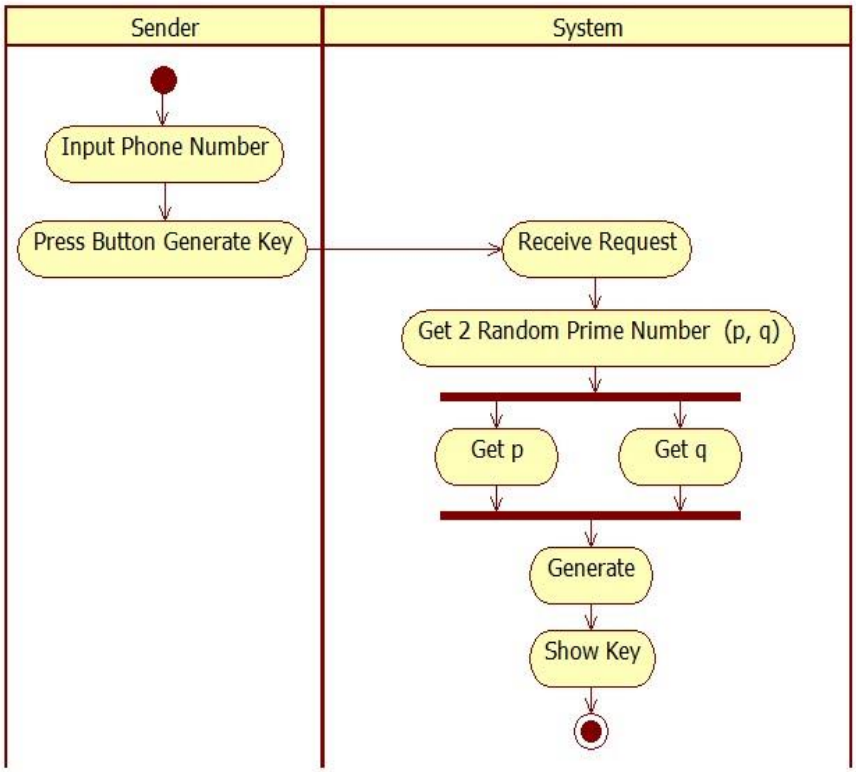
Activity diagram *get key* mengambarkan alur aktivitas mengambil sepasang kunci yang tersimpan di dalam *database* seperti terlihat pada gambar 3.9



Gambar 3.9 Activity Diagram *Get Key*

3. Activity Diagram *Generate Key*

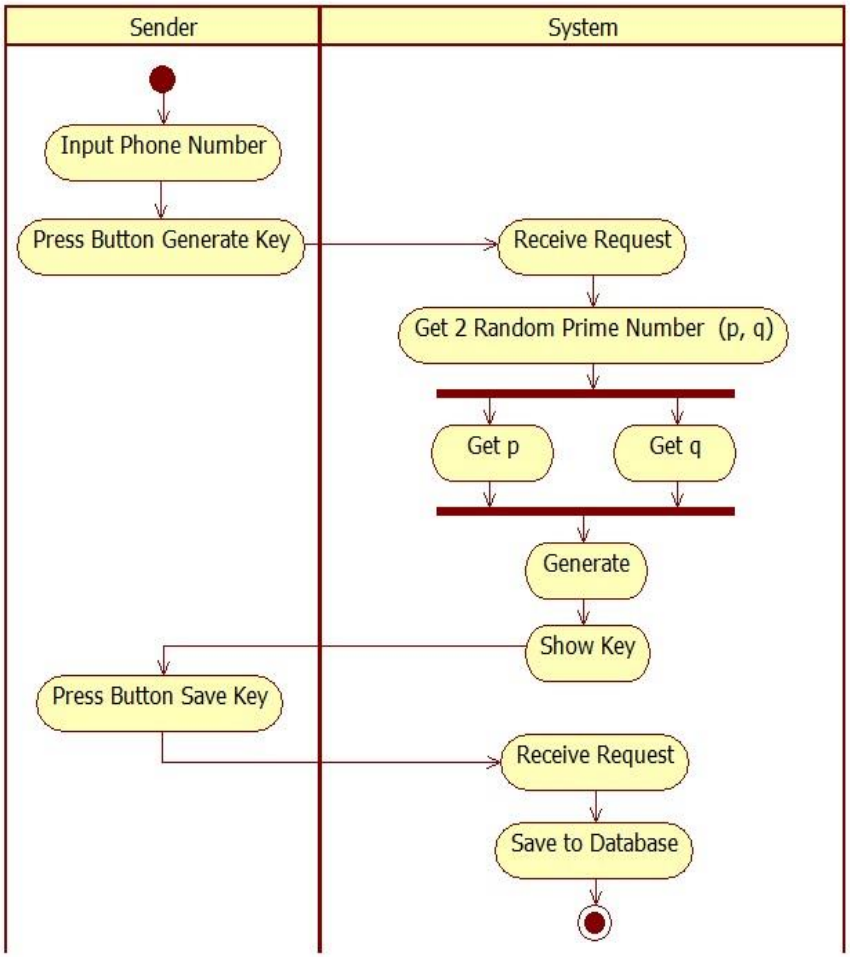
Activity diagram *generate key* mengambarkan alur aktivitas membangkitkan sepasang kunci dengan mangambil 2 bilangan prima besar secara acak seperti terlihat pada gambar 3.10



Gambar 3.10 Activity Diagram *Generate Key*

4 Activity Diagram *Save Key*

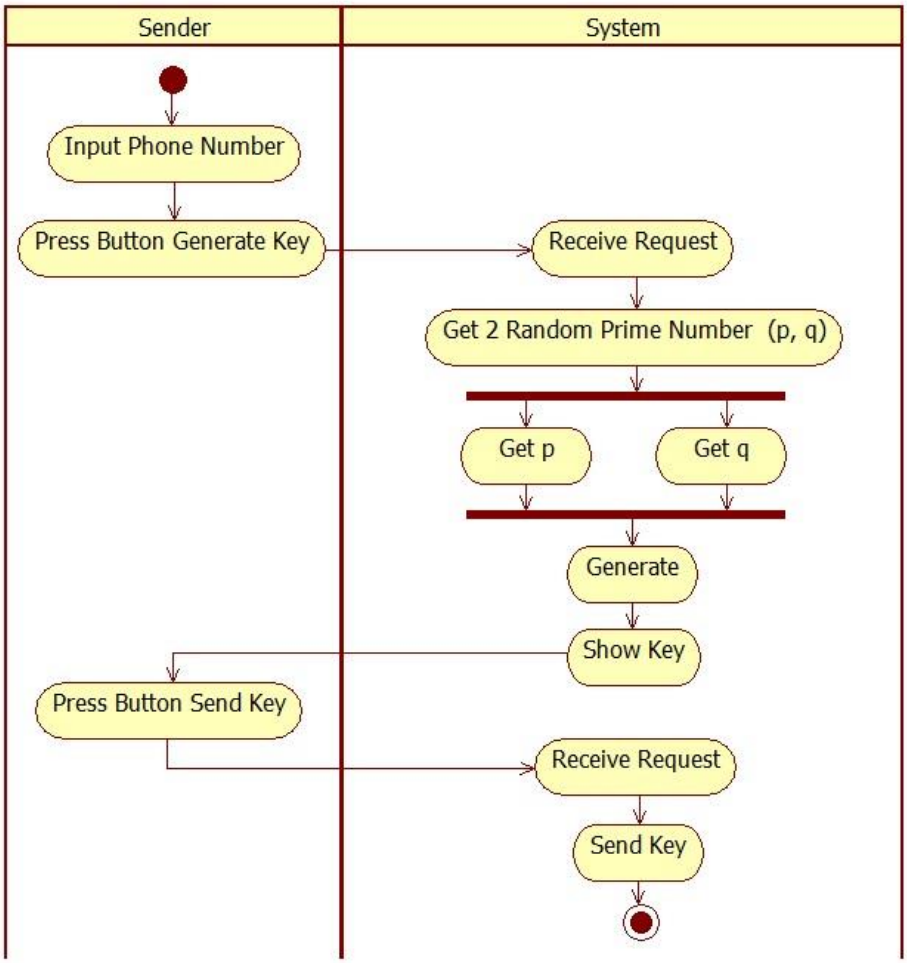
Activity diagram *save key* menggambarkan alur aktivitas menyimpan kunci yang telah dibangkitkan ke dalam database seperti terlihat pada gambar 3.11



Gambar 3.11 Activity Diagram *Save Key*

5 Activity Diagram *Send Key*

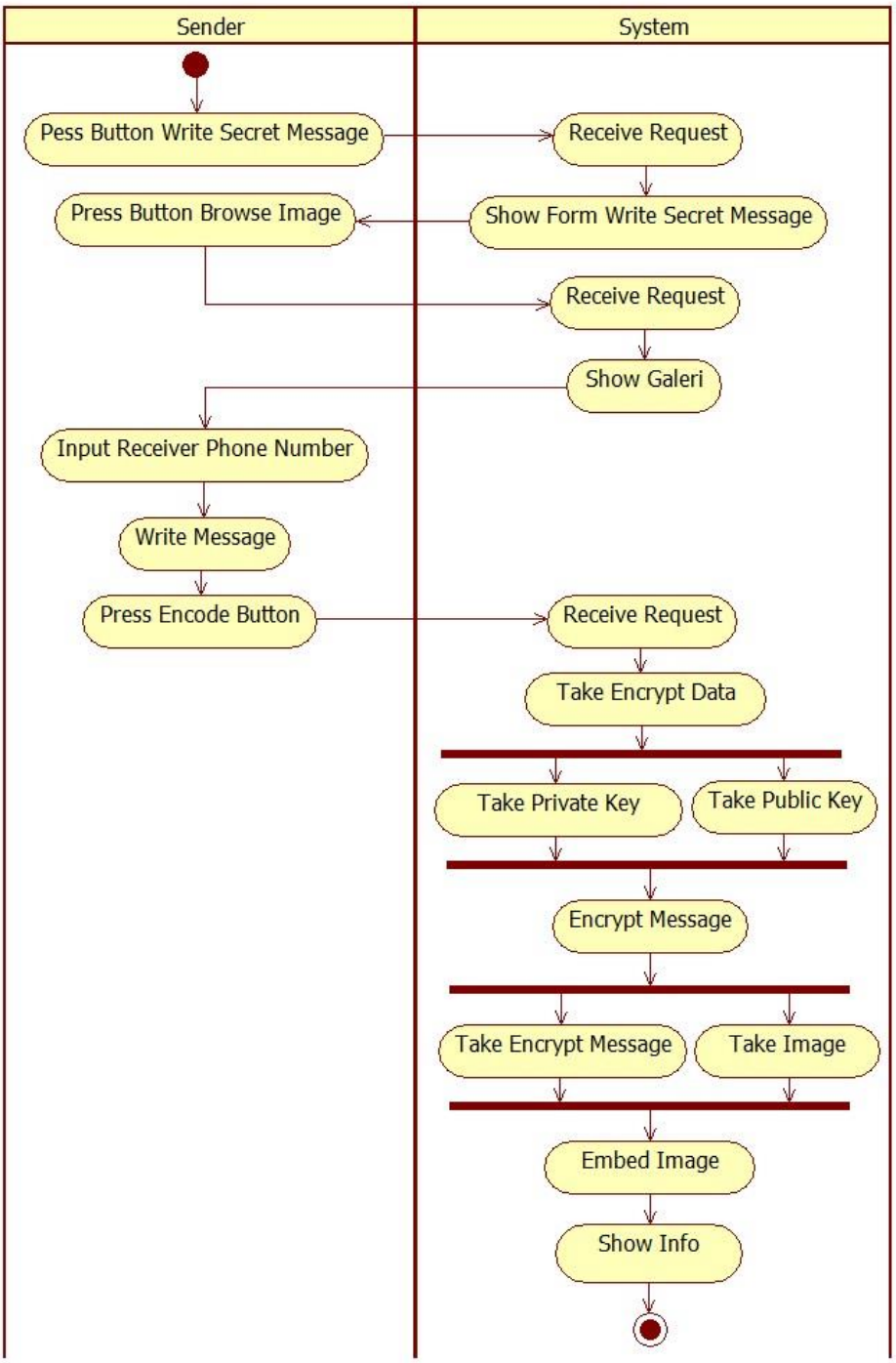
Activity diagram *send key* menggambarkan alur aktivitas mengirim pesan yang telah di *encrypt* kepada *receiver* seperti terlihat pada gambar 3.12



Gambar 3.12 Activity Diagram *Send Key*

6 Activity Diagram *Create New Message*

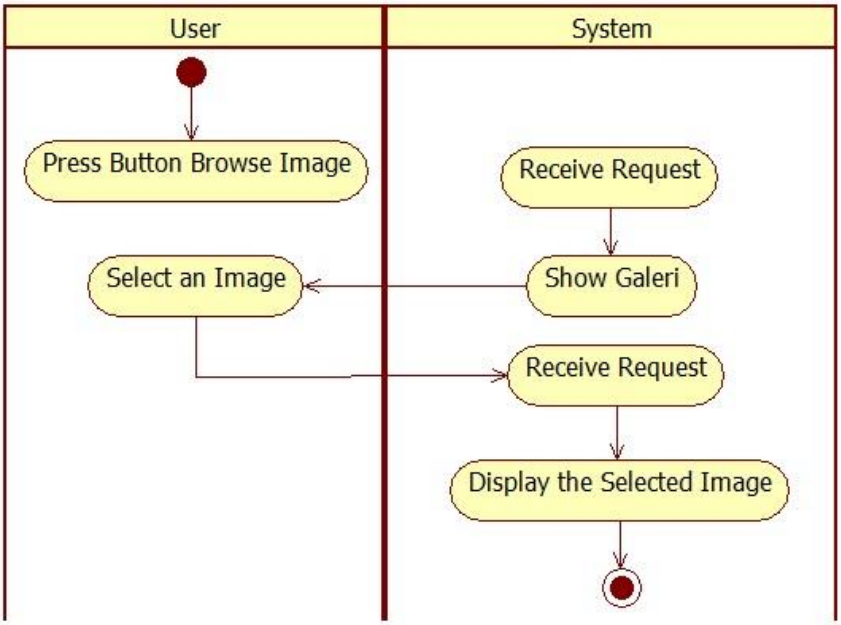
Activity diagram c*reate new message* menggambarkan alur aktivitas membuat pesan baru antara *sender* dengan *system* seperti terlihat pada gambar 3.13



Gambar 3.13 Activity Diagram *Create New Message*

7 Activity Diagram *Browse Image*

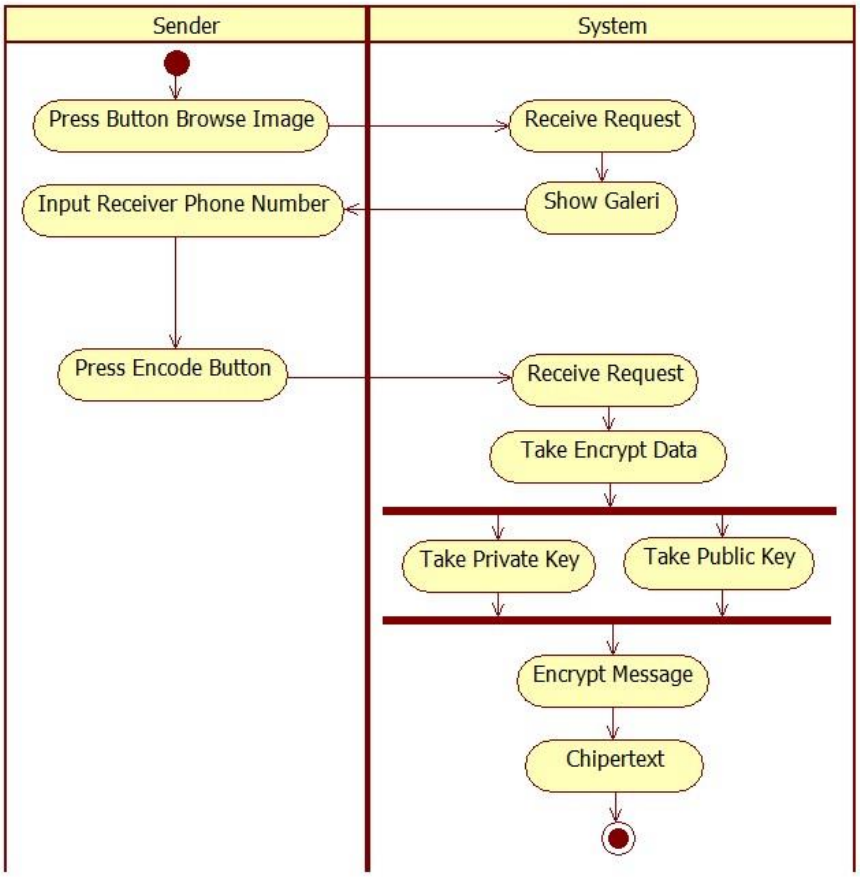
Activity diagram *browse image* menggambarkan alur aktivitas membuka galeri dan memilih gambar *user* dengan *system* seperti terlihat pada gambar 3.14



Gambar 3.14 Activity Diagram *Browse Image*

8 Activity Diagram *Encrypt*

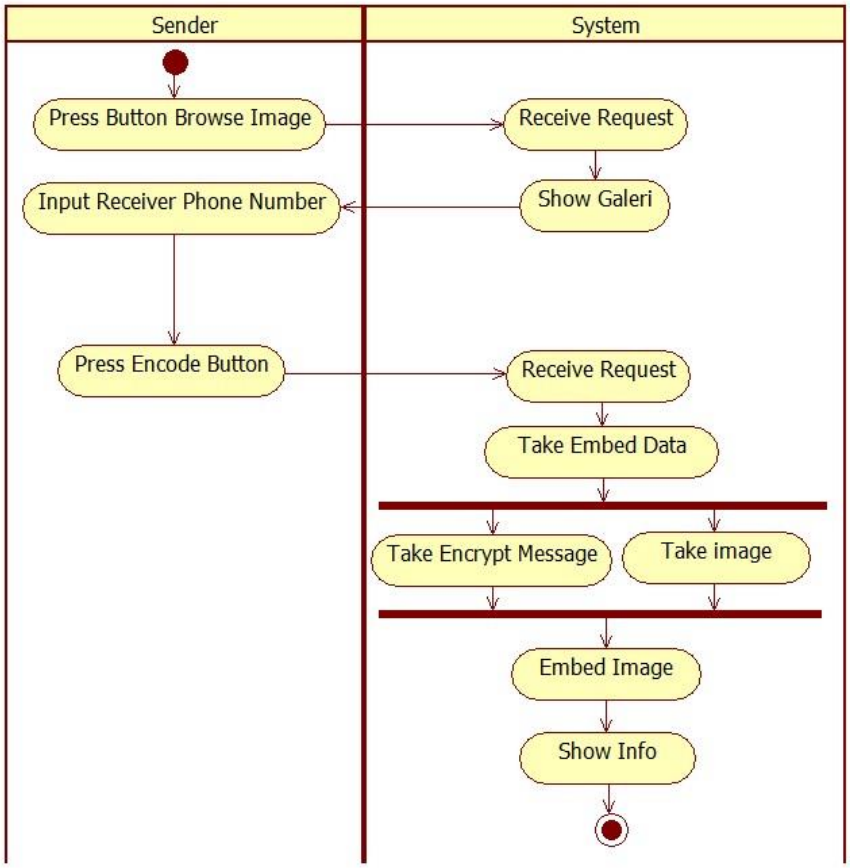
Activity diagram *encrypt* menggambarkan alur aktivitas *encrypt* pesan antara *sender* dengan *system* seperti terlihat pada gambar 3.15



Gambar 3.15 Activity Diagram *Encrypt*

9 Activity Diagram *Embed*

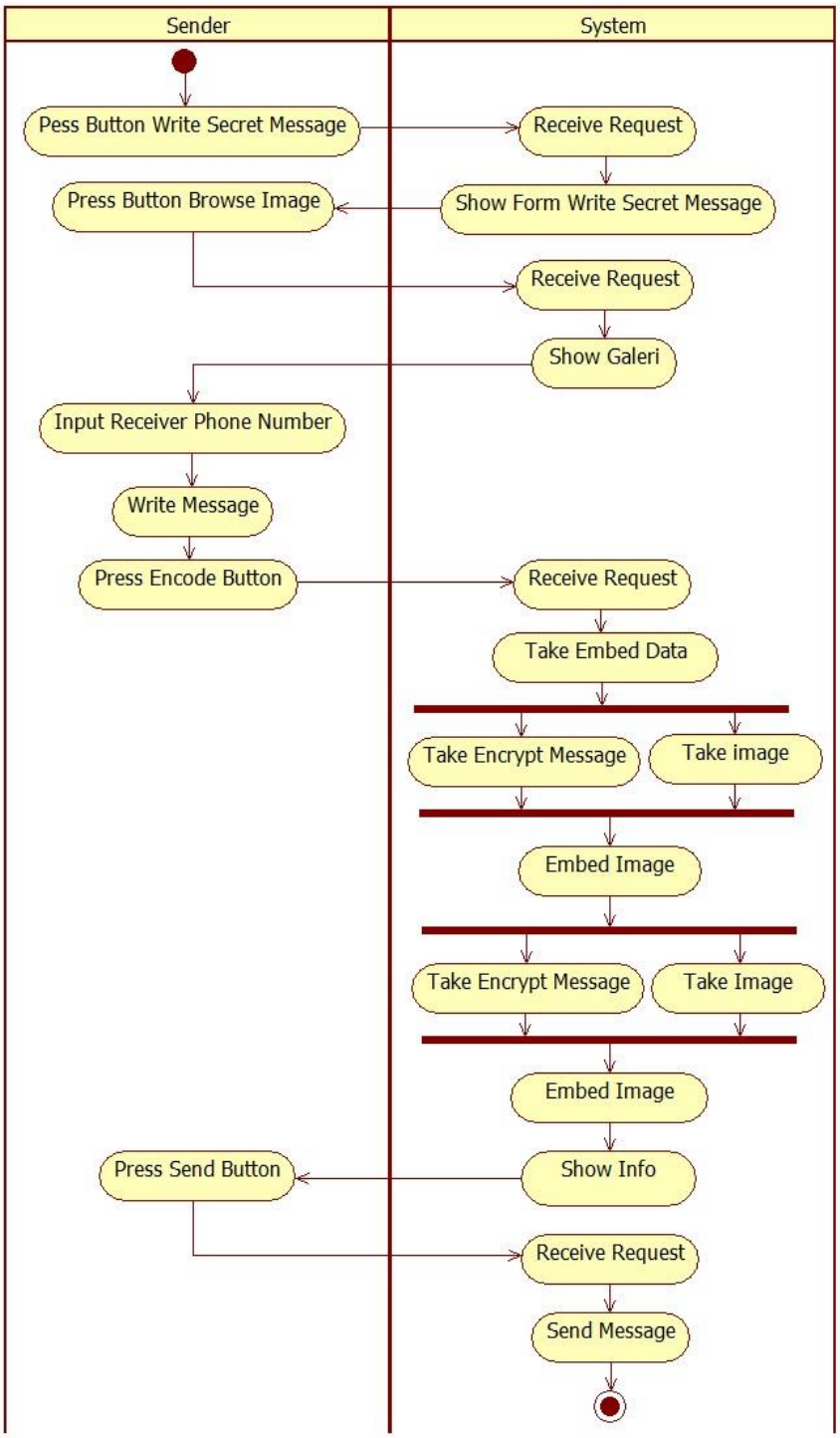
Activity diagram *embed* menggambarkan alir aktivitas *embed/embedding* yang dilakukan antara *sender* dengan *system* seperti terlihat pada gambar 3.16



Gambar 3.16 Activity Diagram *Embed*

10 Activity Diagram *Send Message*

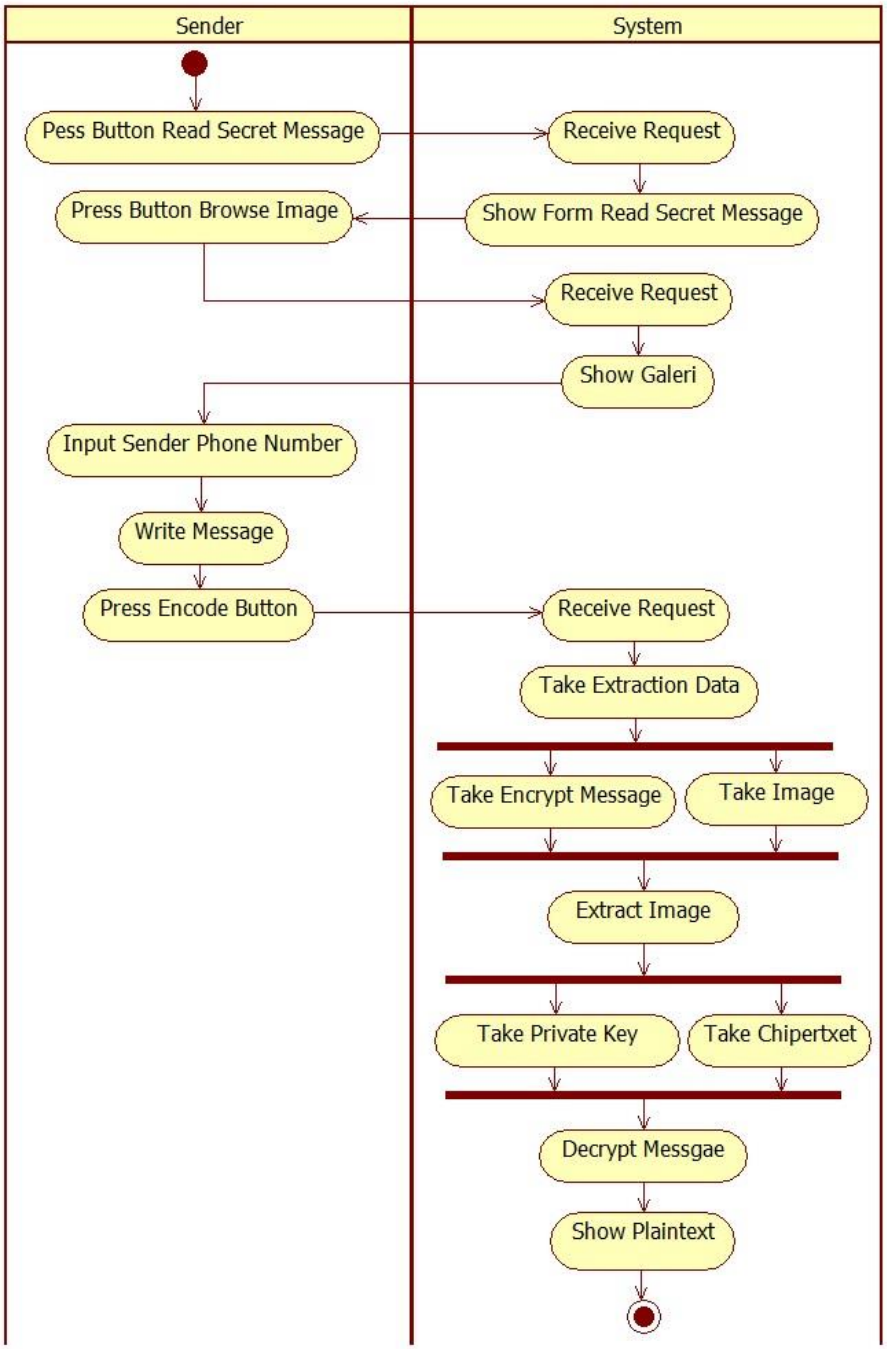
Activity diagram *send message* menggambarkan alur aktivitas *send message* pesan antara *sender* dengan *system* seperti terlihat pada gambar 3.17



Gambar 3.17 Activity Diagram *Send Message*

11 Activity Diagram *Open Message*

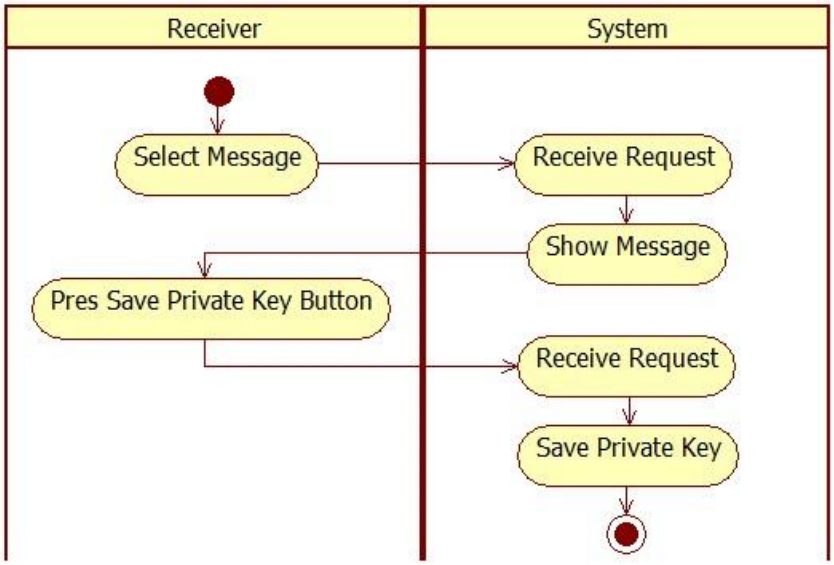
Activity diagram *open message* menggambarkan alur aktivitas membuka pesan yang akan di *decode* antara *receiver* dengan *system* seperti terlihat pada gambar 3.18



Gambar 3.18 Activity Diagram *Open Message*

12 Activity Diagram *Save Private Key*

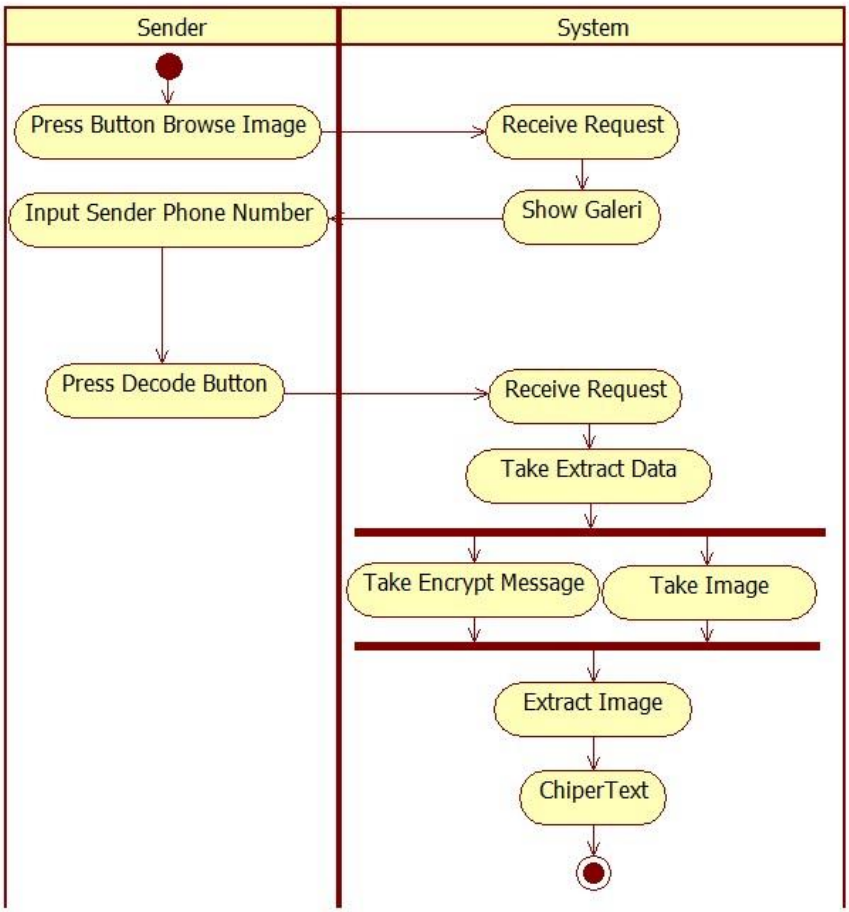
Activity diagram *save private key* menggambarkan alur aktivitas menyimpan *private key* yang dikirimkan antara *sender* dengan *system* seperti terlihat pada gambar 3.19



Gambar 3.19 Activity Diagram *Save Private Key*

13 Activity Diagram *Extract*

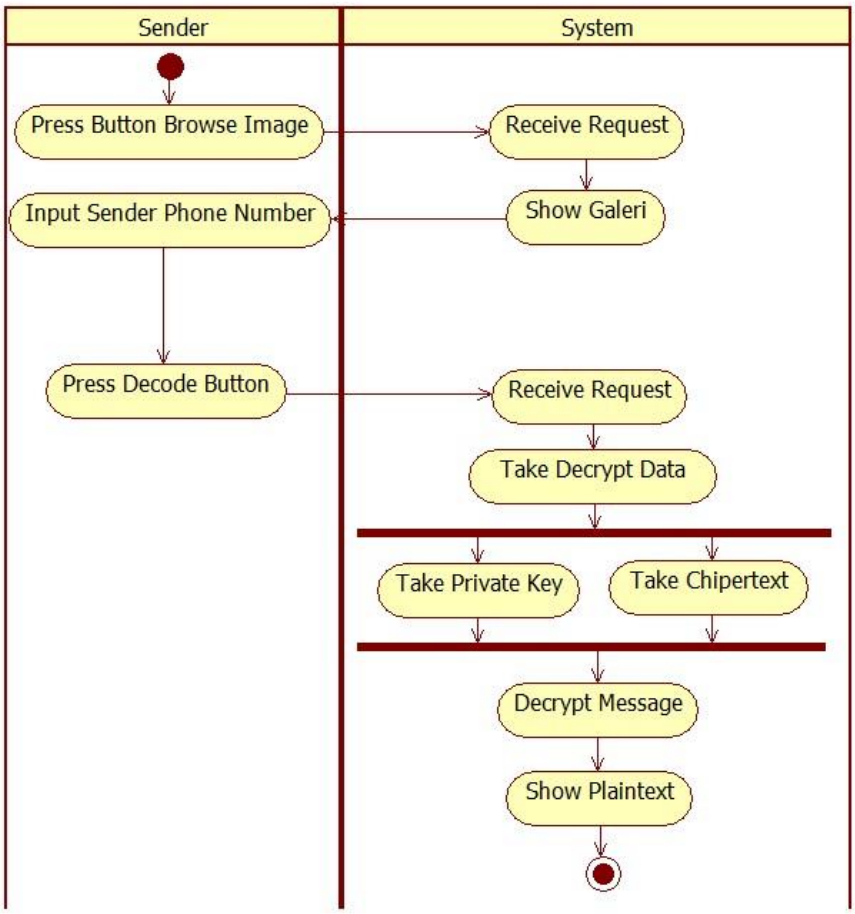
Activity diagram *extract* menggambarkan alur aktivitas *extract* pesan antara *receiver* dengan *system* seperti terlihat pada gambar 3.20



Gambar 3.20 Activity Diagram *Extract*

14 Activity Diagram Decrypt

Activity diagram *decrypt* menggambarkan alur aktivitas *decrypt* pesan antara *receiver* dengan *system* seperti terlihat pada gambar 3.21



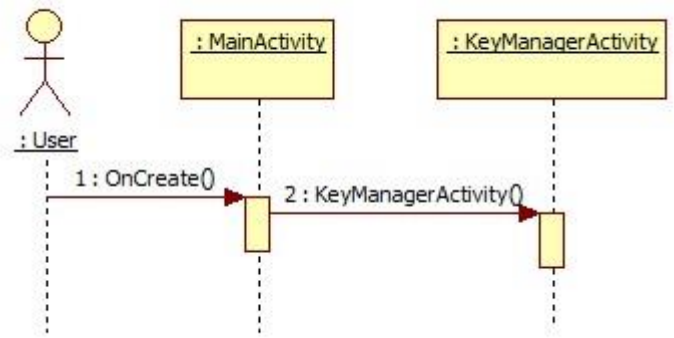
Gambar 3.21 Activity Diagram *Decrypt*

3.1.9.4 Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar *System* yang digambarkan terhadap waktu. *Sequence* diagram yang terdapat pada aplikasi *steganography* yaitu sebagai berikut:

1. **Sequence Diagram *Manager Key***

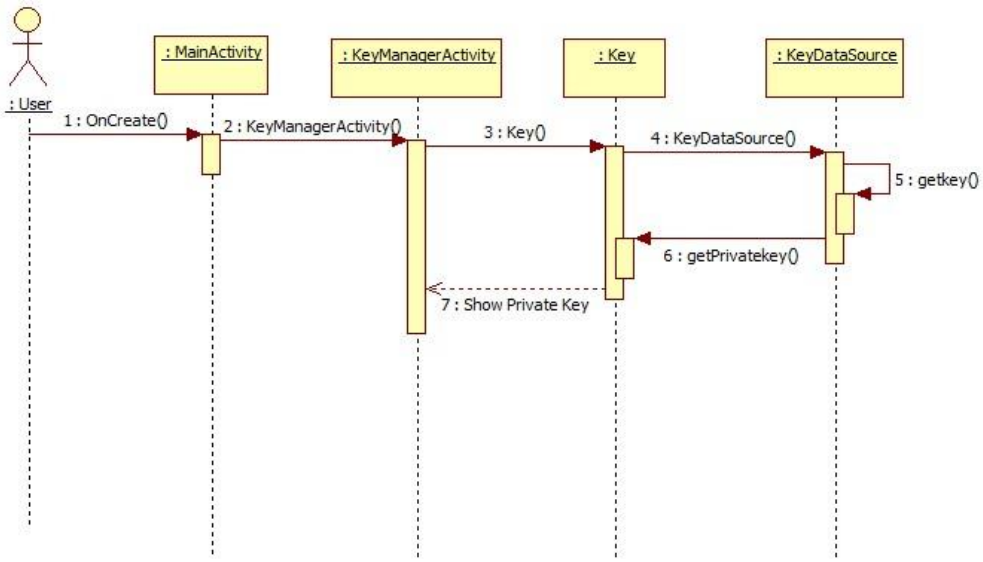
Sequence diagram *key management* atau *manager key* merupakan diagram yang menggambarkan interaksi yang terjadi dalam *system* antara *user* dengan *system* untuk masuk ke dalam *key management*. *Sequence* diagram *key management* aplikasi *steganography* terlihat pada gambar 3.22



Gambar 3.22 Sequence Diagram *Manager Key*

2. Sequence Diagram *Get Key*

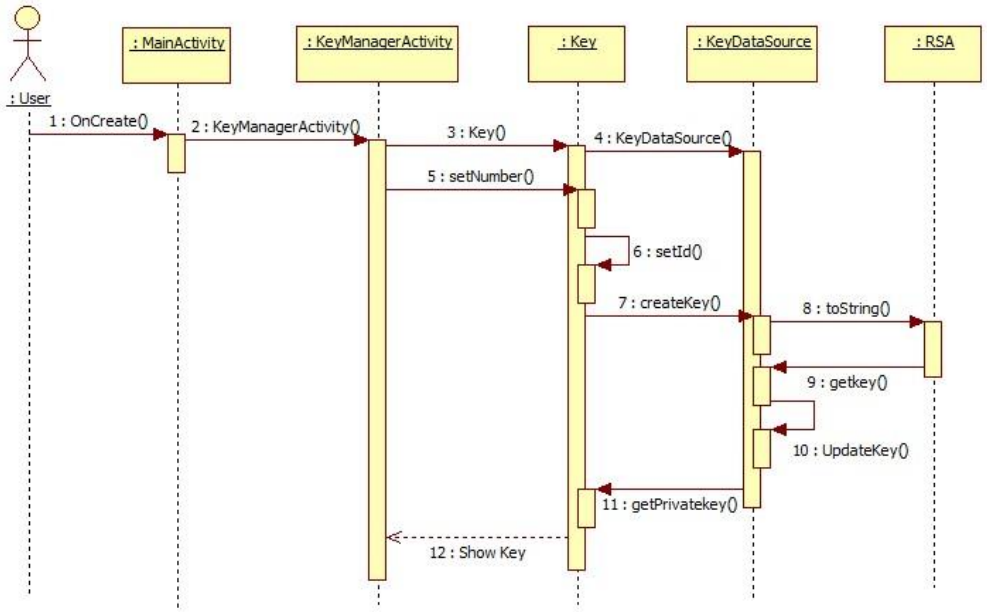
Sequence diagram *get key* merupakan diagram yang menggambarkan interaksi yang terjadi dalam *system* antara *user* dengan *system* untuk mengambil kunci yang tersimpan di dalam *database*. *Sequence* diagram *get key* aplikasi *steganography* terlihat pada gambar 3.23



Gambar 3.23 Sequence Diagram *Get Key*

3. Sequrnce Diagram *Generate Key*

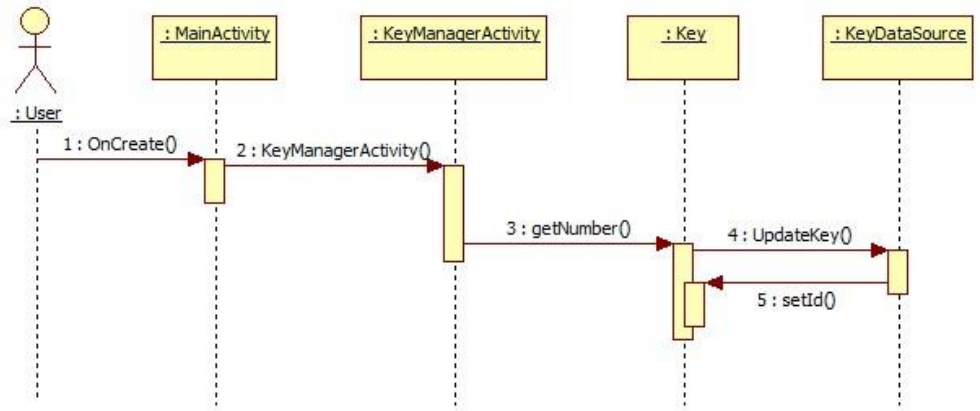
Sequence diagram *generate key* merupakan diagram yang menggambarkan interaksi yang terjadi dalam *system* antara *user* dengan *system* untuk membangkitkan kunci. *Sequence* diagram *generate key* aplikasi *steganography* terlihat pada gambar 3.24



Gambar 3.24 Sequence Diagram *Generate Key*

4. Sequence Diagram *Save Key*

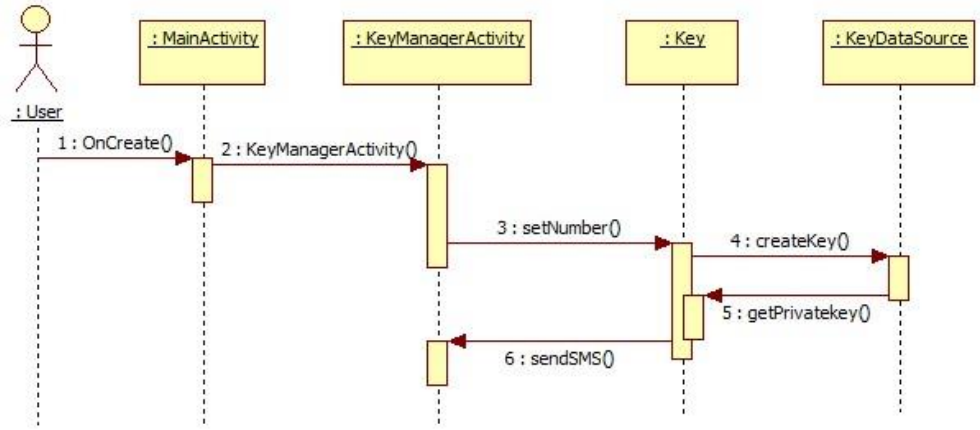
Sequence diagram *save key* merupakan diagram yang menggambarkan interaksi yang terjadi dalam *system* antara *user* dengan *system* untuk menyimpan kunci yang telah dibangkitkan. *Sequence* diagram *save key* aplikasi *steganography* terlihat pada gambar 3.25



Gambar 3.25 Sequence Diagram *Save Key*

5. Sequence Diagram *Send Key*

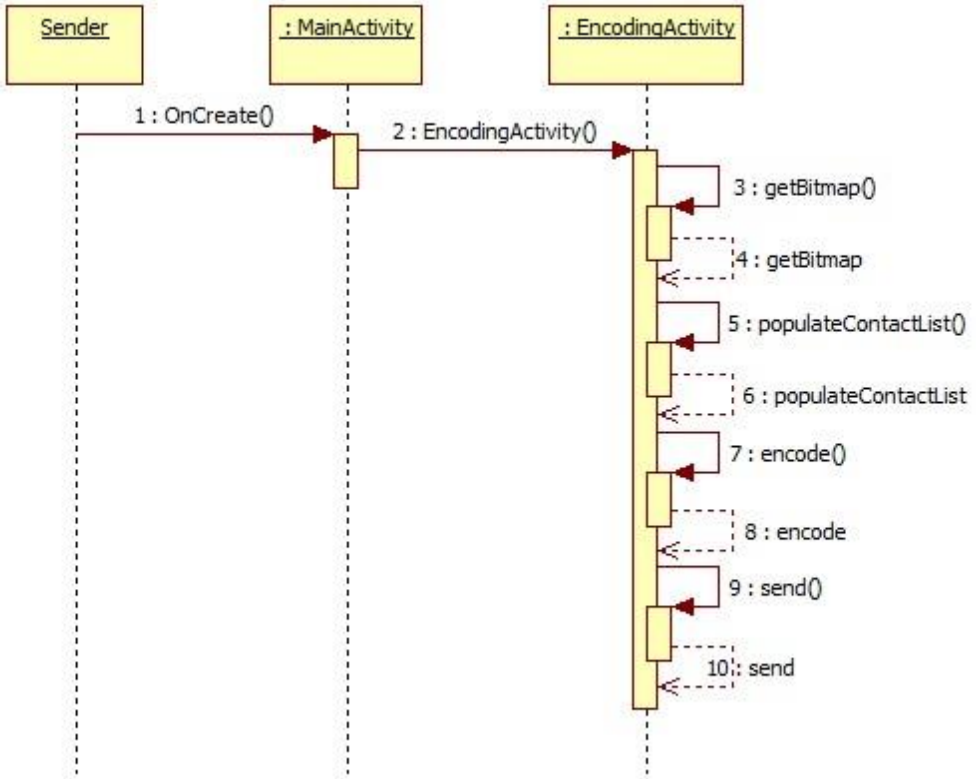
Sequence diagram *send key* merupakan diagram yang menggambarkan interaksi yang terjadi dalam *system* antara *user* dengan *system* untuk mengirimkan kunci. *Sequence* diagram *send key* aplikasi *steganography* terlihat pada gambar 3.26



Gambar 3.26 Sequence Diagram *Send Key*

6. Sequence Diagram *Create New Message*

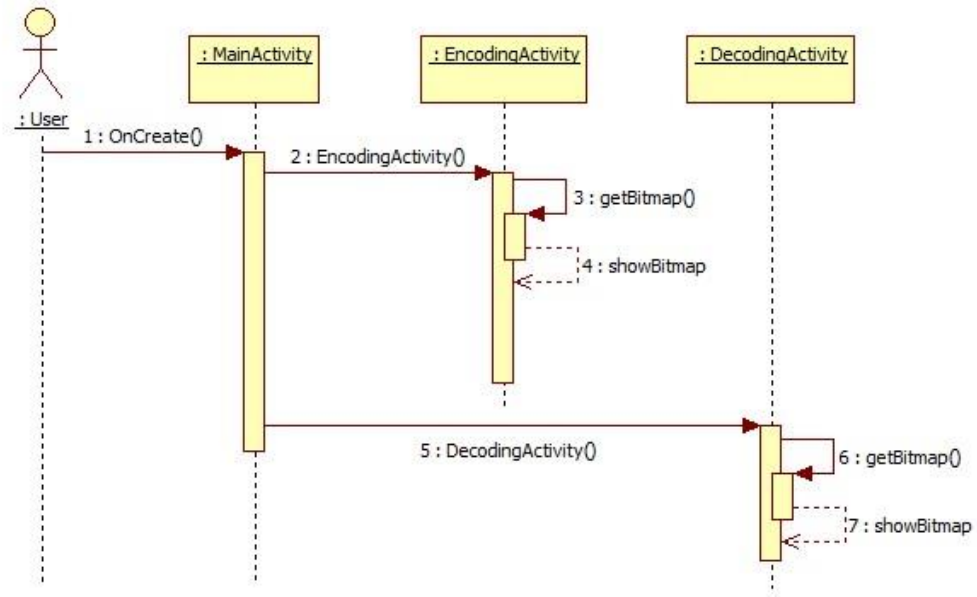
Sequence diagram *get key* merupakan diagram yang menggambarkan interaksi yang terjadi dalam *system* antara *user* dengan *system*. *Sequence* diagram *create new message* aplikasi *steganography* terlihat pada gambar 3.27



Gambar 3.27 Sequence Diagram *Create New Message*

7. Sequence Diagram *Browse Image*

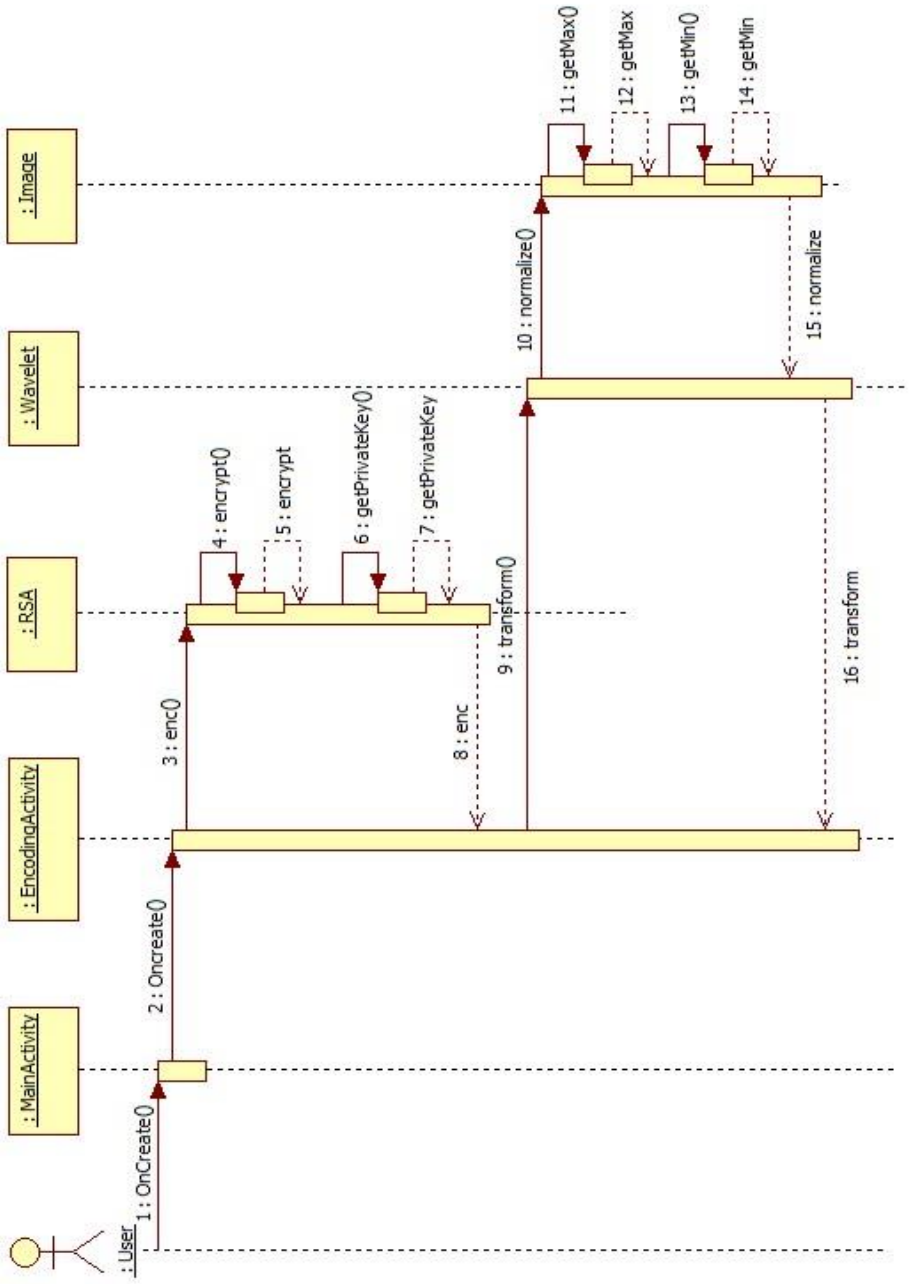
Sequence diagram *browse image* merupakan diagram yang menggambarkan interaksi yang terjadi dalam *system* antara *user* dengan *system*. *Sequence* diagram *browse image* aplikasi *steganography* terlihat pada gambar 3.28



Gambar 3.28 Sequence Diagram *Brwose Image*

8. Sequence Diagram *Encode*

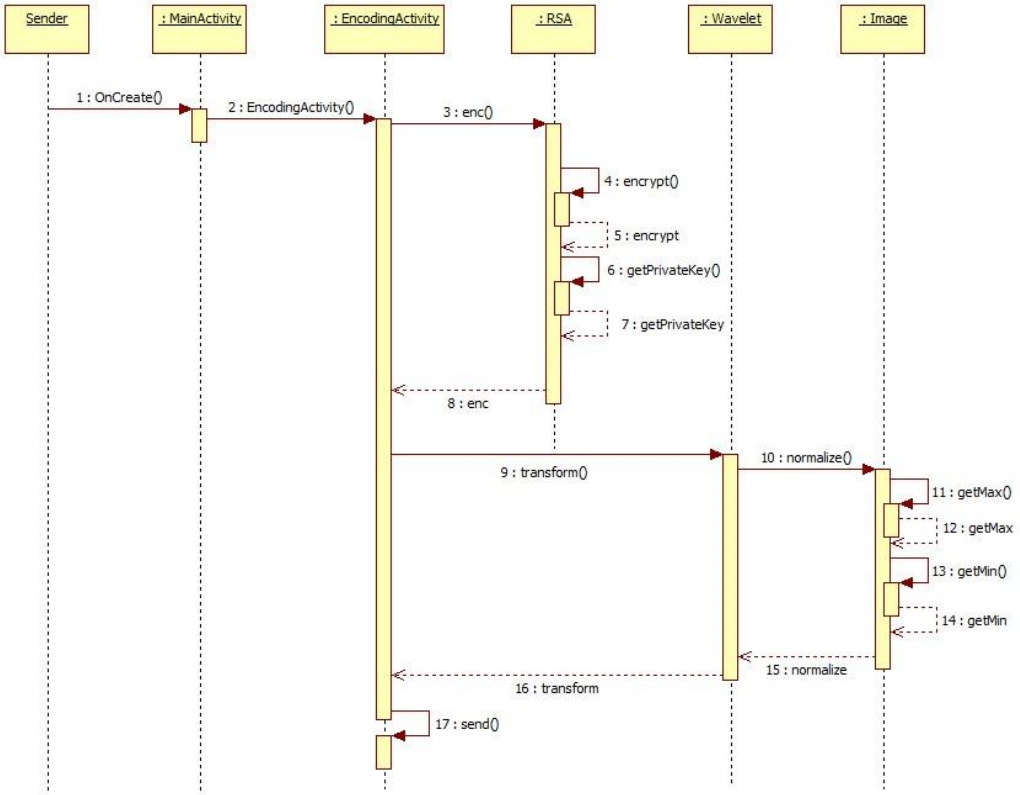
Sequence diagram *encode* merupakan diagram yang menggambarkan interaksi yang terjadi didalam *system* antara *sender* dengan *system* dalam *encrypt* dan *embed*. *Sequence* diagram aplikasi *steganography* terlihat seperti pada gambar 3.29



Gambar 3.29 Sequence Diagram *Encode*

9. Sequence Diagram *Send Message*

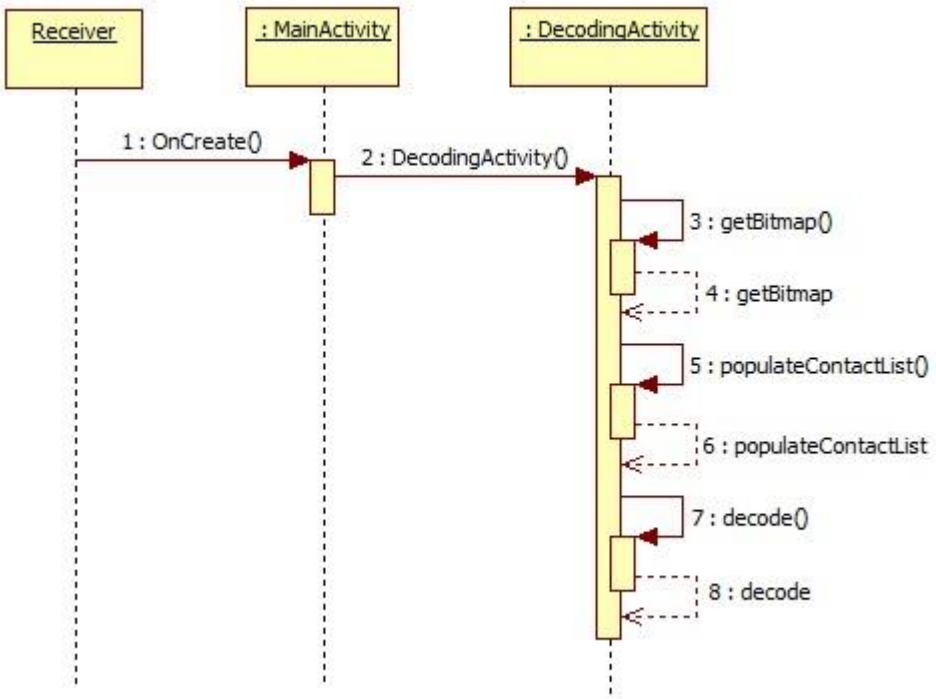
Sequence diagram *send message* merupakan diagram yang mengambarkan interaksi yang terjadi didalam *system* antara *sender* dengan *system* dalam mengirimkan pesan yang telah di *encrypt*. *Sequence* diagram *send message* aplikasi *Steganography* terlihat pada gambar 3.30



Gambar 3.30 Sequence Diagram *Send Message*

10. Sequence Diaram *Open Message*

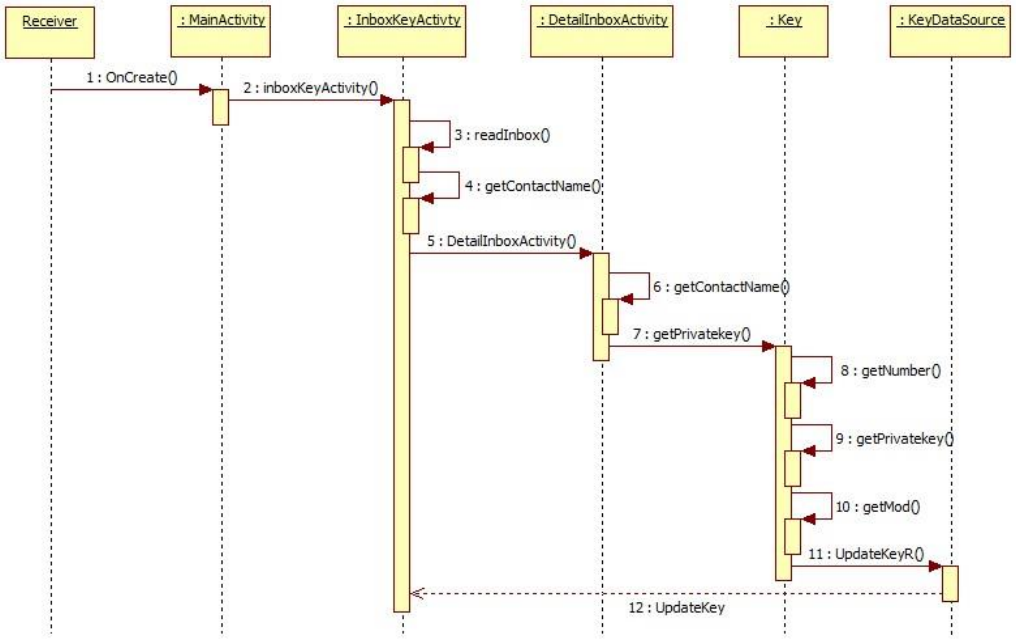
Sequence diagram *open message* merupakan diagram yang mengambarkan interaksi yang terjadi didalam *system* antara *receiver* dengan *system* dalam membuka pesan. *Sequence* diagram *open message* aplikasi *Steganography* terlihat seperti pada gambar 3.31



Gambar 3.31 Sequence Diagram *Open Message*

11. Sequence Diagram *Save Private Key*

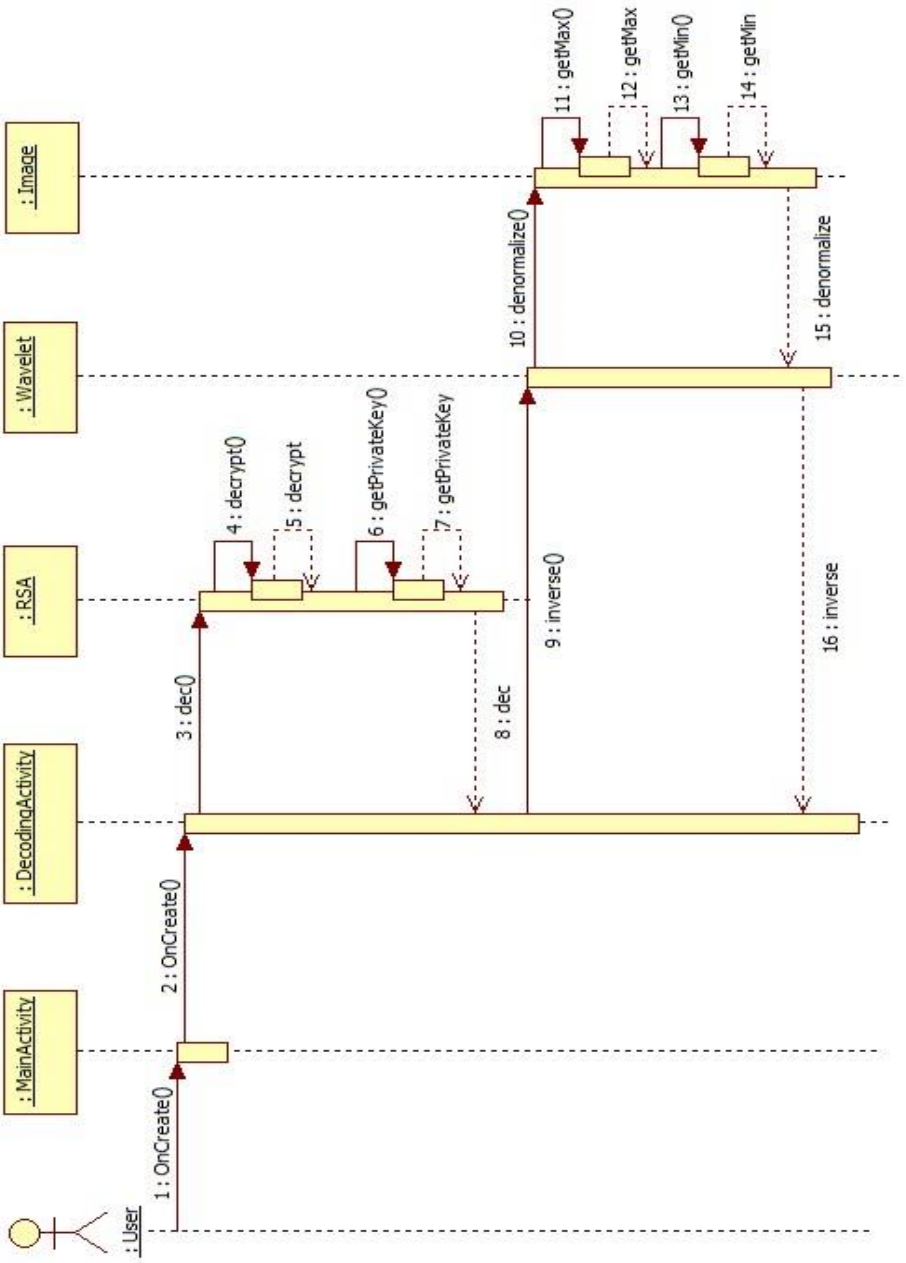
Sequence diagram *save private key* merupakan diagram yang mengambarkan interaksi yang terjadi didalam *system* antara *receiver* dengan *system* dalam menyimpan kunci privat yang dikirimkan ke dalam *database*. *Sequence* diagram *open message* aplikasi *Steganography* terlihat seperti pada gambar 3.32



Gambar 3.32 Sequence Diagram *Save Private Key*

12. Sequence Diagram Decode.

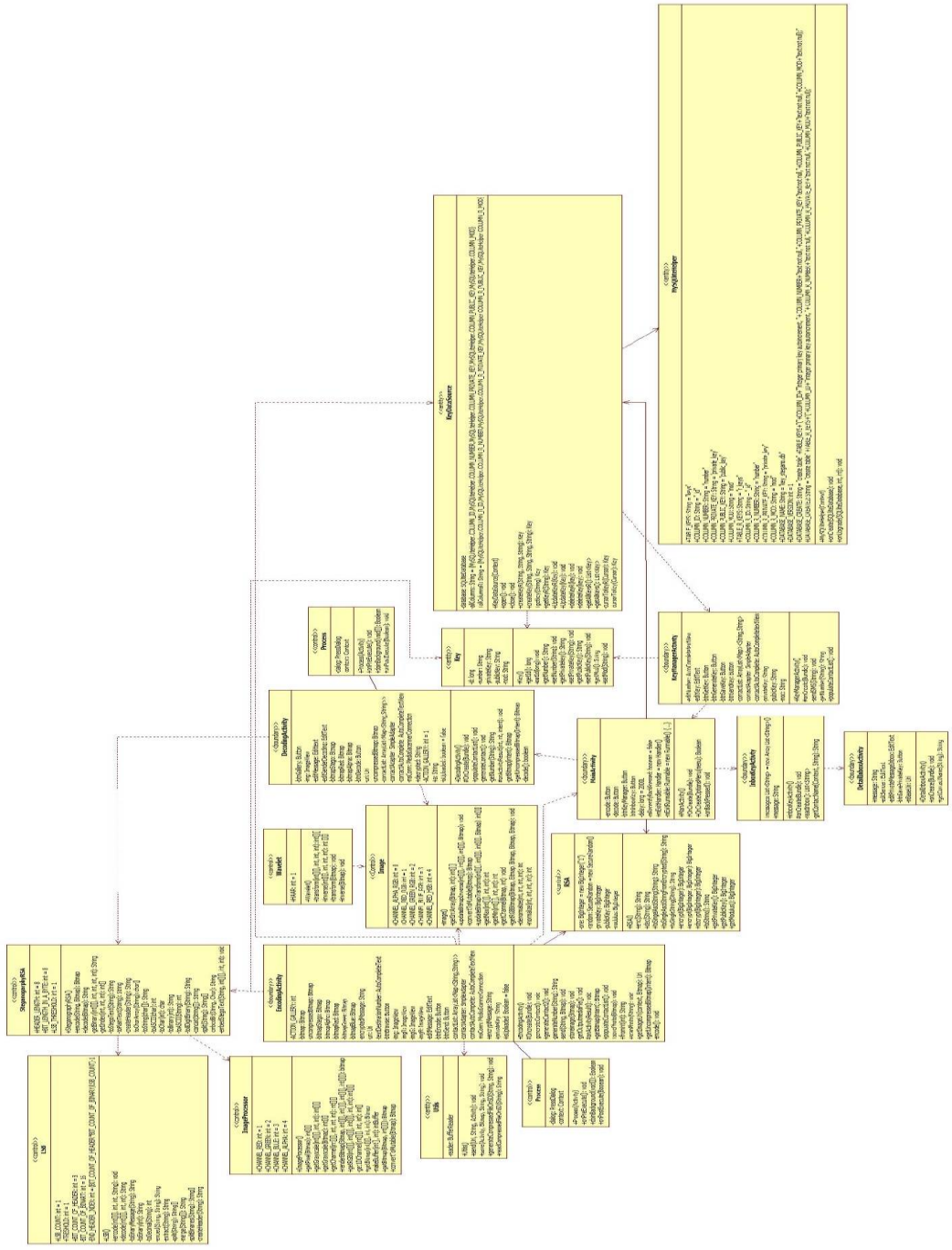
Sequence diagram *decode* merupakan diagram yang menggambarkan interaksi yang terjadi didalam *system* antara *receiver* dengan *system* dalam *extract* dan *decrypt*. *Sequence* diagram aplikasi *steganography* terlihat seperti pada gambar 3.33



Gambar 3.33 Sequence Diagram *Decode*

3.1.9.5 *Class* Diagram

Class Diagram digunakan untuk menggambarkan keadan suatu *system* dengan menjelaskan keterhubungan antara suatu *class* dengan *class* yang lain yang terdapat pada *system*. *class* diagram aplikasi *steganography* terlihat seperti pada gambar 3.34



Gambar 3.34 *Class* Diagram Aplikasi *Steganography*

Berikut jenis-jenis *class* diagram terlihat pada tabel 3.47

Tabel 3.47 Jenis *Class*

Nama Class	Jenis Class
MainActivity	*Boundary*
KeyManagerActivity	*Boundary*
EncodingActivity	*Boundary*
DecodingActivity	*Boundary*
InboxKeyActivity	*Boundary*
Process	*Control*
RSA	*Control*
Wavelet	*Control*
Image	*Control*
DetaiInboxActivity	*Boundary*
Key	*Entity*
KeyDataSource	*Entity*
MySQLliteHelper	*Entity*
SteganograpyRSA	*Control*
Utils	*Entity*
LSB	*Control*
ImageProcessor	*Control*

3.2 Perancangan Sistem

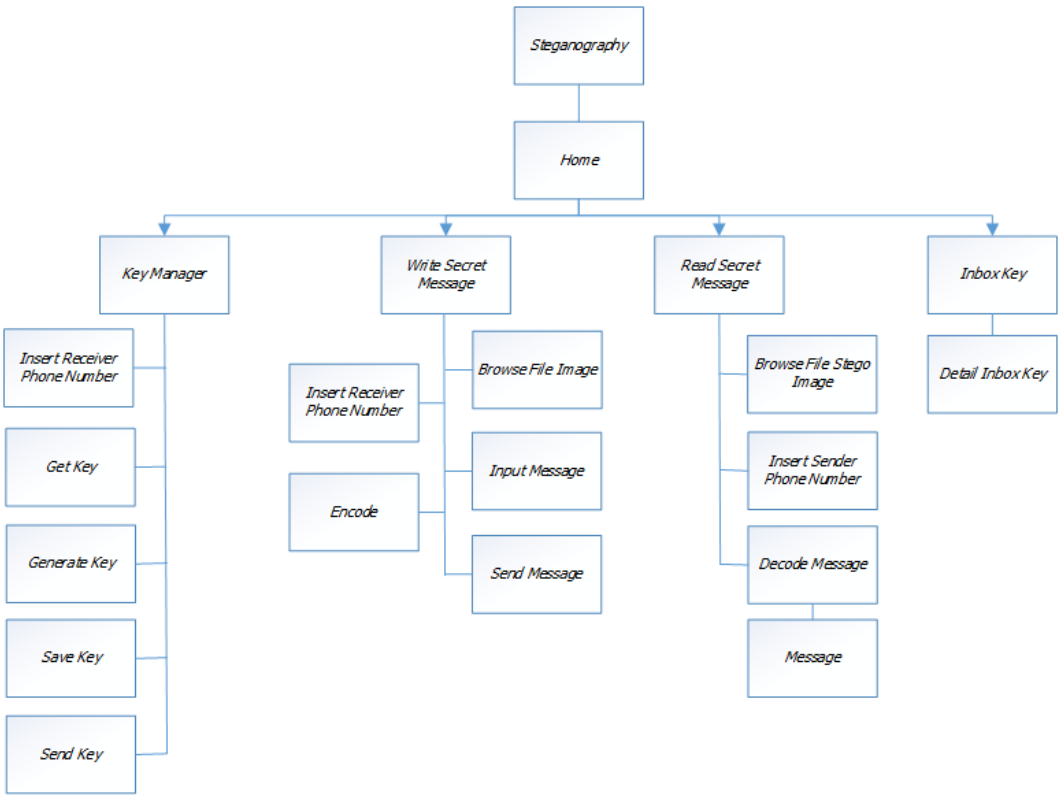
Perancangan sistem merupakan suatu proses yang menggambarkan bagaimana suatu sistem dibangun untuk memenuhi kebutuhan pada fase analisis. Tahap perancangan terdiri dari perancangan arsitektur.

3.2.1 Perancangan Arsitektur

Perancangan arsitektur terdiri dari perancangan struktur menu, perancangan antarmuka, perancangan pesan dan perancangan jaringan semantik.

3.2.1.1 Perancangan Struktur Menu

Perancangan menu dilakukan untuk mempermudah interaksi antara *system* dengan *user*. Gambar 3.35 merupakan struktur menu dari aplikasi *steganography* yang dibangun.



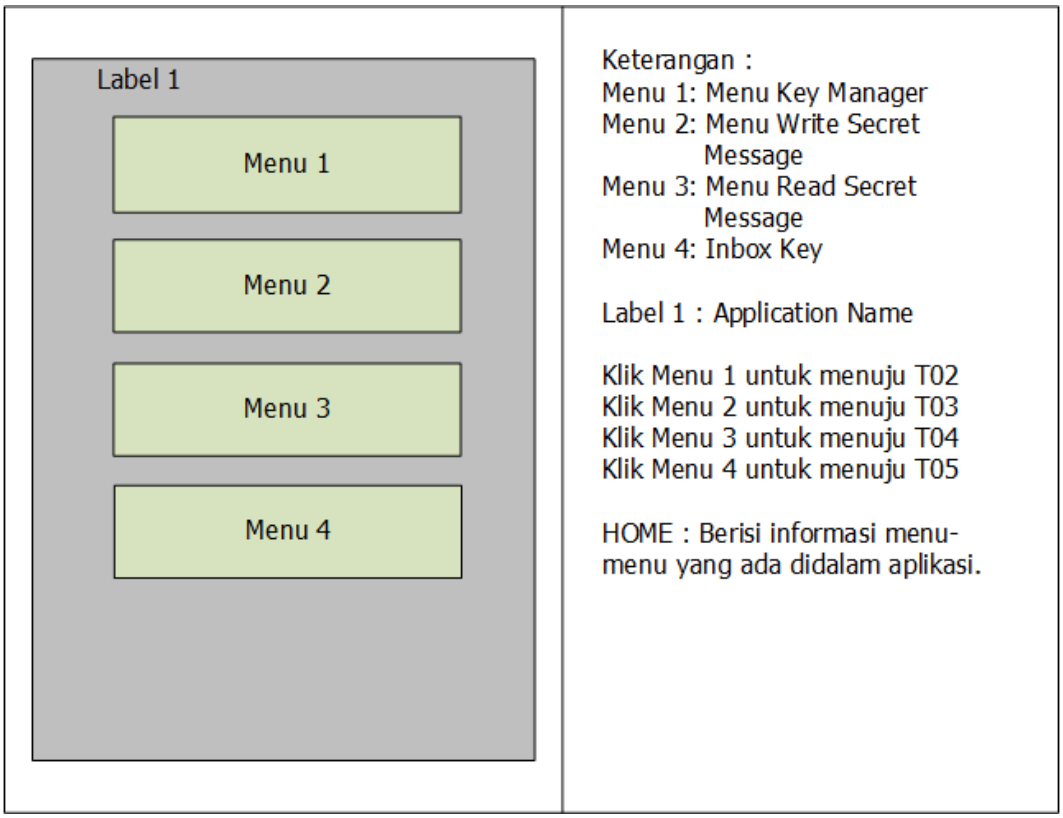
Gambar 3.35 Struktur Menu Aplikasi *Steganography*

3.2.1.2 Perancangan Antarmuka

Perancangan antarmuka adalah tahapan pembuatan rancangan antarmuka untuk digunakan pada pembangunan aplikasi *steganography* yang dibagi menjadi 5 bagian yaitu perancangan antarmuka *home*, *write secret message*, *read secret message*, *manager key* dan *inbox key*.

1. **Perancangan Antarmuka *Home* (T01)**

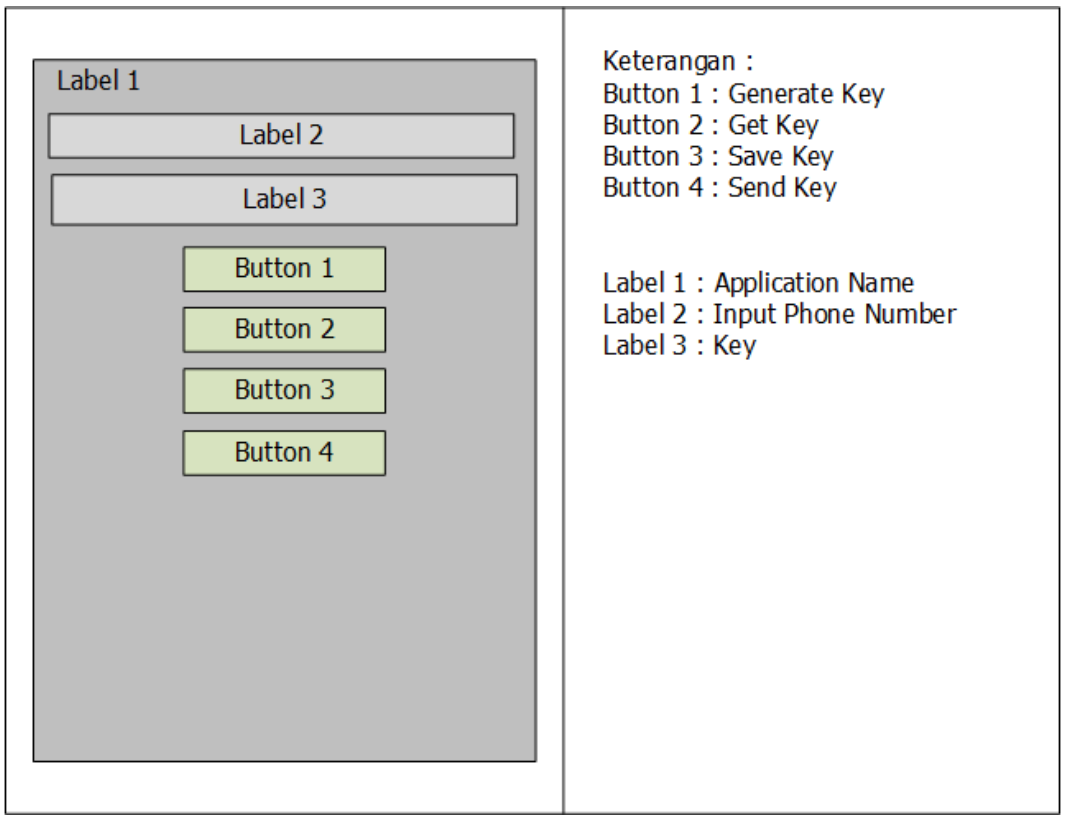
Perancangan antarmuka *home* dari aplikasi *steganography* ini terlihat pada gambar 3.36



Gambar 3.36 Perancangan Antarmuka *Home*

2. Perancangan Antarmuka *Manager Key* (T02)

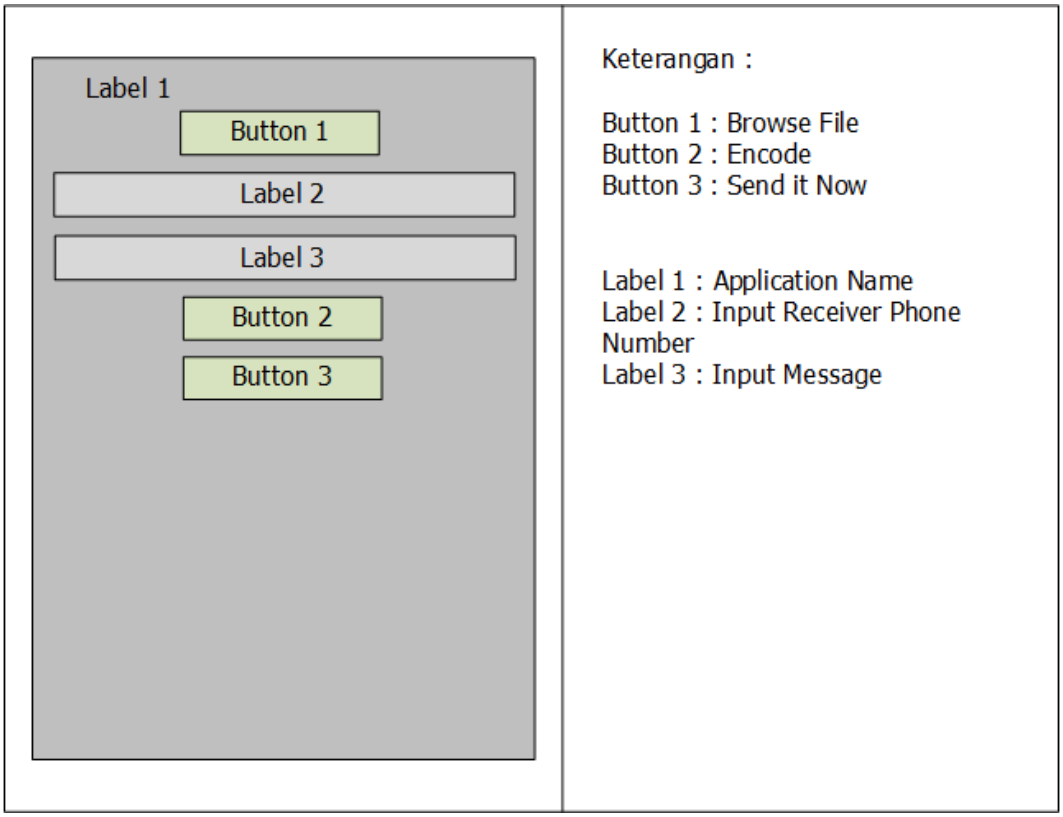
Perancangan antarmuka *Manager Key* merupakan tahapan rancangan antarmuka setelah tahapan menu *home* dipilih. Perancangan antarmuka *manager key* dari aplikasi *steganography* ini terlihat pada gambar 3.37



Gambar 3.37 Perancangan Antarmuka *Key Manager*

3. Perancangan Antarmuka *Write Secret Message* (T03)

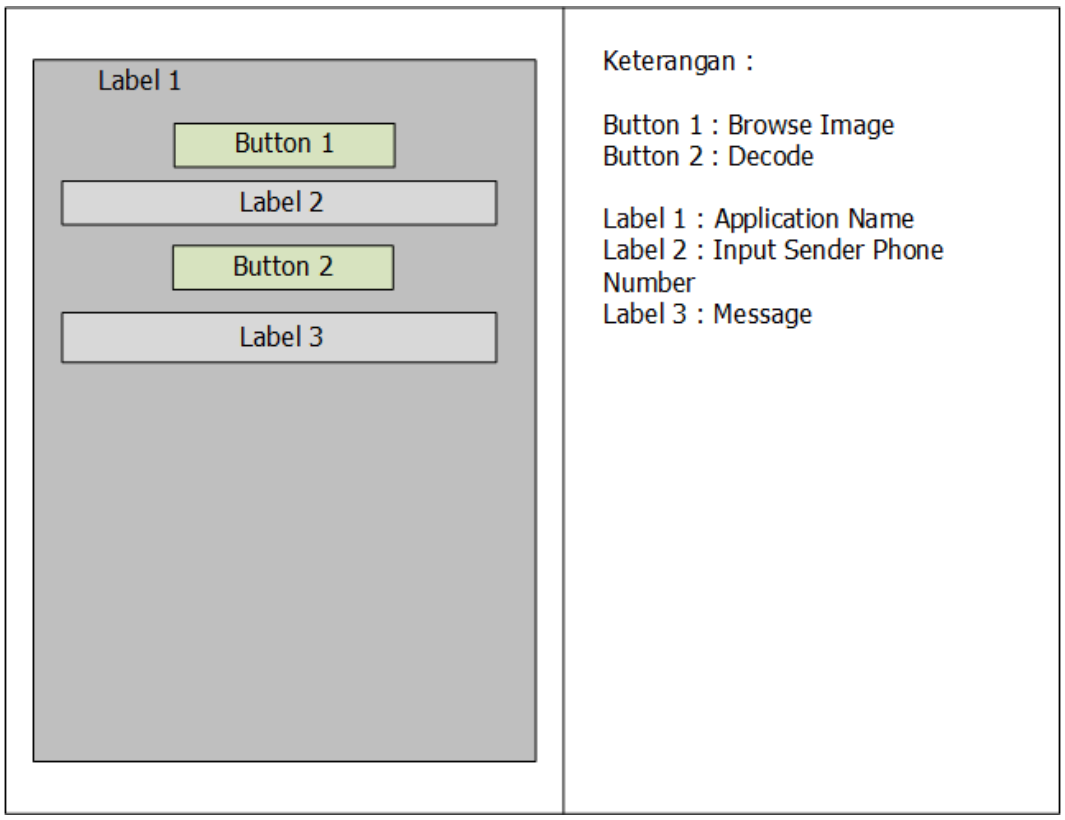
Perancangan antarmuka *write secret message* merupakan tahapan rancangan antarmuka setelah tahapan menu *home* dipilih. Perancangan antarmuka *write secret message* dari aplikasi *steganography* ini terlihat pada gambar 3.38



Gambar 3.38 Perancangan Antarmuka *Write Secret Message*

4. Perancangan Antarmuka *Read Secret Message* (T04)

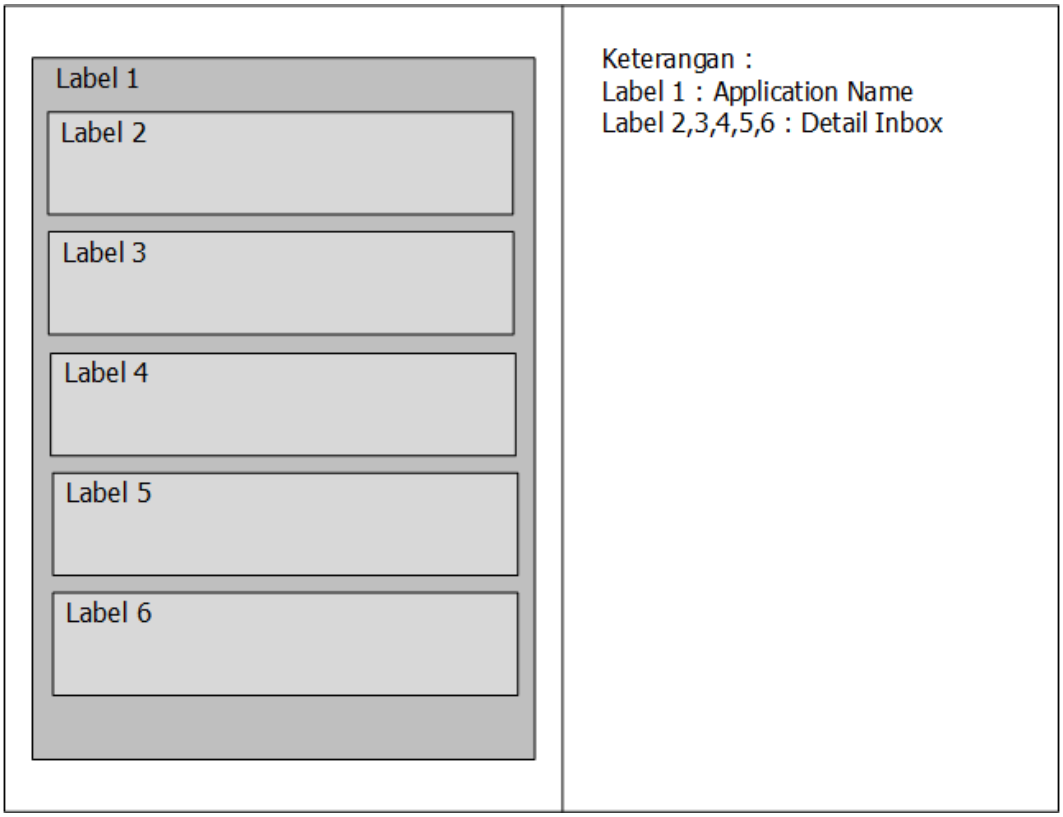
Perancangan antarmuka *read secret message* merupakan tahapan rancangan antarmuka setelah tahapan menu *home* dipilih. Perancangan antarmuka *read secret message* dari aplikasi *steganography* ini terlihat pada gambar 3.39



Gambar 3.39 Perancangan Antarmuka *Read Secret Message*

5. Perancangan Antarmuka *Inbox Key* (T05)

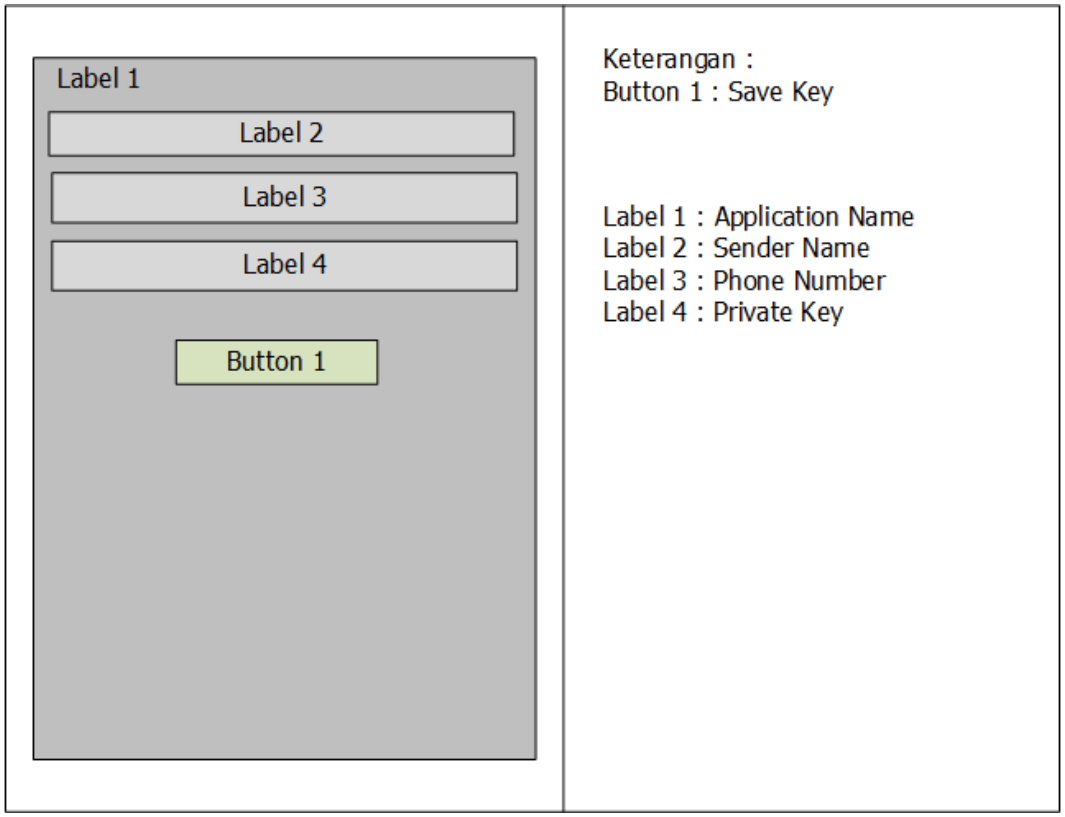
Perancangan antarmuka *inbox key* merupakan tahapan rancangan antarmuka setelah tahapan menu *home* dipilih. Perancangan antarmuka *inbox key* dari aplikasi *steganography* ini terlihat pada gambar 3.40



Gambar 3.40 Perancangan Antarmuka *Inbox Key*

6. Perancangan Antarmuka *Detail Inbox Key* (T06)

Perancangan antarmuka *detail inbox key* merupakan tahapan rancangan antarmuka setelah tahapan menu *inbox key* dipilih. Perancangan antarmuka *detail inbox key* dari aplikasi *steganography* ini terlihat pada gambar 3.41



Gambar 3.41 Perancangan Antarmuka *Detail Inbox Key*

3.2.1.3 Perancangan Pesan

Pesan merupakan tampilan dari suatu perangkat lunak yang berfungsi untuk menyampaikan notifikasi dan informasi kepada *user* agar perangkat lunak lebih interaktif. Perancangan pesan pada aplikasi *steganography* dibagi menjadi 3 bagian yaitu perancangan pesan menu *key manager*, *write secret message* dan *read secret message*.

1. Perancangan Pesan Menu *Key Manager*

Perancangan pesan pada menu management *key* ini terdapat 4 perancangan pesan yaitu sebagai berikut :

1. **Perancangan Pesan Jika Mengambil *key* (M01)**

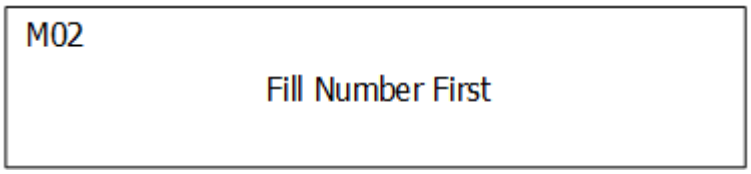
   Pesan ini muncul jika *user* berhasil mengambil *key* pada database. Perancangan pesan ini terlihat pada gambar 3.42



Gambar 3.42 Perancangan Pesan Jika Mengambil Key

2. **Perancangan Pesan Belum Memasukan *Phone Number* (M02)**

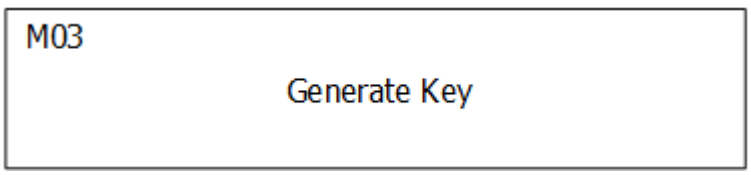
   Pesan ini muncul jika *user* belum memasukan *receiver phone number* dan menekan *button get key*. Perancangan pesan ini terlihat pada gambar 3.43



Gambar 3.43 Perancangan Pesan Jika Belum Memasukan *Phone Number*

3. **Perancangan Pesan Jika Berhasil *Ganerate Key* (M03)**

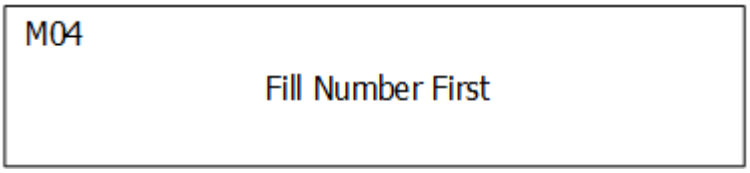
   Pesan ini muncul jika *user* berhasil *generate key*. Perancangan pesan ini terlihat pada gambar 3.44



Gambar 3.44 Perancangan Pesan Jika Berhasil *Generate Key*

4. **Perancangan Pesan jika Belum Memasukan *Phone Number* (M04)**

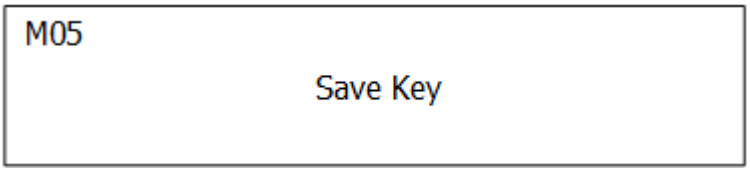
   Pesan ini muncul jika *user* belum memasukan *receiver phone number* dan menekan *button generate key*. Perancangan pesan ini terlihat pada gambar 3.45



Gambar 3.45 Perancangan Pesan Jika Belum Memasukan *Phone Number.*

5. **Perancangan Pesan Jika Menyimpan *key* (M05)**

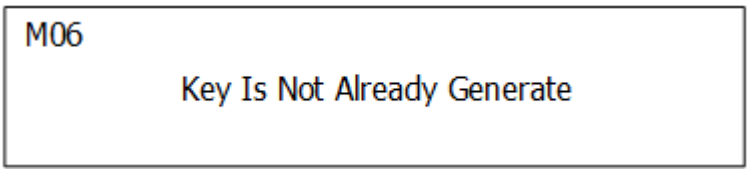
   Pesan ini muncul jika *user* berhasil menyimpan *key*. Perancangan pesan ini terlihat pada gambar 3.46



Gambar 3.46 Perancangan Pesan Jika Berhasil Menyimpan *Key*

6. **Perancangan Pesan Jika *Key* Belum Dihasilkan (M06)**

   Pesan ini muncul jika *key* belum dihasilkan dan *user* menekan *button save key*. Perancangan pesan ini terlihat pada gambar 3.47



Gambar 3.47 Perancangan Pesan Jika *Key* Belum Dihasilkan

7. **Perancangan Pesan Mengirim Pesan *Key* (M07)**

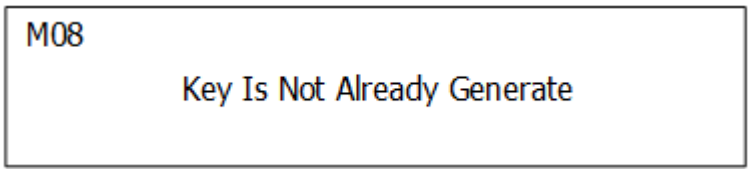
   Pesan ini muncul jika *user* berhasil mengirim pesan *key*. Perancangan pesan ini terlihat pada gambar 3.48



Gambar 3.48 Perancangan Pesan Jika Mengirim Pesan *Key*

8. **Perancangan Pesan Jika *Key* Belum Dihsailkan (M08)**

   Pesan ini muncul jika *key* belum dihasilkan dan *user* menekan *button send key*. Perancangan pesan ini terlihat pada gambar 3.49



Gambar 3.49 Perancangan Pesan Jika Mengirim Pesan *Key*

2. Perancangan Pesan Menu *Write Secret Message*

Perancangan pesan pada menu *write secret message* ini terdapat 5 perancangan pesan yaitu sebagai berikut :

1. **Perancangan Pesan Jika *Phone Number* Belum Diisi (M09)**

   Pesan ini muncul jika *user* belum memasukan *phone number*. Perancangan pesan ini terlihat pada gambar 3.50



Gambar 3.50 Perancangan Pesan Jika Belum Memasukan Phone Number

2. **Perancangan Pesan Jika Citra Belum Dipilih (M10)**

   Pesan ini muncul jika *user* belum memasukan citra atau gambar. Perancangan pesan ini terlihat pada gambar 3.51



Gambar 3.51 Perancangan Pesan Jika Citra Belum Dipilih

3. **Perancangan Pesan Jika Belum Memasukan Informasi Pesan (M11)**

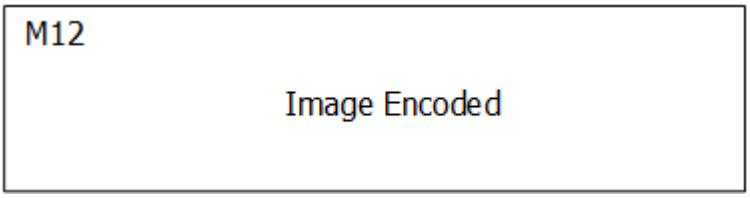
   Pesan ini muncul jika *user* belum memasukan/mengetikan pesan. Perancangan pesan ini terlihat pada gambar 3.52



Gambar 3.52 Perancangan Pesan Jika Belum Memasukan Pesan

4. **Perancangan Pesan Informasi *Encode* Berhasil (M12)**

   Pesan ini muncul jika *encode* telah berhasil dilakukan. Perancangan pesan ini terlihat pada gambar 3.53



Gambar 3.53 Perancangan Pesan *Encode* Berhasil

5. **Perancangan Pesan Jika Belum Berhasil Mengirim Pesan(M13)**

   Pesan ini muncul jika *user* telah menekan *button Send it Now* dilakukan. Perancangan pesan ini terlihat pada gambar 3.54



Gambar 3.54 Perancangan Pesan Jika Belum Mengirim Pesan

3. Perancangan Pesan Menu *Read Secret Message*

Perancangan pesan pada menu *read secret message* ini terdapat 2 perancangan pesan yaitu sebagai berikut :

1. **Perancangan Pesan Jika Citra S*tego* Belum Dipilih (M14)**

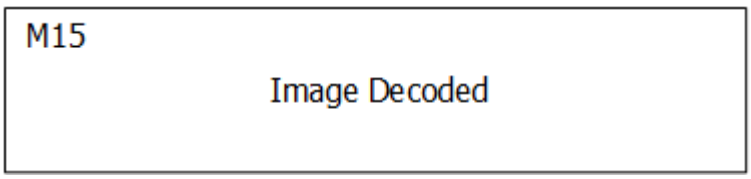
   Pesan ini muncul jika *user* belum memasukan citra *stego*. Perancangan pesan ini terlihat pada gambar 3.55



Gambar 3.55 Perancangan Pesan Jika Belum Memilih Citra Stego

2. **Perancangan Pesan Berhasil *Decode* (M15)**

   Pesan ini muncul jika *user* berhasil men*decode*. Perancangan pesan ini terlihat pada gambar 3.56



Gambar 3.56 Perancangan Pesan Berhasil *Decode*

3. **Perancangan Pesan Belum Berhasil *Decode* (M16)**

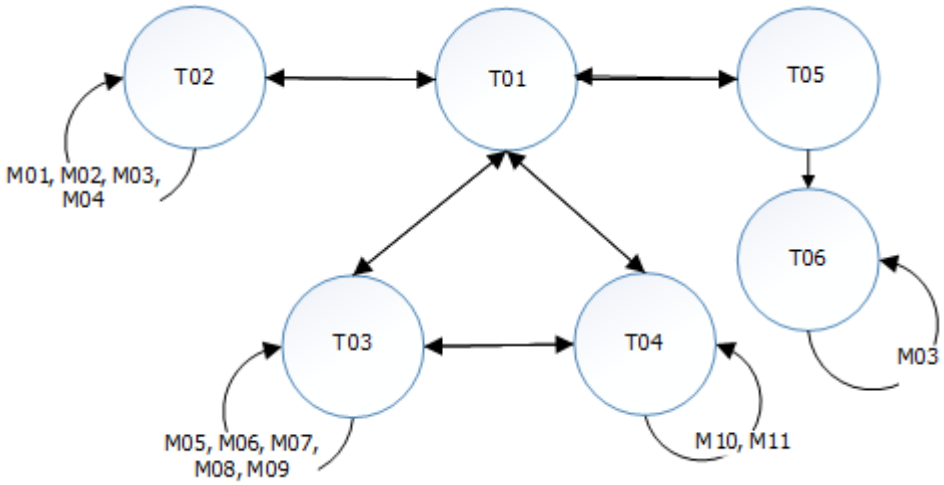
   Pesan ini muncul jika *user* belum berhasil men*decode* . Perancangan pesan ini terlihat pada gambar 3.57



Gambar 3.57 Perancangan Pesan Belum Berhasil *Decode*

3.2.1.4 Perancangan Jaringan Semantik

Jaringan semantik adalah diagram yang menggambarkan aliran-aliran menu dan pesan dalam sebuah program. Jaringan semantik dari aplikasi *steganography* yang dibangun yaitu seperti pada gambar 3.58



Gambar 3.58 Jaringan Semantik Aplikasi *Steganography*