BAB 4
IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1 Implementasi

Tahap implementasi merupakan tahap penerjemahan perancangan berdasarkan hasil analisis ke dalam suatu bahasa pemrograman yang digunakan serta penerapan perangkat lunak yang dibangun pada lingkungan yang sesungguhnya. Tujuan implementasi adalah untuk menerapkan perancangan yang telah dilakukan terhadap aplikasi, sehingga *user* dapat memberi masukan demi berkembangnya aplikasi yang telah dibangun. Adapun pembahasan untuk implementasi sistem meliputi kebutuhan perangkat keras, perangkat lunak, pengujian *alpha*, dan pengujian *beta*.

4.1.1 Implementasi Perangkat Pendukung

Perangkat keras dan perangkat lunak yang digunakan untuk melakukan implementasikan dan pengujian dari aplikasi adalah sebagai berikut:

4.1.1.1 Implementasi Perangkat Keras

Implementasi perangkat keras merupakan relisasi dari perancangan kebutuhan perangkat keras. Aplikasi *steganography* ini diimplementasikan pada perangkat *android* dengan spesifikasi seperti terlihat pada tabel 4.1.

Tabel 4.1 Spesifikasi Perangkast Keras

Perangkat Keras	Spesifikasi
CPU	APQ8064CPU Quad-Core 1.5 GHz
RAM	2 Gb
Harddisk	8 Gb
Resolusi Layar	720 x 1280 *pixel*s

4.1.1.2 Implementasi Perangkat Lunak

Implementasi perangkat lunak merupakan proses instalasi perangkat lunak, sehingga dapat beroperasi dengan benar. Perangkat lunak yang digunakan yaitu seperti terlihat pada tabel 4.2.

Tabel 4.2 Spesifikasi Perangkat Lunak

Perangkat Lunak	Keterangan
System Operasi	Windows 8 Professional
Bahasa Pemrograman	Java
Tools Pemrograman	*Eclipse*
Perangkat Pendukung	JDK 7
Picture Viewer	Windows Photo Viewer
System Operasi Android	4.4.2 Kitkat
Picture Editor	Adobe Photoshop CS3

4.1.2 Implementasi Aplikasi

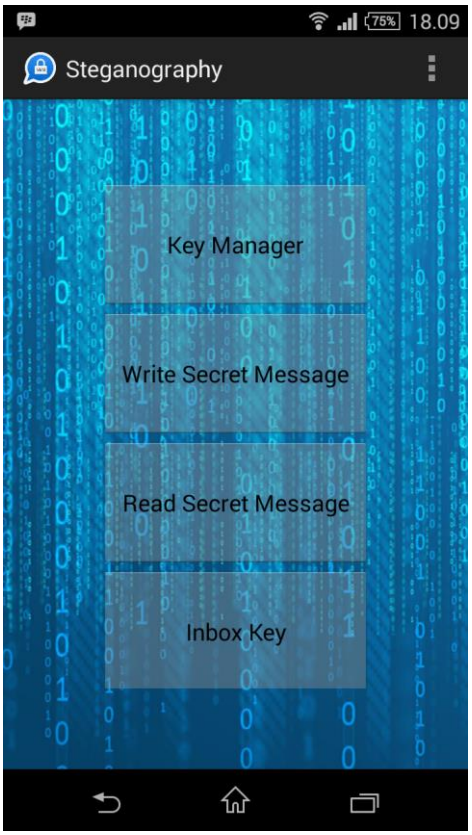
Aplikasi *steganography* merupakan sebuah aplikasi yang dibuat untuk membantu untuk menyamarkan pesan agar terjamin kerahasiaan dengan cara menyandikan dan meyisipkannya ke dalam *image* yang akan dikirimkan. Aplikasi *steganography* menggunakan metode *haar wavelet* dan algoritma *RSA* untuk menyamarkan dan menyisipkan pesan dilengkapi dengan *management key*-nya.

4.1.3 Implementasi Antarmuka

Implementasi antarmuka aplikasi *steganography* adalah sebagai berikut :

1. Tampilan Halaman Menu Utama

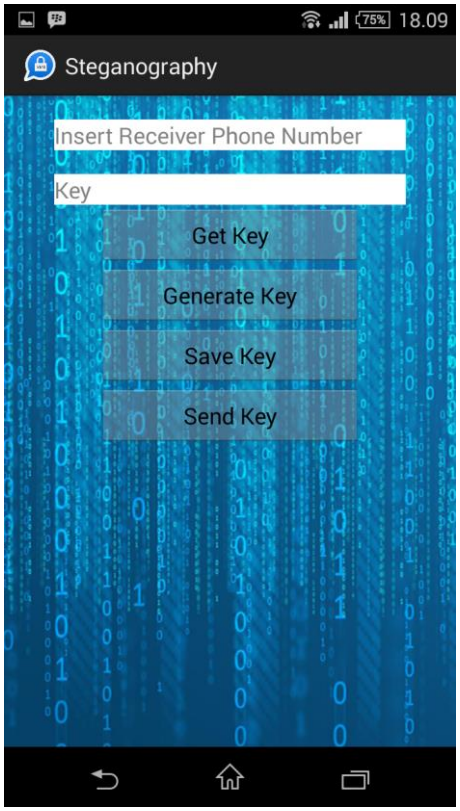
Antarmuka tampilan halaman utama merupakan halaman utama dari aplikasi *steganography*. Pada halaman utama ini terdapat 4 *button* yaitu *key manager*, *write secret message*, *read secret message* dan *inbox key*. Tampilan menu utama aplikasi *steganography* terlihat pada gambar 4.1



Gambar 4.1 Tampilan Menu Utama

2. Tampilan *Manager Key*

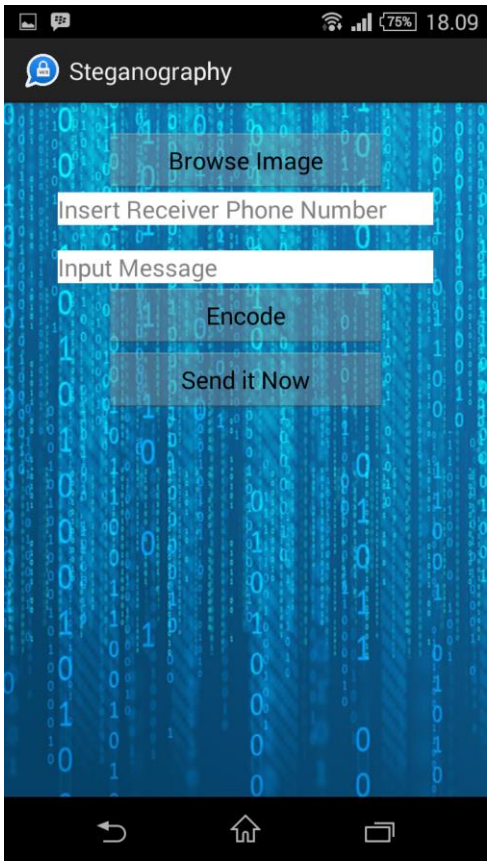
Antarmuka tampilan *manager key* merupakan halaman dimana *user* dapat melakukan *generate key* yang digunakan dalam proses *encrypt* dan *decrypt* pesan. *User* juga dapat mengirimkan *key* yang telah di*generate* kepada calon *receiver*. Tampilan *manager key* aplikasi *steganography* terlihat pada gambar 4.2.



Gambar 4.2 Tampilan Menu *Manager Key*

3. Tampilan Menu *Write Secret Message*

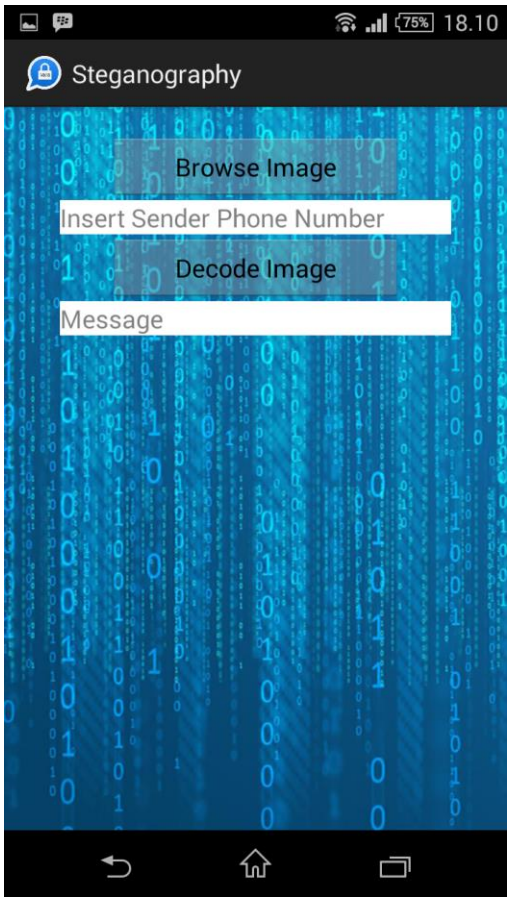
Antarmuka *write secret message* baru merupakan halaman setelah *user* menekan *button write secret message* pada halaman utama. Pada halaman ini *user* dapat melakukan *encode message*. Tampilan pesan baru aplikasi *steganography* terlihat pada gambar 4.3



Gambar 4.3 Tampilan Menu *Write Secret Message*

4. Tampilan Menu *Read Secret Message*

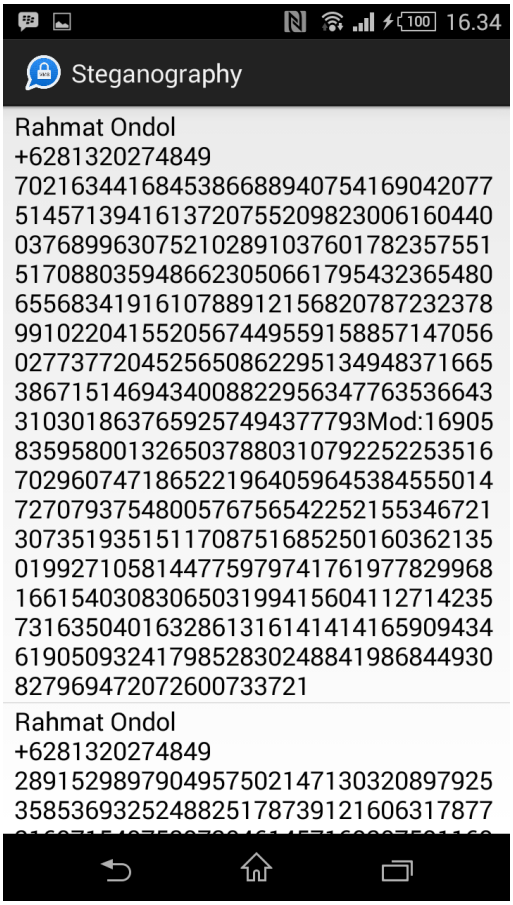
Antarmuka *read secret message* baru merupakan halaman setelah *user* menekan *button read secret message* pada halaman utama. Pada halaman ini *user* dapat melakukan *decode message*. Tampilan pesan baru aplikasi *steganography* terlihat pada gambar 4.4



Gambar 4.4 Tampilan Menu *Read Secret Message*

5. Tampilan *Inbox Key*

Antarmuka tampilan *inbox key* merupakan halaman *list* kotak masuk pesan tersimpan. Tampilan kotak masuk aplikasi *steganography* terlihat pada gambar 4.5.



Gambar 4.5 Tampilan Menu *Inbox Key*

6. Tampilan *Detail Inbox Key*

Antarmuka tampilan *detail inbox key* merupakan halaman setelah *user* memilih pesan *privet key* yang telah dikirimkan untuk disimpan. Tampilan *detail inbox key* aplikasi *steganography* terlihat pada gambar 4.6



Gambar 4.6 Tampilan *Detail Inbox Key*

4.2 Pengujian Aplikasi

Pengujian aplikasi dilakukan untuk menentukan kesalahan atau kekurangan pada perangkat lunak yang diuji. Pengujian bermaksud untuk mengetahui perangkat lunak yang dibangun sudah sesuai dengan perancangan yang dibuat dan memenuhi kebutuhan *user*.

Pengujian yang dilakukan yaitu pengujian *alpha* dan pengujian *beta*. Pengujian *alpha* dilakukan dengan pengujian *black box* dan *white box*. Pengujian *black box* digunakan untuk mengetahui kesalahan proses secara fungsional sedangkan *white box* digunakan untuk menguji performa dari metode yang digunakan. Pengujian *beta* digunakan untuk mengetahui tanggapan *user* terhadap aplikasi dengan menggunakan kuesioner.

4.2.1 Pengujian Alpha

Pengujian fungsional yang digunakan untuk menguji aplikasi adalah metode pengujian *alpha*. Pengujian *alpha* dilakukan dengan menggunakan metode *black box* dan *white box.* Pengujian *black box* berfokus pada persyaratan fungsional perangkat lunak sedangkan *white box* berfokus pada Algoritma dari metode yang digunakan. *Scenario* pengujian dari aplikasi *steganography* dapat dilihat pada tabel 4.3

Tabel 4.3 Scenario Pengujian Alpha

No	Komponen Yang Diuji	*Scenario* Pengujian	Jenis Pengujian
1	Menu *Manager Key*	Memilih *Button Manager Key*	*Black box*
		Memasukan No. *Receiver*	*Black box*
		Menekan *Button Generate Key*	*Black box*
		Menekan *Button Save Key*	*Black box*
		Menekan *Button Send Key*	*Black box*
2	Menu *Write Secret Message*	Menekan *Button Write Secret Message*	*Black box*
		Memasukan No. Tujuan	*Black box*
		Mengetikan Pesan	*Black box*
		Menekan *Button Encode*	*Black box*
		Menekan *Button Send It Now*	*Black box*
3	Menu *Read Secret Message*	Menekan *Button Read Secret Message*	*Black box*

		Menekan *Button Browse* dan Memilih *Image* .png	*Black box*
		Memasukan No. Tujuan	*Black box*
		Mengetikan Pesan	*Black box*
		Menekan *Button Decode*	*Black box*
4	Menu *Inbox Key*	Menekan *Button Inbox Key*	*Black box*
		Memilih Pesan	*Black box*
5	Menu *Detail Inbox Key*	Menekan Pesan Masuk	*Black box*
		Menekan *Button Save Key* Jika Pesan Masuk Merupakan Kunci	*Black box*

4.2.1.1 Kasus dan Hasil Pengujian *Black Box*

Pengujian perangkat lunak yang sudah dibangun dilakukan dengan menggunakan metode *black box*, berdasarkan pengamatan yang sudah dilakukan maka dihasilkan kesimpulan dari pengujian pada tabel 4.4

Tabel 4.4 Pengujian *Black Box*

No	Kasus yang Diuji	*Scenario* dan Hasil Uji		
		Scenario Uji	Hasil yang diharapkan	Kesimpulan
1	Menu *Manager Key*	Memilih *Button Manager Key*	*System* masuk ke menu *Manager Key*	[√] Berhasil [ ] Tidak Berhasil
		Memasukan no. *Receiver*	*System* menerima no. tujuan yang diketik	[√] Berhasil [ ] Tidak Berhasil
		Menekan *Button Generate Key*	*System* membangkitkan pasangan kunci	[√] Berhasil [ ] Tidak Berhasil
		Menekan *Button Save key*	*System* menyimpan kunci ke dalam *database*	[√] Berhasil [ ] Tidak Berhasil
		Menekan *Button Send key*	*System* mengirimkan *private key* kepada no. yang dituju atau *receiver*	[√] Berhasil [ ] Tidak Berhasil
2	Menu *Inbox Key*	Menekan *Button Inbox Key*	*System* menampilkan *list* kotak masuk	[√] Berhasil [ ] Tidak Berhasil
		Memilih pesan	*System* menampilkan *detail* kotak masuk	[√] Berhasil [ ] Tidak Berhasil
3	Menu *Detail Inbox Key*	Memilih pesan yang berisikan *private key*	*System* menampilkan pesan	[√] Berhasil [ ] Tidak Berhasil
		Menekan *Button Save Key* jika pesan masuk	*System* menyimpan kunci ke dalam *database*	[√] Berhasil [ ] Tidak Berhasil

		merupakan kunci		
4	Menu *Write Secret Message*	Menekan *Button Write Secret Message*	*System* menampilkan tampilan *Write Secret message*	[√] Berhasil [ ] Tidak Berhasil
		Menekan *Button Browse* dan memilih gambar	*System* menampilkan galeri gambar	[√] Berhasil [ ] Tidak Berhasil
		Memasukan no. tujuan	*System* menampilkan nama dan nomor *receiver* pesan	[√] Berhasil [ ] Tidak Berhasil
		Mengetikan *Message*	*System* menampilkan pesan plaintext yang diketikan *user*	[√] Berhasil [ ] Tidak Berhasil
		Menekan *Button Encode*	*System* menampilkan *info* "*Image Encoded"*, yang menandakan bahwa gambar telah berhasil di *encode.*	[√] Berhasil [ ] Tidak Berhasil
		Menekan *Button Send It Now*	*System* mengirimkan *Stego Image* melaui MMS	[√] Berhasil [ ] Tidak Berhasil
4	Menu *Read Secret Message*	Menekan *Button Read Secret Message*	*System* menampilkan tampilan *Read Secret Message*	[√] Berhasil [ ] Tidak Berhasil
		Menekan *Button Browse* dan memilih gambar yang akan diunkap	*System* menampilkan galeri gambar	[√] Berhasil [ ] Tidak Berhasil
		Memasukan no. tujuan	*System* menampilkan nama dan nomor *receiver* pesan	[√] Berhasil [ ] Tidak Berhasil
		Menekan *Button Decode*	*System* menampilkan *info* "*Image Decoded"*, yang menandakan bahwa gambar telah berhasil di *decode* dan *system* menampilkan pesan telah disisipkan sebelumnya ke dalam gambar.	[√] Berhasil [ ] Tidak Berhasil

4.2.2 Pengujian *White Box*

Pengujian *white box* bertujuan untuk mengetahui kinerja logika yang dibuat pada sebuah sistem apakah berjalan dengan baik atau tidak. Pengujian *white box* yang akan dilakukan pada aplikasi yaitu menggunakan *flow graph* yang digunakan untuk menggambarkan alur dari algoritma dan *graph matrix* yang digunakan untuk menggenerasi *flow graph*. Algoritma yang akan diujikan pada pengujian *white box* ini yaitu pengujian *encrypt RSA, decrypt RSA,* dan *generate key RSA.*

4.2.2.1 Kasus dan Hasil Pengujian *White Box*

Berikut ini adalah kasus menguji perangkat lunak yang telah dibangun dengan menggunakan metode *white box*. Terdapat 3 pengujian *white box* yaitu: *encrypt*, *decrypt* dan *generate key*.

1. Pengujian *RSA Encryption*

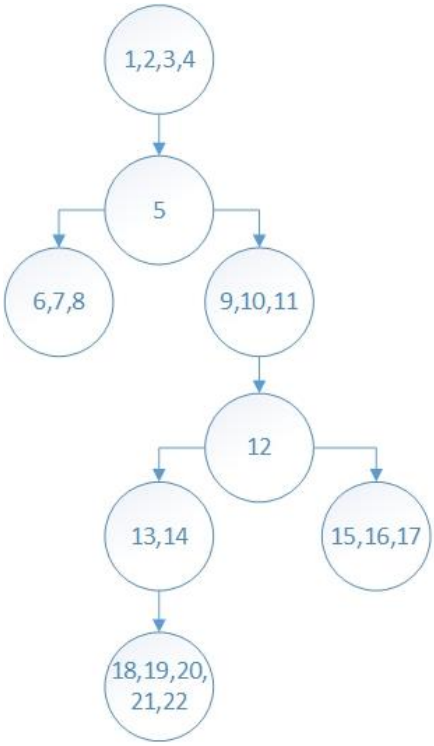
Pengujian *encryption* pada Algoritma *RSA* terlihat pada tabel 4.5

Tabel 4.5 Pengujian *White Box RSA Encryption*

Line	Source
1	`btnEncrypt.setOnClickListener(new OnClickListener() {`
2	`  @Override`
3	`  public void onClick(View v) {`
4	`    Key key;`
5	`      if ((key = MainActivity.dao`
6	`        .getKey(getNumber(textDestinationNumber.getText() .toString()))) == null) {`
7	`        Toast.makeText(getApplicationContext(), "Not already set",`
8	`        Toast.LENGTH_LONG).show();`
9	`      } else {`
10	`        String encrypted = MainActivity.rsa.enc(editMessage`
11	`          .getText().toString(), key.getMod());`
12	`          try {`
13	`              byte[] utf16 = encrypted.getBytes("UTF-16LE");`
14	`              encrypted = new String(utf16, "UTF-16LE");`
15	`          } catch (UnsupportedEncodingException e) {`
16	`              e.printStackTrace();`
17	`              Log.d("haha", "hihi");`
	`          }`
18	`          editEncryptedMessage.setText(encrypted);`
19	`          InboxActivity.message = encrypted;`
20	`          Log.d("Encrypted", encrypted);`
21	`          DetailInboxActivity.message = encrypted;`
22	`          editPrivateKey.setText(key.getPrivateKey() + "Mod:"+ key.getMod());`
	`      }`
	`  }`
	`});`

a) Flow Graph RSA Encryption

Berdasarkan tabel 4.5 maka dapat dibentuk *flow graph* seperti pada gambar 4.7



Gambar 4.7 Flow Graph RSA Encryption

Keterangan :

◯ = Menggambarkan kondisi

→ = Menggambarkan aksi

Dari gambar IV.1 dapat dihitung *cyclomatic complexity* sebagai berikut:

V(G) = E – N + 2

V(G) = 8 *edge* – 7 *node* + 2 = 3

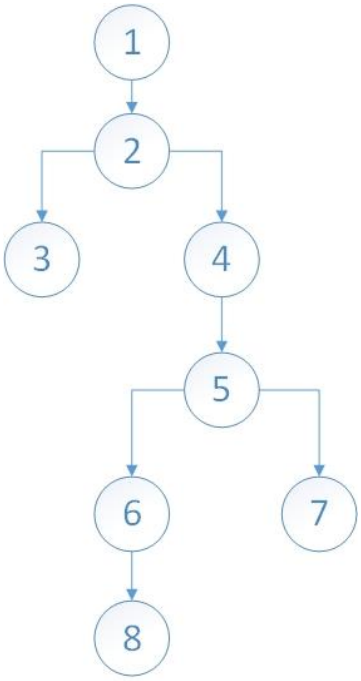
Jadi, *cyclomatic complexity* untuk gambar 4.1 adalah 3. Berdasarkan *cyclomatic complexity* tersebut, maka terdapat 3 *path* yang terdiri dari:

Path 1: 1-2-3-4-3-5-6-7-6-8

Path 2: 1-2-3-4-3-5-9-10-11-12-13-14-18-19-10-21-22

Path 3: 1-2-3-4-3-5-9-10-11-12-15-16-17

Maka gambar 4.7 disederhanakan berdasarkan kondisi atau simpul pada gambar 4.8



Gambar 4.8 Hasil Penyederhanaan Flow Graph RSA Encryption

b) Graph Matrix RSA Encryption

Graph matrix encryption dari Algoritma *RSA* terlihat pada tabel 4.6

Tabel 4.6 *Graph Matrix RSA Encryption*

	2	**3**	**4**	**5**	**6**	**7**	**8**	**sum**
1	1							0
2		1	1					1
3								0
4				1				0
5					1	1		1
6							1	0
Total								**2**

V(G) = Jumlah *Graph Matric* + 1

V(G) = 2 + 1 = 3

Berdasarkan pengujian yang dilakukan pada setiap metode, dihasilkan nilai *Cyclomatic Complexity* yang sama yaitu 3, Maka dapat disimpulkan bahwa pengujian *white box* pada proses *RSA Encryption* berjalan dengan baik, karena setiap pengujian menghasilkan nilai yang sama.

2. Pengujian *RSA Decryption*

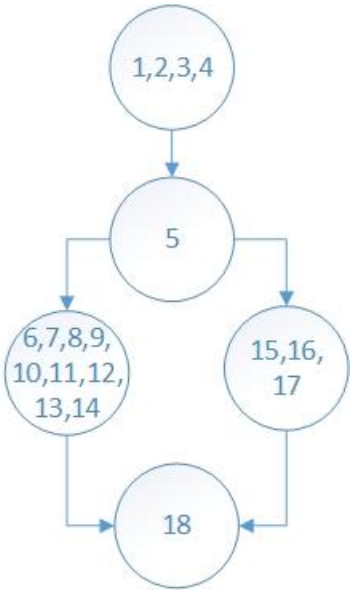
Pengujian *decryption* dari Algoritma *RSA* terlihat pada tabel 4.7

Tabel 4.7 Pengujian *White Box RSA Decryption*

Line	Source
1	`btnDecrypt.setOnClickListener(new OnClickListener() {`
2	`  @Override`
3	`  public void onClick(View v) {`
4	`    Key key = null;`
5	`      if ((key = MainActivity.dao.getKeyR(editSender.getText().toString())) != null) {`
6	`        Log.d("Keyz", "Private: " + key.getPrivateKey());`
7	`        Log.d("Keyz", "Mod: " + key.getMod());`
8	`        String decrypted = MainActivity.rsa`
9	`          .toSingleString(MainActivity.rsa.dec(`
10	`            editEncryptedMessageInbox.getText()`
11	`              .toString(), key.getPrivateKey(),`
12	`            key.getMod()));`
13	`        editPrivateKeyMessageInbox.setText(key.getPrivateKey());`
14	`        editDecryptedMessage.setText(decrypted);`
15	`      } else {`
16	`        Toast.makeText(getApplicationContext(),`
17	`          "Key not available", Toast.LENGTH_LONG).show();`
	`      }`
	`    }`
	`});`

a) Flow Graph RSA Decryption

Berdasarkan tabel 4.7 maka dapat dibentuk *flow graph* seperti pada gambar 4.9



Gambar 4.9 Flow Graph RSA Decryption

Keterangan :

◯ = Menggambarkan kondisi

→ = Menggambarkan aksi

Dari gambar IV.3 dapat dihitung *cyclomatic complexity* sebagai berikut:

V(G) = E – N + 2

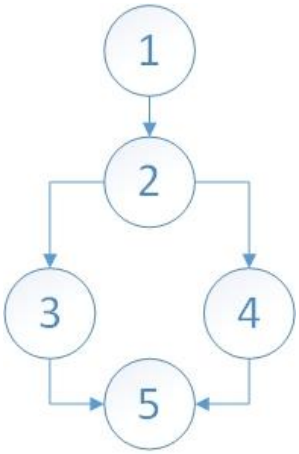
V(G) = 5 *edge* – 5 *node* + 2 = 2

Jadi, *cyclomatic complexity* untuk gambar 4.1 adalah 2. Berdasarkan *cyclomatic complexity* tersebut, maka terdapat 2 *path* yang terdiri dari:

Path 1: 1-2-3-4-3-5-6-7-6-8-9-10-11-12-13-14-18

Path 2: 1-2-3-4-3-5-15-16-17-18

Maka gambar 4.9 disederhanakan berdasarkan kondisi atau simpul pada gambar 4.10



Gambar 4.10 Penyederhanaan *Flow Graph RSA Decryption*

b) Graph Matrix RSA Encryption

Graph matrix decryption dari Algoritma *RSA* terlihat pada tabel 4.8

Tabel 4.8 *Graph Matrix RSA Decryption*

	2	**3**	**4**	**5**	**sum**
1	1				0
2		1	1		1
3			1		0
4				1	0
Total					**1**

V(G) = Jumlah *Graph Matric* + 1

V(G) = 1 + 1

V(G) = 2

Berdasarkan pengujian yang dilakukan pada setiap metode, dihasilkan nilai *Cyclomatic Complexity* yang sama yaitu 2, Maka dapat disimpulkan bahwa pengujian *white box* pada proses *RSA Decryption* berjalan dengan baik, karena setiap pengujian menghasilkan nilai yang sama.

3. Pengujian *Generate Key*

Pengujian *generate key* pada Algoritma *RSA* terlihat pada tabel 4.9

Tabel 4.9 Pengujian *White Box Generate Key*

Line	Source
1	`public void encrypt(int m){`
2	`getPQ();`
3	`computeN();`
4	`computePhi();`
5	`getE();`
6	`computeD();`
7	`computePublicKey();`
8	`computePrivateKey();`
9	`int z=(int)(Math.pow(m, publicKey[0])%publicKey[1]);`
10	`int x=(int)(Math.pow(z, privateKey[0])%privateKey[1]);`
11	`}`
12	`public void getPQ(){`
13	`p=getPrimeNumber();`
14	`q=getPrimeNumber();`
15	`}`
16	`public int getPrimeNumber(){`
17	`int r;`
18	`outer:`
19	`while(true){`
20	`r=(int)(Math.random()*100);`
21	`for(int i=2; i<r; i++){`
22	`if(r%i==0){`
23	`break outer;`
24	`}`
25	`}`
26	`}`
27	`return r;`
28	`}`
29	`public void computeN(){`
30	`n=p*q;`
31	`}`

```
32  public void computePhi(){
33      phi=(p-1)*(q-1);
34  }

35  public void getE(){
36      int r=(int)(Math.random()*phi);
37      if(r-1>1){
38        r=r-1;
39      }
40      e=r;
41  }

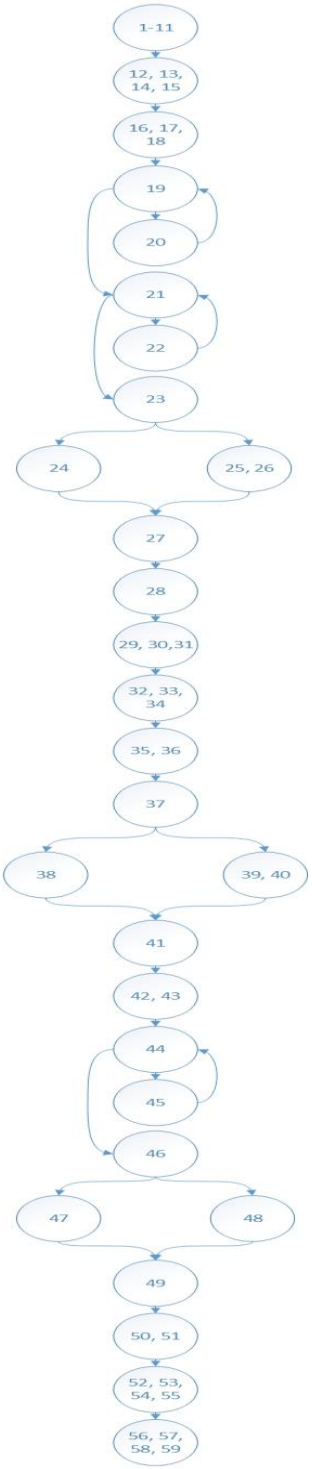
42  public void computeD(){
43      int r;
44      while(true){
45        r=(int)(Math.random()*100);
46        if((r*e)%phi==1){
47            d=r;
48            break;
49        }
50      }
51  }

52  public void computePublicKey(){
53      publicKey[0]=e;
54      publicKey[1]=n;
55  }

56  public void computePrivateKey(){
57      privateKey[0]=d;
58      privateKey[1]=n;
59  }
```

a) Flow Graph Key Generate

Berdasarkan tabel 4.10 maka dapat dibentuk *flow graph* seperti pada gambar 4.11



Gambar 4.11 *Flow Graph Key Generate*

Keterangan :

= Menggambarkan kondisi

= Menggambarkan aksi

Dari gambar IV.13 dapat dihitung *cyclomatic complexity* sebagai berikut:

V(G) = E – N + 2

V(G) = 34 *edge* – 29 *node* + 2 = 7

Jadi, *cyclomatic complexity* untuk gambar 4.11 adalah 4. Berdasarkan *cyclomatic complexity* tersebut, maka terdapat 2 *path* yang terdiri dari:

Path 1: 1-11-12-13-14-15-16-17-18-19-20-19-21-22-21-23-25-26-28-29-30-31-32-33-34-35-36-37-39-40-41-42-43-44-45-44-46-48-49-50-51-52-53-54-55-56-57-58-59

Path 2: 1-11-12-13-14-15-16-17-18-19-21-22-21-23-25-26-28-29-30-31-32-33-34-35-36-37-39-40-41-42-43-44-45-44-46-48-49-50-51-52-53-54-55-56-57-58-59

Path 3: 1-11-12-13-14-15-16-17-18-19-21-23-25-26-28-29-30-31-32-33-34-35-36-37-39-40-41-42-43-44-45-44-46-48-49-50-51-52-53-54-55-56-57-58-59

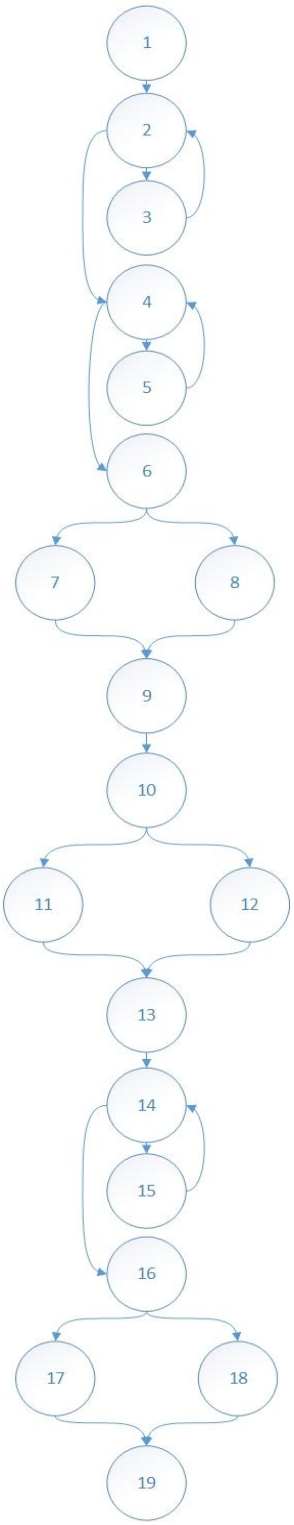
Path 4: 1-11-12-13-14-15-16-17-18-19-20-19-21-22-21-23-24-28-29-30-31-32-33-34-35-36-37-39-40-41-42-43-44-45-44-46-48-49-50-51-52-53-54-55-56-57-58-59

Path 5: 1-11-12-13-14-15-16-17-18-19-20-19-21-22-21-23-24-28-29-30-31-32-33-34-35-36-37-38-40-41-42-43-44-45-44-46-48-49-50-51-52-53-54-55-56-57-58-59

Path 6: 1-11-12-13-14-15-16-17-18-19-20-19-21-22-21-23-25-26-28-29-30-31-32-33-34-35-36-37-39-40-41-42-43-44-46-48-49-50-51-52-53-54-55-56-57-58-59

*Path*7: 1-11-12-13-14-15-16-17-18-19-20-19-21-22-21-23-25-26-28-29-30-31-32-33-34-35-36-37-39-40-41-42-43-44-45-44-46-47-49-50-51-52-53-54-55-56-57-58-59

Maka gambar 4.11 disederhanakan berdasarkan kondisi atau simpul pada gambar 4.12



Gambar 4.12 Penyederhanaan *Flow Graph Key Generate*

b) Graph Matrix Key Generate

Graph matrix key generate dari Algoritma *RSA* terlihat pada tabel 4.10

Tabel 4.10 *Graph Matrix Key Generate*

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	**sum**
1	1																		
2		1	1																1
3	1																		
4				1	1														1
5			1																
6						1	1												1
7								1											
8								1											
9									1										
10										1	1								1
11												1							
12												1							
13													1						
14														1	1				1
15													1						
16																1	1		1
17																		1	
18																		1	
Total																			6

V(G) = Jumlah *Graph Matric* + 1

V(G) = 6 + 1

V(G) = 7

Berdasarkan pengujian yang dilakukan pada setiap metode, dihasilkan nilai *Cyclomatic Complexity* yang sama yaitu 7, Maka dapat disimpulkan bahwa pengujian *white box* pada proses *key generate* Algoritma *RSA* berjalan dengan baik, karena setiap pengujian menghasilkan nilai yang sama.

4.2.3 Pengujian *Black Box*

Pengujian yang dilakukan yaitu terdiri pengujian performansi, pengujian kualitas citra digital, dan pengujian ketahanan citra digital.

4.2.3.1 Pengujian Performansi

Pengujian performansi pada citra PNG dibagi menjadi dua yaitu pengujian proses *embedding* dan *extracting* dengan panjang pesan yang berbeda dan pengujian proses *embedding* dan *extracting* dengan dimensi yang berbeda.

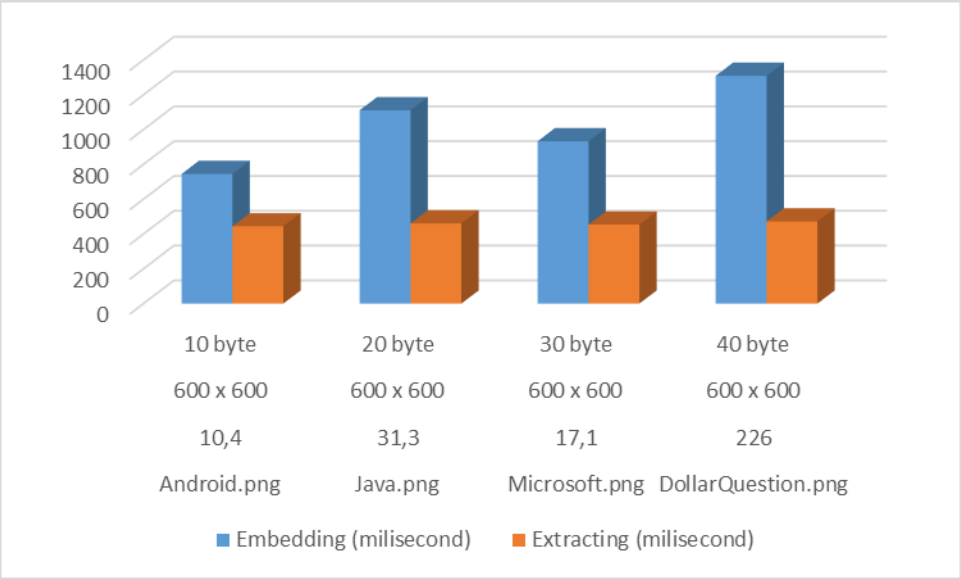
Pengujian performansi dilakukan untuk mengetahui kinerja dari proses *embedding* dan *extracting* steganografi menggunakan metode *haar wavelet* dilihat dari waktu prosesnya. Pengujian dilakukan terhadap 4 citra png.

1. **Pengujian Proses *Embedding* dan *Extracting* dengan panjang pesan yang berbeda.**

   Hasil pengujian performansi dari proses *embedding* dan *extracting* dengan panjang pesan yang berbeda dapat dilihat pada tabel 4.11

Tabel 4.11 Pengujian Proses *Embedding* dan *Extracting* dengan panjang pesan yang berbeda

Citra PNG	Size	*Dimensions*	Panjang Pesan (byte)	*Embedding (milisecond)*	*Extracting (milisecond)*
Android.png	10,400	600 x 600	10 byte	745,04	444,03
Java.png	31,300	600 x 600	20 byte	1111,06	461,13
Microsoft.png	17,100	600 x 600	30 byte	933,05	454,05
DollarQuestion .png	226,000	600 x 600	40 byte	1309,07	472,03



Gambar 4.13 Grafik Pengujian Proses *Embedding* dan *Extracting* dengan panjang pesan yang berbeda

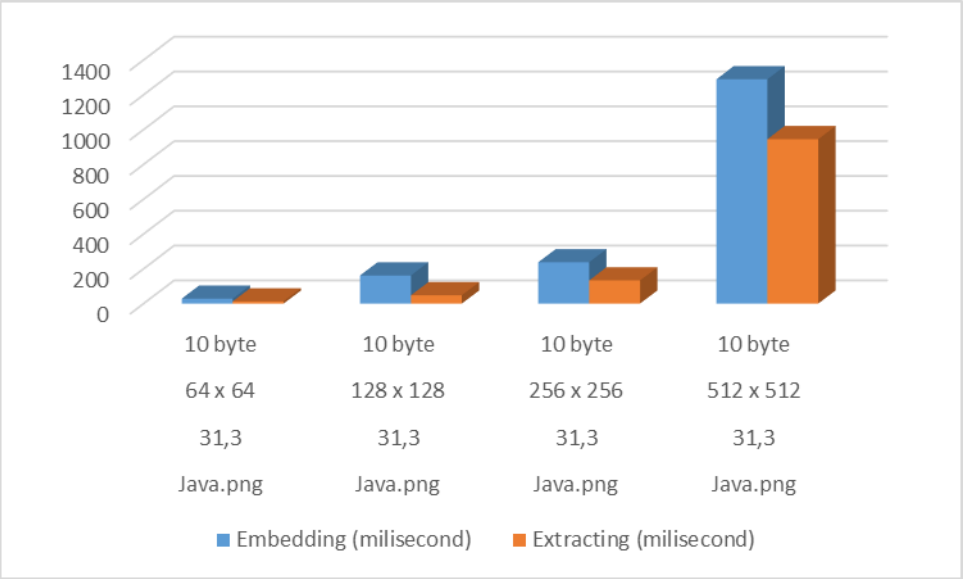
Berdasarkan hasil pengujian pengaruh panjang pesan terhadap lama waktu proses *embedding* dan *extracting* seperti pada tabel 4.11, dapat disimpulkan bahwa panjang pesan *(byte)* mempengaruhi proses *embedding* dan *extracting*, semakin besar panjang pesan *(byte)* semakin lama waktu proses *embedding* dan *extracting.*

2. **Pengujian Proses *Embedding* dan *Extracting* dengan dimensi yang berbeda**

Hasil pengujian performansi dari proses proses *embedding* dan *extracting* dengan dimensi yang berbeda dapat dilihat pada tabel 4.12

Tabel 4.12 Pengujian Proses *Embedding* dan *Extracting* dengan dimensi yang berbeda

Citra PNG	Size	*Dimensions*	Panjang Pesan(byte)	*Embedding (milisecond)*	*Extracting (milisecond)*
Java.png	31,300	64 x 64	10 byte	28,4	12,7
Java.png	31,300	128 x 128	10 byte	161,5	47,9
Java.png	31,300	256 x 256	10 byte	237,4	133,3
Java.png	31,300	512 x 512	10 byte	1289,12	944,8



Gambar 4.14 Grafik Pengujian Proses *Embedding* dan *Extracting* dengan dimensi yang berbeda

Berdasarkan hasil pengujian pengaruh ukuran dimensi citra terhadap lama waktu proses *embedding* dan *extracting* seperti pada tabel 4.12, dapat disimpulkan bahwa ukuran dimensi citra mempengaruhi proses *embedding* dan *extracting*, semakin besar ukuran dimensi citra semakin lama waktu proses *embedding* dan *extracting.*

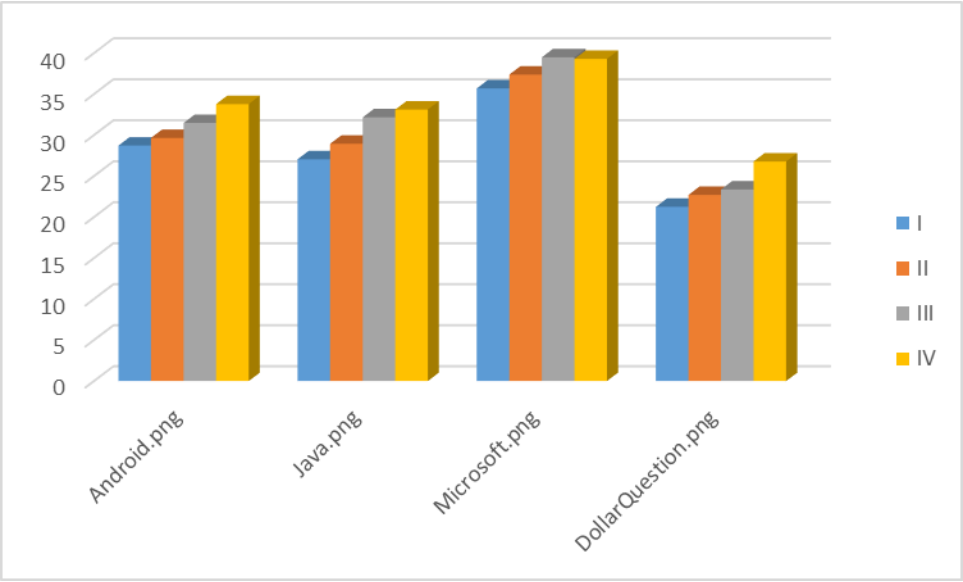
4.2.3.2 Pengujian Kualitas Citra

Pengujian kualitas citra dilakukan untuk mengetahui kualitas citra yang telah disisipi pesan. Kualitas pada citra dapat diketahui menggunakan *Peak Signal to Noise Ratio* (PSNR) yang merupakan standar pengukuran error pada pengolahan citra. Kualitas citra yang telah dimanipulasi pada umumnya mempunyai nilai PSNR yang berada pada rentang antara 20 – 40db[6].

Pengujian kualitas pada citra PNG ini dilakukan terhadap 4 citra PNG. Hasil dari pengujian ini terlihat pada tabel 4.13

Tabel 4.13 Hasil Pengujian Kualitas Citra

***Citra* PNG**	**Ukuran (byte)**	**Nilai PSNR *Citra* (db)**			
		I	**II**	**III**	**IV**
Android.png	9.480	28.72	29.65	31.48	33.78
Java.png	30.200	27.05	28.94	32.17	33.12
Microsoft.png	17.200	35.70	37.38	39.49	39.29
DollarQuestion.png	226.000	21.25	22.71	23.37	26.78
Rata-rata PSNR		28,18	29,67	31,62	33,24
		30,65			



Gambar 4.15 Grafik Pengujian Kualitas Citra

Berdasarkan pada pengujian kualitas terhadap empat citra *png* dengan pesan yang sama, didapatkan rata-rata kualitas citra yaitu 30.65 db seperti pada tabel 4.13, yang berarti kualitas citra yang telah disisipi pesan masih dalam keadaan baik. Pengujian kualitas citra terhadap aplikasi *steganography* dilakukan menggunakan *tools* PSNR 1.2.

4.2.3.3 Pengujian Ketahanan

Pengujian ketahanan dilakukan dengan merusak data citra yang menjadi media stego dan menguji ketahanan dari teknik *steganogrphy* terhadap manipulasi yang dilakukan pada media stego. Proses pengujian dilakukan dengan cara merubah format data citra PNG menjadi format lain serta diikuti dengan proses manipulasi, keudian dikembalikan lagi kedalam format PNG. Setelah itu dilakukan proses *extracting* untuk mendapatkan data *hiding*-nya. Hasil pengujian ketahanan yang dilakukan pada 4(empat) buah *sample* citra dapat dilihat pada tabel 4.14 dan tabel 4.15

Tabel 4.14 Hasil Pengujian Untuk *Robustness*

Citra PNG	Jumlah Bit Eror (%)					Validasi (1= Valid, 0= Tidak Valid)				
	TE1	TE2	TE3	TE4	TE5	TE1	TE2	TE3	TE4	TE5
Android.png	100	100	100	100	100	0	0	0	0	0
Java.png	100	100	100	100	100	0	0	0	0	0
Microsoft.png	100	100	100	100	100	0	0	0	0	0
DollarQuestion.png	100	100	100	100	100	0	0	0	0	0

Tabel 4.15 Tingkat Keberhasilan Proses *Extracing* Pengujian Ketahanan

Citra PNG	Jumlah Bit Eror(%)				
	TE1	TE2	TE3	TE4	TE5
Android.png	0	0	0	0	0
Java.png	0	0	0	0	0
Microsoft.png	0	0	0	0	0
DollarQuestion.png	0	0	0	0	0

Keterangan:

TE1 : Pengujian perubahan format data dari PNG ke JPEG kemudian kembali ke PNG

TE2 : Pengujian perubahan level *brightness* / *contrast*

TE3 : Pengujian perubahan *saturation* warna pada data citra

TE4 : Pengujian pemberian efek *blur*

TE5 : Pengujian *cropping*

Berdasarkan hasil pengujian data *hiding* dalam citra dengan format PNG dengan cara merubah format mengalami kegagalan dalam proses ekstraksi. Begitu pula ketika dilakukan pengujian terhadap manipulasi data citra seperti perubahan level *brightness* dan *contrast*, perubahan saturasi warna, pemberian efek *blur* serta manipulasi data citra dengan melakukan *cropping*, data *hiding* yang ada dalam media stego tersebut rusak ini terbukti ketika proses ekstraksi dilakukan, validitas data pesan yang tersimpan adalah 0% karena sama sekali tidak sama dengan data aslinya sehingga dapat dikatakan data rusak saat terjadi manipulasi pada media stego tersebut. Hal ini disebabkan karena bit-bit yang disisipkan ketika ditransformasikan kembali, nilai-nilainya akan disebarkan secara merata. Saat akan melakukan ekstraksi, nilai-nilai yang didapatkan dari hasil ekstraksi akan berbeda dengan saat sebelum disisipkan data hiding. Dari hasil pengujian didapatkan apabila ada perubahan pada bit-bit seperti ketika mengalami kompresi, akan berpengaruh dan menyebabkan nilai koefisian berubah. Dengan pengujian ini, dapat dikatakan metode *steganography haar waevelet* tidak *robust* terhadap proses manipulasi pada data citra yang menjadi penampungnya.

4.2.4 Pengujian Beta

Pengujian *beta* merupakan pengujian yang dilakukan secara objektif yang mana melakukan pengujian secara langsung ke lapangan, dengan tujuan untuk mengetahui sejauh mana kualitas dari sistem. Pengujian *beta* dilakukan dengan penelitian terhadap responden atau calon pengguna sistem untuk mengetahui penilaian pengguna terhadap aplikasi. Teknik pengumpulan data dilakukan

melalui pengisian kuesioner oleh pengguna yang terdiri dari 20 masyarakat umum di lingkungan kampus Universitas Komputer Indonesia.

Untuk megetahui tanggapan dan penilaian pengguna terhadap aplikasi *steganography* citra digital pada layanan mms berbasis *android*, telah disebar kuesioner kepada responden berdasarkan target pengguna, kemudian dilakukan penghitungan dengan menggunakan skala kontinum dimana data dianalisis dengan menghitung rata-rata jawaban berdasarkan skoring setiap jawaban dari responden yang kemudian berdasarkan rata-rata jawaban dari responden lalu kemudian dijumlahkan.

4.2.4.1 Data Kuesioner

Adapun data kuesioner dari aplikasi *steganography* adalah sebagai berikut:

1. Jumlah responden yang terdiri dari 20 orang masyarakat umum dengan umur 17 tahun keatas.
2. *Point* yang menjadi bahan penilaian adalah:
   a. Membantu mengamankan pesan pada citra digital.
   b. Tingkat keamanan yang baik.
   c. Memenuhi kebutuhan yang diharapkan.

Berikut adalah daftar pernyataan kepada masing-masing pengguna pada tabel 4.16

Tabel 4.16 Daftar Pertanyaan Kuesioner

Pengguna	Pernyataan
Masyarakat Umun	Pesan yang sifatnya rahasia harus dijaga keamanannya.
	Aplikasi *Steganography* membantu menjaga kerahasiaan pesan.
	Aplikasi *Steganogrphy* dengan baik *encode* dan *decode* pesan.
	Aplikasi *Steganography* memiliki tingkat keamanan yang baik.
	Key Management di dalam aplikasi berjalan dengan baik.
	Aplikasi Steganography mudah digunakan.

4.2.4.2 Pengolahan Data Kuesioner

Adapun skala jawaban dari kuesioner pada tabel 4.17

Tabel 4.17 Skala Jawaban Kuesioner

Jawaban	Skor
SS = Sangat Setuju	5
S = Setuju	4
R = Ragu-ragu	3
KS = Kurang Setuju	2
TS = Tidak Setuju	1

Berikut adalah hasil pengumpulan data dari masing-masing jawaban untuk setiap pengguna. Jawaban dari masing-masing responden dapat dilihat pada tabel 4.18

Tabel 4.18 Jawaban Kuesioner Mahasiswa

No	**Pernyataan**	**Jawaban**					**Total**
		SS	**S**	**R**	**KS**	**TS**	
1	Pesan yang sifatnya rahasia harus dijaga keamanannya.	10	8	2			**20**
2	Aplikasi *Steganography* membantu menjaga kerahasiaan pesan.	15	2	3			**20**
3	Aplikasi *Steganography* dengan baik *encode* dan *decode* pesan.	10	5	5			**20**
4	Aplikasi Steganography memiliki tingkat keamanan yang baik.	12	6	2			**20**
5	*Key Management* di dalam aplikasi berjalan dengan baik	7	9	4			**20**
6	Aplikasi Steganography mudah digunakan.	17	3				**20**

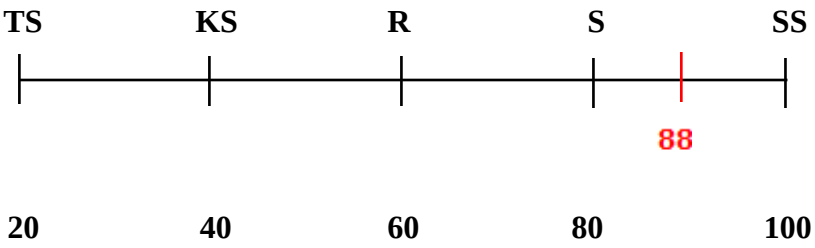
4.2.4.3 Hasil Pengolahan Kuesioner

Berikut merupakan data pengolahan kuesioner seperti pada tabel 4.19

Tabel 4.19 Hasil Pengolahan Data Kuesioner Pertanyaan (1)

Jawaban	Skor	Jumlah Responden	Jawaban Skor (Skor x Jumlah Responden)
SS = Sangat Setuju	5	10	50
S = Setuju	4	8	32
R = Ragu-ragu	3	2	6
KS = Kurang Setuju	2	0	0
TS = Tidak Setuju	1	0	0
		Total	88

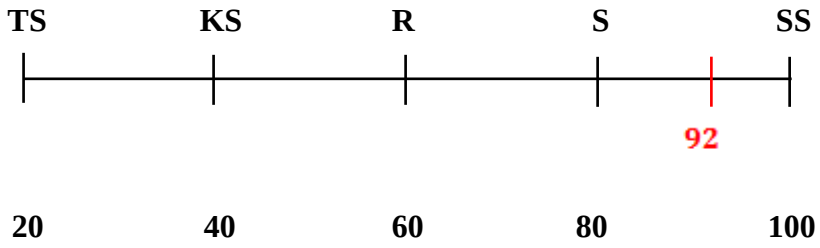
Jika semua responden (20 orang responden) menjawab SS, maka jumlah skor ideal (kriterium) untuk seluruh item adalah 100 (diperoleh dari x 20). Berdasarkan hasil penelitian, jumlah skor yang diperoleh adalah 88. Jadi berdasarkan data tersebut, maka tingkat persetujuan responden terhadap pertanyaan nomor (1) adalah (88 : 100) x 100% = 88% dari yang diharapkan (100%). Jadi berdasarkan pengolahan data yang diperoleh dari 20 responden, maka rata-rata 88 terletak pada daerah setuju. Secara kontinum dapat digambarkan sebagai berikut:



Tabel 4.20 Hasil Pengolahan Data Kuesioner Pertanyaan (2)

Jawaban	Skor	Jumlah Responden	Jawaban Skor (Skor x Jumlah Responden)
SS = Sangat Setuju	5	15	75
S = Setuju	4	2	8
R = Ragu-ragu	3	3	9
KS = Kurang Setuju	2	0	0
TS = Tidak Setuju	1	0	0
		Total	92

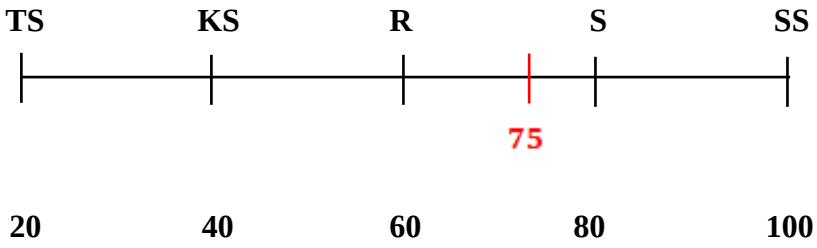
Jika semua responden (20 orang responden) menjawab SS, maka jumlah skor ideal (kriterium) untuk seluruh item adalah 100 (diperoleh dari 5 x 20). Berdasarkan hasil penelitian, jumlah skor yang diperoleh adalah 137. Jadi berdasarkan data tersebut, maka tingkat persetujuan responden terhadap pertanyaan nomor (2) adalah (92 : 100) x 100% = 92% dari yang diharapkan (100%). Jadi berdasarkan pengolahan data yang diperoleh dari 20 responden, maka rata-rata 92 terletak pada daerah setuju. Secara kontinum dapat digambarkan sebagai berikut:



Tabel 4.21 Hasil Pengolahan Data Kuesioner Pertanyaan (3)

Jawaban	Skor	Jumlah Responden	Jawaban Skor (Skor x Jumlah Responden)
SS = Sangat Setuju	5	10	50
S = Setuju	4	5	20
R = Ragu-ragu	3	5	15
KS = Kurang Setuju	2	0	0
TS = Tidak Setuju	1	0	0
		Total	75

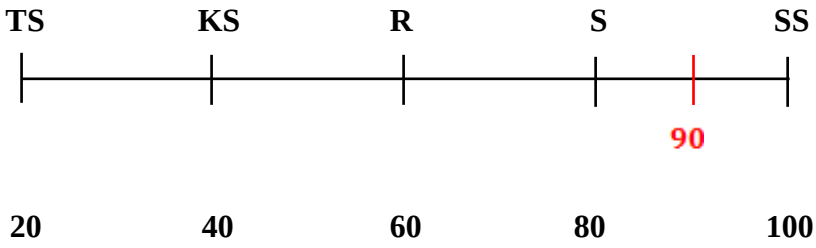
Jika semua responden (20 orang responden) menjawab SS, maka jumlah skor ideal (kriterium) untuk seluruh item adalah 100 (diperoleh dari 5 x 20). Berdasarkan hasil penelitian, jumlah skor yang diperoleh adalah 130. Jadi berdasarkan data tersebut, maka tingkat persetujuan responden terhadap pertanyaan nomor (3) adalah (75 : 100) x 100% = 75% dari yang diharapkan (100%). Jadi berdasarkan pengolahan data yang diperoleh dari 20 responden, maka rata-rata 75 terletak pada daerah setuju. Secara kontinum dapat digambarkan sebagai berikut:



Tabel 4.22 Hasil Pengolahan Data Kuesioner Pertanyaan (4)

Jawaban	Skor	Jumlah Responden	Jawaban Skor (Skor x Jumlah Responden)
SS = Sangat Setuju	5	12	60
S = Setuju	4	6	24
R = Ragu-ragu	3	2	6
KS = Kurang Setuju	2	0	0
TS = Tidak Setuju	1	0	0
		Total	90

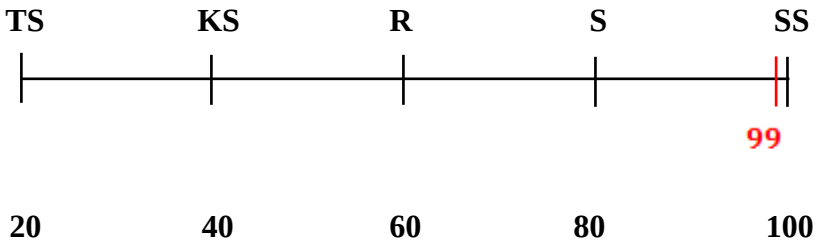
Jika semua responden (20 orang responden) menjawab SS, maka jumlah skor ideal (kriterium) untuk seluruh item adalah 150 (diperoleh dari 5 x 20). Berdasarkan hasil penelitian, jumlah skor yang diperoleh adalah 90. Jadi berdasarkan data tersebut, maka tingkat persetujuan responden terhadap pertanyaan nomor (4) adalah (90 : 100) x 100% = 90% dari yang diharapkan (100%). Jadi berdasarkan pengolahan data yang diperoleh dari 20 responden, maka rata-rata 90 terletak pada daerah setuju. Secara kontinum dapat digambarkan sebagai berikut:



Tabel 4.23 Hasil Pengolahan Data Kuesioner Pertanyaan (5)

Jawaban	Skor	Jumlah Responden	Jawaban Skor (Skor x Jumlah Responden)
SS = Sangat Setuju	5	9	45
S = Setuju	4	8	32
R = Ragu-ragu	3	4	12
KS = Kurang Setuju	2	0	0
TS = Tidak Setuju	1	0	0
		Total	99

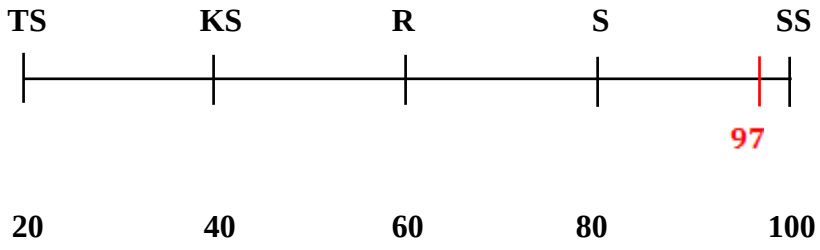
Jika semua responden (20 orang responden) menjawab SS, maka jumlah skor ideal (kriterium) untuk seluruh item adalah 100 (diperoleh dari 5 x 20). Berdasarkan hasil penelitian, jumlah skor yang diperoleh adalah 99. Jadi berdasarkan data tersebut, maka tingkat persetujuan responden terhadap pertanyaan nomor (5) adalah (99 : 100) x 100% = 99% dari yang diharapkan (100%). Jadi berdasarkan pengolahan data yang diperoleh dari 20 responden, maka rata-rata 99 terletak pada daerah setuju. Secara kontinum dapat digambarkan sebagai berikut:



Tabel 4.24 Hasil Pengolahan Data Kuesioner Pertanyaan (6)

Jawaban	Skor	Jumlah Responden	Jawaban Skor (Skor x Jumlah Respoinden)
SS = Sangat Setuju	5	17	85
S = Setuju	4	3	12
R = Ragu-ragu	3	0	0
KS = Kurang Setuju	2	0	0
TS = Tidak Setuju	1	0	0
		Total	97

Jika semua responden (20 orang responden) menjawab SS, maka jumlah skor ideal (kriterium) untuk seluruh item adalah 100 (diperoleh dari 5 x 20). Berdasarkan hasil penelitian, jumlah skor yang diperoleh adalah 97. Jadi berdasarkan data tersebut, maka tingkat persetujuan responden terhadap pertanyaan nomor (6) adalah (97 : 100) x 100% = 97% dari yang diharapkan (100%). Jadi berdasarkan pengolahan data yang diperoleh dari 20 responden, maka rata-rata 97 terletak pada daerah setuju. Secara kontinum dapat digambarkan sebagai berikut:



4.2.4.4 Kesimpulan Pengujian Beta

Berdasarkan jawaban hasil kuesioner yang diberikan kepada para responden, maka dapat disimpulkan bahwa :

1. Aplikasi steganogrphy ini dapat menjaga isi pesan yang dikirimkan.
2. Aplikasi steganogrphy menjaga kerahasiaan pesan.
3. Aplikasi steganogrphy dengan baik encode dan decode pesan.
4. Aplikasi steganogrphy memiliki tingkat keamanan yang baik.
5. Aplikasi steganogrphy sesuai dengan kebutuhan pengguna.

4.2.4.5 Kesimpulan Hasil Pengujian

Berdasarkan pada pengujian yang dilakukan maka didapatkan kesimpulan sebagai berikut :

1. Terdapat 2 kesimpulan dalam proses *embedding* dan *extracting* pada metode *haar wavelet* yaitu :
   a. Semakin besar panjang pesan *(byte)* semakin lama waktu proses *embedding* dan *extracting.*
   b. Semakin besar ukuran dimensi citra semakin lama waktu proses *embedding* dan *extracting*.
2. Waktu proses *extracting* lebih cepat dibandingkan dengan waktu proses *embedding.* Hal ini dikarenakan pada *embedding* terjadi proses membaca lokasi dan menyisipkan, sedangkan pada *extracting* proses yang terjadi adalah membaca lokasi penyisipan saja.
3. Kualitas citra yang dihasilkan setelah disisipi pesan masih tergolong baik dengan PSNR rata-rata **30,65** db.
4. Metode *haar wavelet* tidak dapat mengekstrak citra yang telah dimanipulasi.