BAB IV
IMPLEMENTASI DAN PENGUJIAN SISTEM

Bab ini menjelaskan implementasi dari hasil analisis, kebutuhan perangkat keras dan perangkat lunak yang diperlukan dalam mengembangkan sistem. Selain itu akan dibahas tentang pengujian aplikasi yang dibuat. Implementasi yang dilakukan mencakup data-data yang digambarkan dengan tampilan. Setelah implementasi maka dilakukan pengujian sistem yang baru dimana akan dilihat kekurangankekurangan pada aplikasi yang baru untuk pengembangan sistem selanjutnya.

4.1 Implementasi Sistem

Setelah sistem dianalisis dan didesain secara rinci maka selanjutnya akan dilakukan tahap implementasi. Implementasi Sistem adalah proses penggunaan/penerapan program aplikasi yang telah dibuat atau diperbaiki pada proses perancangan. Tujuan dari tahap impelentasi adalah untuk mengkonfirmasikan modul program perancangan pada para pelaku sistem sehingga pengguna dapat memberi masukan kepada pembangun sistem.

4.1.1 Kebutuhan Perangkat Keras

Berikut ini adalah spesifikasi dari perangkat keras server untuk mengimplementasikan aplikasi *digital signature* dan enkripsi untuk keamanan e-dokumen di PT Len Industri (Persero) dapat dilihat di tabel 4.1:

Tabel 4. 1 Spesifikasi Perangkat Keras Untuk Server

Perangkat Keras	**Keterangan**
Processor	Intel Dual Core 2.50 GHz
RAM	1 GHz
Harddisk	140 GHz
VGA	On Board 384 MB
Monitor	LCD 14 Inch, resolusi 1024 x 768 pixel
Mouse dan Keyboard	Standar

Berikut ini adalah spesifikasi dari perangkat keras client untuk mengimplementasikan aplikasi digital signature dan enkripsi untuk keamanan e-dokumen di PT Len Industri (Persero) dapat dilihat di tabel 4.2:

Tabel 4. 2 Spesifikasi Perangkat Keras Untuk Client

Perangkat Keras	**Keterangan**
Processor	Intel Dual Core 2.50 GHz
RAM	1 GHz
Harddisk	140 GHz
VGA	On Board 384 MB
Monitor	LCD 14 Inch, resolusi 1024 x 768 pixel
Mouse dan Keyboard	Standar

4.1.2 Kebutuhan Perangkat Lunak

Berikut ini adalah spesifikasi dari perangkat lunak untuk mengimplementasikan aplikasi *digital signature* dan enkripsi untuk keamanan e-dokumen di PT Len Industri (Persero) seperti pada tabel 4.3:

Tabel 4. 3 Spesifikasi Perangkat Lumak

Perangkat Lunak	**Keterangan**
Sistem Operasi	Windows 7
Bahasa Pemrograman	Java
Development Tools	NetBeans IDE 7.1
Perangkat Lunak Untuk Kompilasi	JDK (*Java Development Kit*) 63.1
Perangkat Lunak Untuk Menjalankan Aplikasi yang Dibangun	JRE (*Java Runtime Environtment*)

Simulasi pengiriman file dilakukan dengan cara yaitu di jaringan LAN (*Local Area Network*) PT Len Industri (Persero) .

4.1.3 Implementasi Antarmuka

Dari hasil analisis perancangan antarmuka yang telah pada bab sebelumnya, selanjutnya perancangan tersebut diimplementasikan dalam suatu antarmuka pada aplikasi digital signature dan enkripsi untuk keamanan e-dokumen di PT Len Industri (Persero) seperti pada tabel 4.4.

Tabel 4. 4 Tabel Form Perangkat Lunak

No	Menu	Use Case	File
1.	Home	-	Utama.Java
2.	Compose	*Generate key pair signature*, pilih dokumen, buat *signature*, *generate key pair encryption*, pilih pesan, *encrypt* pesan	Buatpesan.java, kunci ecdsa.java, ecdsaprivatkey.java, ecdsapublikkey.java, kuncieecc.java, myECDSA.java
3.	Inbox	-	Pesaninbox.java, inbox.java
4.	Sent	-	Pesansent.java

Berikut adalah tampilan antarmuka pada perangkat lunak *digital signature* dan enkripsi untuk keamanan e-dokumen di PT Len Industri (Persero).

1. Form antarmuka halaman utama aplikasi *digital signature* dan enkripsi.

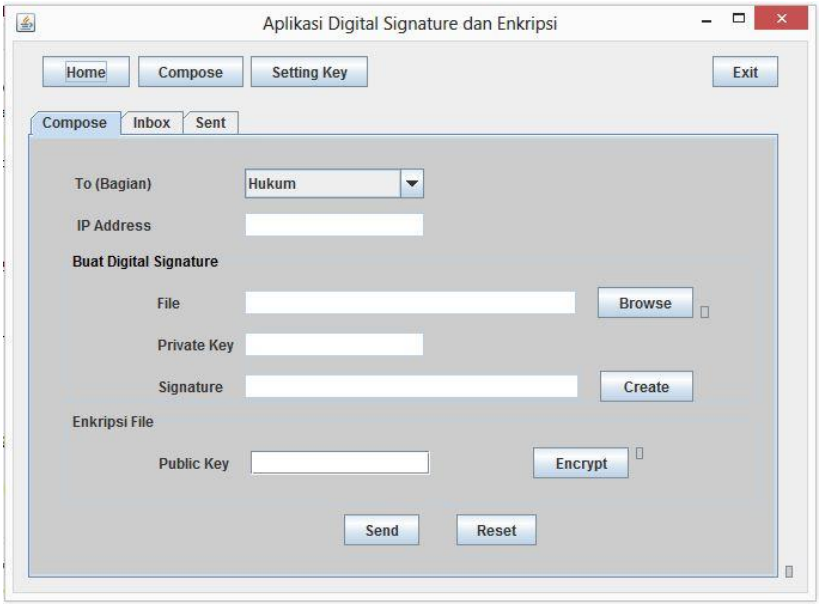
Gambar 4.1 merupakan tampilan halaman utama dari aplikasi *digital signature* dan enkripsi



Gambar 4. 1 Form Halaman Utama

2. Form antarmuka buat pesan

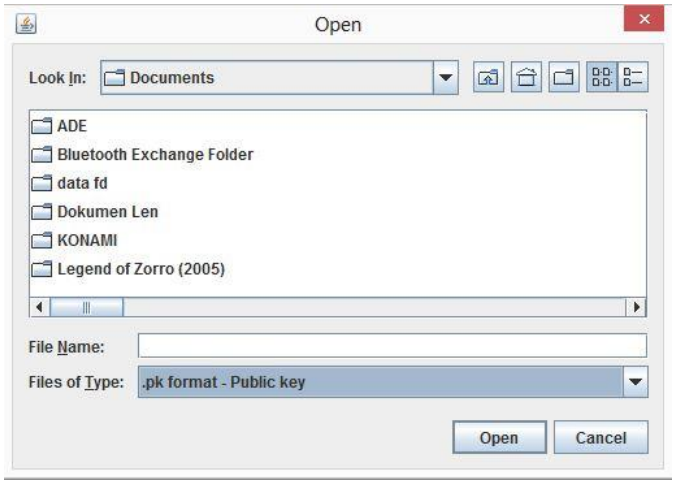
Berikut merupakan tampilan *compose* seperti pada gambar 4.2



Gambar 4. 2 Form Antarmuka Buat Pesan

3. Form dialog *file chooser save public key*

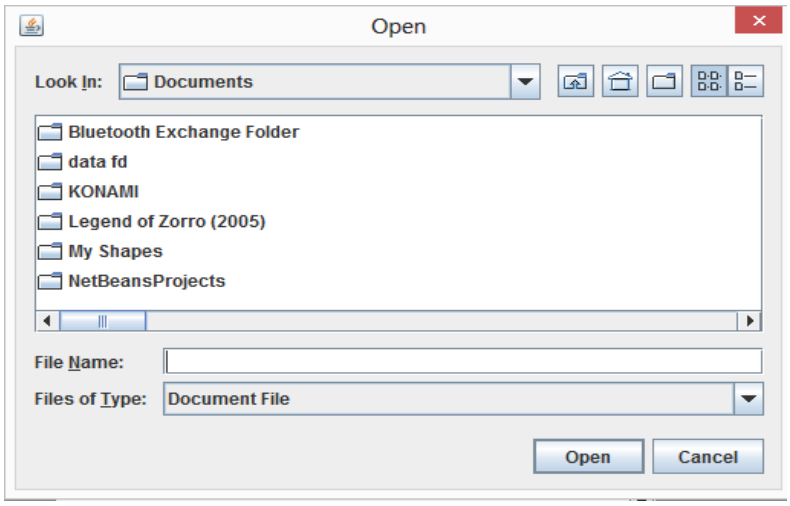
Gambar 4.3 berikut merupakan tampilan antarmuka ketika *user* akan menyimpan kunci publik.



Gambar 4. 3 Form Dialog File Chooser Save Public Key

4. Form dialog *file chooser* pilih *file* yang akan dibuatkan *digital signature*

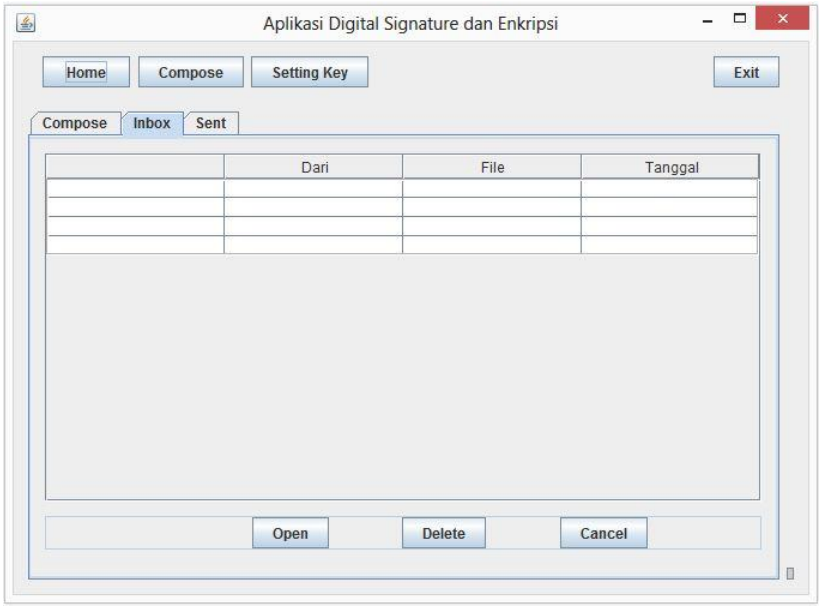
Gambar 4.4 merupakan tampilan antar muka ketika *user* akan memilih *file* untuk dibuat *digital signature.*



Gambar 4. 4 Form Dialog File Chooser Pilih File

5. Form antarmuka pesan masuk

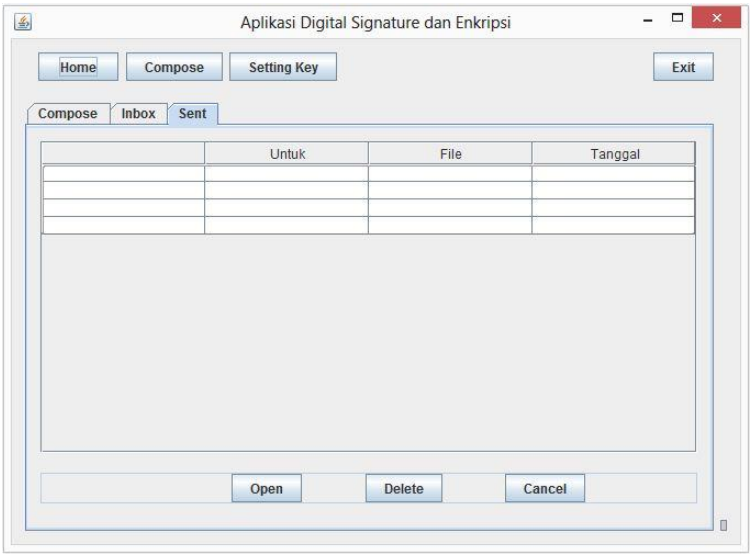
Gambar 4.5 merupakan tampilan antarmuka pesan masuk pada aplikasi *digital signature* dan enkripsi untuk keamanan e-dokumen di PT Len Industri.



Gambar 4. 5 Form Antar Muka Pesan Masuk

6. Form antarmuka pesan keluar

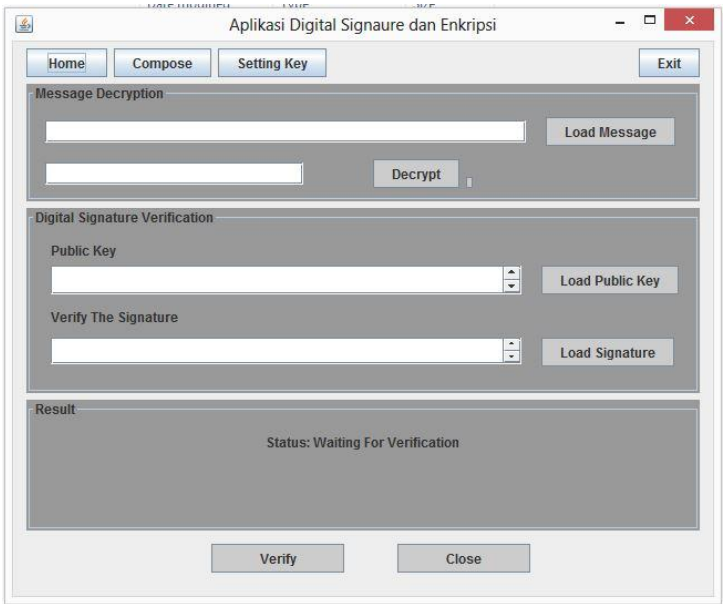
Gambar 4.6 merupakan tampilan antarmuka pesan keluar pada aplikasi *digital signature* dan enkripsi untuk keamanan e-dokumen di PT Len Industri.



Gambar 4. 6 Form Antarmuka Pesan Keluar

7. Form antarmuka dekripsi dan verifikasi

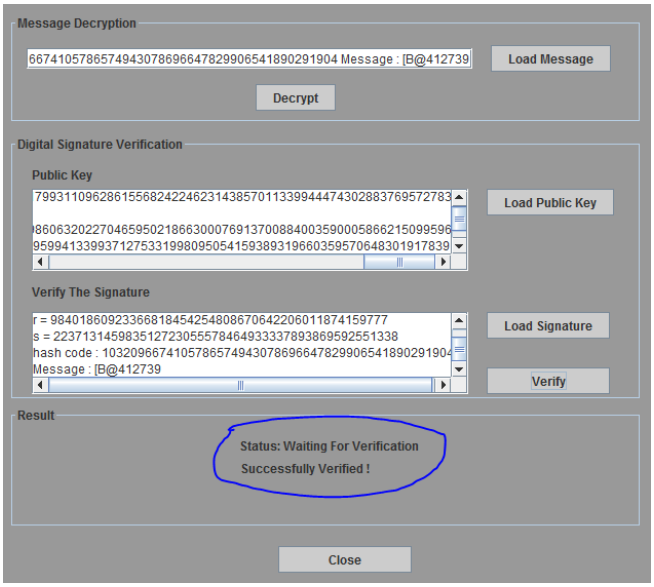
Gambar 4.7 merupakan tampilan antarmuka dekripsi dan verifikasi pesan pada aplikasi *digital signature* dan enkripsi untuk keamanan e-dokumen di PT Len Industri.



Gambar 4. 7 Form Antarmuka Dekripsi dan Verifikasi

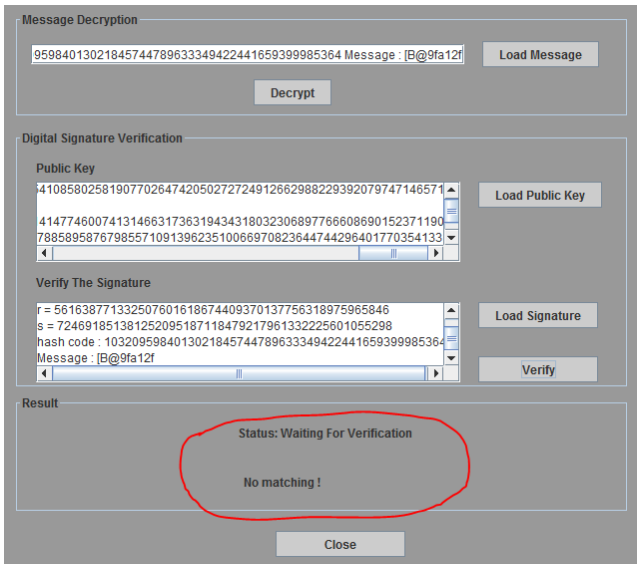
8. Form verifikasi

Jika verifikasi valid maka akan muncul pesan informasi seperti pada gambar 4.8



Gambar 4. 8 Form Verifikasi Valid

9. Form verifikasi tidak valid maka akan muncul pesan informasi seperti pada gambar 4.9



Gambar 4. 9 Form Verifikasi Tidak Valid

4.2 Pengujian Sistem

Setelah aplikasi *digital signature* dan enkripsi untuk keamanan e-dokumen di PT Len Industri (Persero) dibuat, langkah selanjutnya adalah pengujian sistem. Aplikasi *digital signature* dan enkripsi ini perlu diuji untuk menunjukan kesesuaian aplikasi dengan persoalan dan perancangan aplikasi tersebut.

4.2.1 Rencana Pengujian

Jenis pengujian yang dilakukan pada pembangunan aplikasi *digital signature* dan enkripsi ini adalah pengujian *white box* dan *black box.* Rencana pengujian yang akan dilakukan pada aplikasi *digital signature* dan enkripsi ini dapat dilihat pada tabel 4.5

Tabel 4. 5 Rencana Pengujian Alpha

No	Proses	Jenis Pengujian
1.	Pengujian *Generate Key Pair Signature*	*White Box*
2.	Pengujian *Create Signature*	*White Box*
3.	Pengujian *Verify Signature*	*White Box*
4.	Pengujian *Generate Key Pair Encryption*	*White Box*
5.	Pengujian *Encryption*	*White Box*
6.	Pengujian *Decryption*	*White Box*
7.	Pengujian *Compose Message*	*Black Box*
8.	Pengujian *Inbox*	*Black Box*
9.	Pengujian *Sent*	*Black Box*

4.2.2 Pengujian White Box

Pengujian *white box* digunakan untuk mengetahui kinerja logika yang dibuat pada sebuah perangkat lunak apakah berjalan dengan baik atau tidak. Pengujian *white box* akan digunakan pada *Elliptic Curve Digital Signature Algorithm* (ECDSA) dan ElGamal berbasis kurva eliptik, untuk mengukur kinerja logika berdasarkan *pseudocode* yang telah dibuat pada tahap analisis.

Berikut adalah tahap pengujian *white box*:

1. Langkah pertama ubah *pseudocode* menjadi flowchart.

2. Ubah *flowchart* menjadi *flowgraph* ke dalam bentuk yang lebih sederhana.
3. Tahap pengujian, dimana tahap pengujian ini dilakukan dengan 5 cara yaitu, menghitung *region*, menghitung *Cyclomatic Complexity*, menghitung *independent path*, menggunakan graph matriks, menghitung *predicate node*.

4.2.2.1 Pengujian Generate Key Pair Signature

Pengujian *generate key pair signature* berikut dimulai dengan merubah pseudocode menjadi flow chart, kemudian flowchart dirubah menjadi flowgraph. Tabel 4.6 merupakan *source code* pengujian *Generate Key Pair Signature.*

1. *Source code generate key pair signature*

Tabel 4. 6 *Source code generate key pair signature*

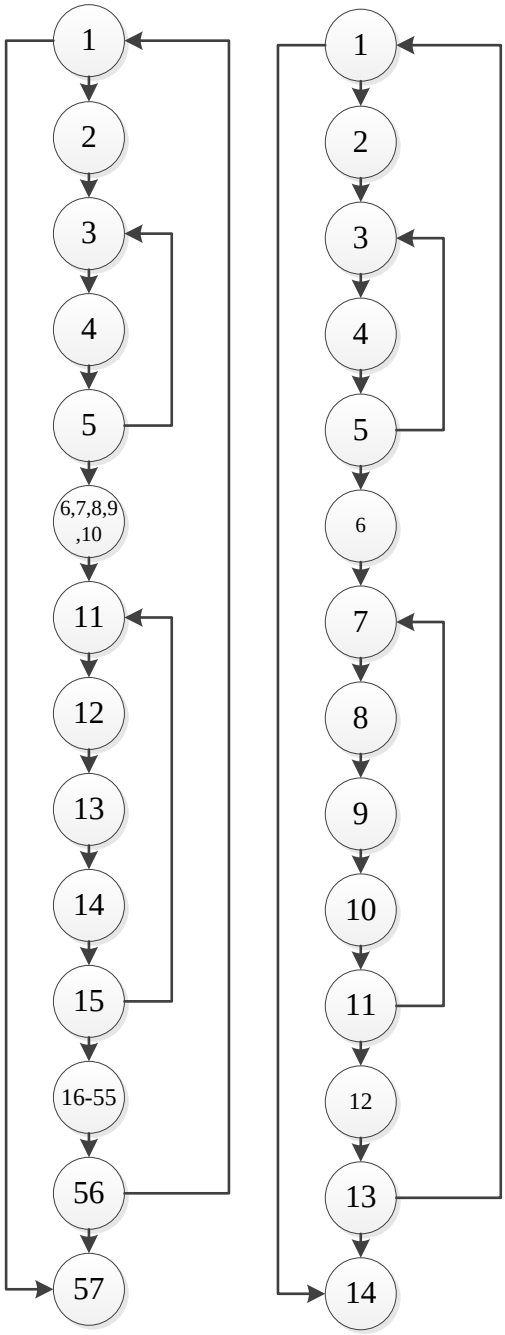
```
1.      BigInteger q=null;
2.      // generation of q :
3.      do {
4.          q = new BigInteger(N, new Random());
5.    } while (q.bitLength() != N
    ||!q.isProbablePrime(CERTAINTY));
6.
7.      // generation of p :
8.      BigInteger pm1 = null;
9.      BigInteger p = null;
10.     int iter =0;
11.     do {
12.         pm1 = q.multiply(new BigInteger(L - N, new Random()));
13.         p = pm1.add(new BigInteger("1"));
14.         iter++;
15.     } while (p.bitLength() != L ||
    !p.isProbablePrime(CERTAINTY));
16.
17.     // generation of g :
18.     pm1 = p.subtract(new BigInteger("1")); // pm1 = p-1
19.     BigInteger z = pm1.divide(q); // z = (p-1)/q
20.     BigInteger h = BigInteger.valueOf((long) H); // h = 2
21.     BigInteger g = h.modPow(z, p); // g = h^z mod p
22.
23.     // generation of x :
```

```
24.     BigInteger x = new BigInteger(N, new Random());
25.
26.     // generation of y :
27.     BigInteger y = g.modPow(x, p); // y = g^x mod p
28.
29.     System.out.println("\nChecking the parameters ... \n");
30.
31.     System.out.println("p size : " + p.bitLength());
32.     System.out.println("q size : " + q.bitLength());
33.     System.out.println("z size : " + z.bitLength());
34.
35.     Boolean test =
    (p.subtract(BigInteger.ONE).equals(q.multiply(z)))? true :
    false;
36.     System.out.println("p - 1 = qz ? : " + test);
37.
38.     System.out.println("p prime? : " +
    p.isProbablePrime(100));
39.     System.out.println("q prime? : " +
    q.isProbablePrime(100));
40.
41.     Boolean test2 = (   g.equals(h.modPow(z, p))       ) ?
    true:false;
42.     System.out.println("g = h^z mod p ? : " + test2);
43.
44.
45.     Boolean test3 = y.equals(g.modPow(x, p))?true:false;
46.     System.out.println("y = g^x mod p ? :" + test3);
47.
48.     System.out.println("The public key is : (p=" + p + ",
    q="+q+", g="+g+", y="+y+")");
49.     publicKey.setP(p);
50.     publicKey.setQ(q);
51.     publicKey.setG(g);
52.     publicKey.setY(y);
53.     System.out.println("The private key is : x="+x);
54.     privateKey.setX(x);
```

```
55.
56.     return iter;
57. }
```

2. *Flowgraph generate key pair signature*

Berikut merupakantahap perubahan *source code* di tabel 4.6 menjadi *flowgraph*



Gambar 4. *10 Flowgraph Generate Key Pair Signature*

3. Tahap Pengujian

   a. $V_{(G)}$ = Edge – Node + 2

      = 17– 14 + 2

      = 3 + 2

      = 5

   b. Independent Path

      Path 1 = 1-14

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

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

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

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

   c. Graph Matriks

Tabel 4. 7 *Graph Matriks Key Pair Signature*

Node	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Sum
1		1												1	1
2			1												0
3				1											0
4					1										0
5			1			1									1
6							1								0
7								1							0
8									1						0
9										1					0
10											1				0
11							1					1			1
12													1		0
13	1													1	1
14															
SUM+1															4+1

d. Predicate Node

V(G) = Jumlah node yg memiliki lebih dari 1 jalur + 1

= 4 + 1

= 5

4. Tabel hasil pengujian

Tabel 4. 8 Tabel Hasil Pengujian White Box Generate Key Pair Signature

NO	PATH	YANG DIHARAPKAN	HASIL YANG DI CAPAI	KESIMPULAN
1	1-14	Berhasil *generate key pair*	Berhasil *generate key pair*	[√] Diterima [ ] Ditolak
2	1-2-3-4-5-6-7-8-9-10-11-12-13-14	Berhasil membuat sepasang kunci *signature* dan kembali ke langkah awal	Berhasil membuat sepasang kunci *signature* dan kembali ke langkah awal	[√] Diterima [ ] Ditolak
3	1-2-3-4-5-3-4-5-6-7-8-9-10-11-12-13-14	Menentukan nilai q bilangan prima acak	Nilai q = bilangan prima acak	[√] Diterima [ ] Ditolak
4	1-2-3-4-5-6-7-8-9-10-11-7-8-9-10-11-12-13-14	Menentukan nilai p bilangan prima acak	Nilai p = bilangan prima acak	[√] Diterima [ ] Ditolak
5	1-2-3-4-5-6-7-8-9-10-11-12-13-1-2-3-4-5-6-7-8-9-10-11-	Apabila nilai tidak sesuai kembali lagi ke langkah awal	Menentukan kembali nilai q = bilangan acak prima	[√] Diterima [ ] Ditolak

	12- 13-14			

4.2.2.2 Pengujian *Create Signature*

Pengujian *Create Signature* berikut dimulai dengan merubah *source code* menjadi *flowgraph*. Tabel 4.9 merupakan *source code* pengujian *create signature*.

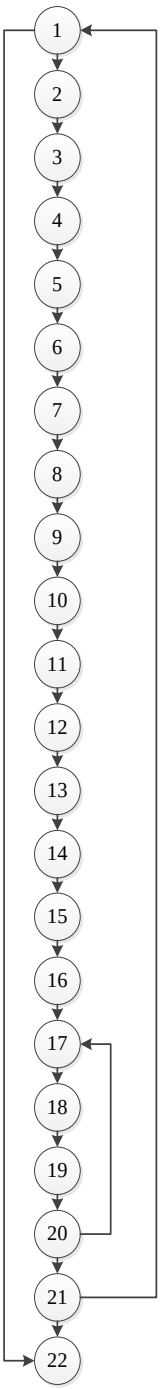
1. *Source code create signature*

Tabel 4. 9 *Source code create signature*

```
1.  BigInteger k = new BigInteger(N, new Random());
2.      BigInteger       r       =       publicKey.getG().modPow(k,
    publicKey.getP()).mod(publicKey.getQ());
3.
4.      // creation of s :
5.      BigInteger kInv = k.modInverse(publicKey.getQ()); // K^(-
    1) mod Q
6.      BigInteger xr = privateKey.getX().multiply(r); // X * R
7.
8.      BigInteger hash = null;
9.      try {
10.         hash = hashing(message);
11.     } catch (UnsupportedEncodingException ex) {
12. Logger.getLogger(myECDSA.class.getName()).log(Level.SEVERE,
    null, ex);
13.     }
14.     BigInteger hmXr = hash.add(xr); // H(m) + XR
15.     BigInteger tmp = kInv.multiply(hmXr); //
16.     BigInteger s = tmp.mod(publicKey.getQ());
17.     Message   newmessage   =   new   Message(new   Signature(r,s),
    message);
18.
19.     System.out.println("The  signature  is  :  (r="  +  r  +  ",
    s="+s+")");
20.     return newmessage;
21. }
```

2. *Flowgraph create signature*

Berikut merupakan tahap perubahan *source code create signature* dari tabel 4.9 menjadi *flowgraph* seperti terlihat pada gambar 4.11.



Gambar 4. 11 *Flowgraph Pengujian Create Signature*

3. Tahap Pengujian

a. $V_{(G)}$ = Edge – Node + 2

= 24– 22 + 2

= 2 + 2

= 4

b. Independent Path

Path 1 = 1-22

Path 2 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22

Path 3 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-17-18-19-20-21-22

Path 4 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22

c. Graph Matriks

Tabel 4. 10 Graph Matriks Create Signature

No	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	Sum
1		1																				1	1
2			1																				0
3				1																			0
4					1																		0
5						1																	0
6							1																0
7								1															0
8									1														0
9										1													0
10											1												0
11												1											0
12													1										0
13														1									0
14															1								0

15															1						0
16																1					0
17																	1				0
18																		1			0
19																			1		0
20																1				1	1
21	1																			1	1
22																					0
Sum + 1																					3+1=4

d. Predicate Node

V(G) = Jumlah node yg memiliki lebih dari 1 jalur + 1

= 3 + 1

= 4

4. Tabel hasil pengujian

Tabel 4. 11 Tabel Hasil Pengujian White Box Create Signature

NO	PATH	YANG DIHARAPKAN	HASIL YANG DI CAPAI	KESIMPULAN
1	1-22	Berhasil *create signature*	Berhasil *create signature*	[√] Diterima [ ] Ditolak
2	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22	Berhasil membuat *digital signature*	Berhasil membuat *digital signature*	[√] Diterima [ ] Ditolak
3	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-	Apabila gagal membuat signature kembali lagi memilih pesan	*Generate signature*	[√] Diterima [ ] Ditolak

	16-17-18-19-20-17-18-19-20-21-22	untuk men-*generate signature*		
4	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22	Apabila langkah-langkah sebelumnya tidak berhasil maka kembali ke langkah pertama	Apabila langkah-langkah sebelumnya tidak berhasil maka kembali ke langkah pertama	[√] Diterima [ ] Ditolak

4.2.2.3 Pengujian *Verify Signature*

Pengujian *Verify Signature* berikut dimulai dengan merubah *source code* menjadi *flowgraph*. Tabel 4.12 merupakan *pseudocode* pengujian *verify signature.*

1. *Source code verify signature*

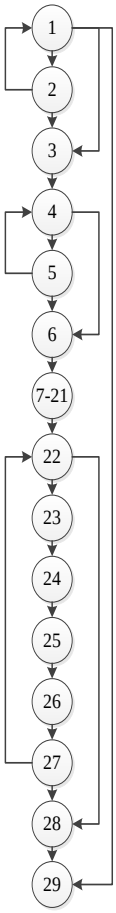
Tabel 4. 12 Source code verify signature

```
1.  if
    ((message.getSignature().getR().compareTo(BigInteger.ZERO)!=
    1)                                                          ||
    (message.getSignature().getR().compareTo(publicKey.getQ())!=  -
    1)) {
2.          return false;
3.      }
4.      if
    ((message.getSignature().getS().compareTo(BigInteger.ZERO)!=
    1)                                                          ||
```

```
    (message.getSignature().getS().compareTo(publicKey.getQ())!=  -
    1)) {
5.          return false;
6.      }
7.  BigInteger                          w                          =
    message.getSignature().getS().modInverse(publicKey.getQ());
8.      BigInteger hash = null;
9.      try {
10.         hash = hashing(message.getText());
11.     } catch (UnsupportedEncodingException ex) {
12.
    Logger.getLogger(myECDSA.class.getName()).log(Level.SEVERE,
    null, ex);
13.     }
14.     BigInteger u1 = (hash.multiply(w)).mod(publicKey.getQ());
15.     BigInteger                       u2                        =
    (message.getSignature().getR().multiply(w)).mod(publicKey.getQ
    ());
16.     BigInteger     gu1     =     (publicKey.getG().modPow(u1,
    publicKey.getP()));
17.     BigInteger     yu2     =     publicKey.getY().modPow(u2,
    publicKey.getP());
18.     BigInteger tmp = gu1.multiply(yu2);
19.     BigInteger tmpModP = tmp.mod(publicKey.getP());
20.     BigInteger v = tmpModP.mod(publicKey.getQ());
21.
22.     if (v.compareTo(message.getSignature().getR()) == 0) {
23.         System.out.println("true");
24.         return true;
25.     } else {
26.         System.out.println("false");
27.         return false;
28.     }
29. }
```

2. *Flowgraph verify signature*

   Berikut tahap *source code verify signature* menjadi *flowgraph*



Gambar 4. 12 Flow Graph Verifying Signature

3. Tahap Pengujian

a. $V_{(G)}$ = Edge – Node + 2

= 21 – 15 + 2

= 6 + 2

= 8

b. Independent Path

Path 1 = 1-29

Path 2 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29

Path 3 = 1-3-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29

Path 4 = 1-2-3-4-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29

Path 5 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-25-26-27-28-29

Path 6 = 1-2-1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-18-19-20-21-22-23-24-25-26-27-28-29

Path 7 = 1-2-3-4-5-4-5-6-7-8-9-10-11-12-13-14-15-16-18-19-20-21-22-23-24-25-26-27-28-29

Path 8= 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-18-19-20-21-22-23-24-25-26-27-22-23-24-25-26-27-28-29

c. Graph Matriks

Tabel 4. 13 Graph Matriks Verifying Signature

No	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	Sum
1		1	1																										1	2
2	1		1																											1
3				1																										0
4					1	1																								1
5				1		1																								1
6							1																							0
7								1																						0
8									1																					0
9										1																				0
10											1																			0
11												1																		0
12													1																	0
13														1																0
14															1															0
15																1														0
16																	1													0
17																		1												0
18																			1											0
19																				1										0
20																					1									0
21																						1								0
22																							1					1		1
23																								1						0
24																									1					0
25																										1				0
26																											1			0

27																				1					1		1
28																										1	0
29																											0
Sum + 1																											7+1

d. Predicate Node

V(G) = Jumlah node yg memiliki lebih dari 1 jalur + 1

= 7 + 1

= 8

4. Tabel hasil pengujian

Tabel 4. 14 Tabel Hasil Pengujian Verify Signature

NO	PATH	YANG DIHARAPKAN	HASIL YANG DI CAPAI	KESIMPULAN
1	1-29	Berhasil verifikasi *signature*	Berhasil verifikasi *signature*	[√] Diterima [ ] Ditolak
2	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-18-19-20-21-22-23-24-25-26-27-28-29	Berhasil membuat *digital signature*	Berhasil membuat *digital signature*	[√] Diterima [ ] Ditolak
3	1-3-5-6-7-8-9-10-11-12-13-14-15-16-18-19-20-21-22-23-24-25-26-27-28-29	Menghitung nilai R, apabila berhasil maka melanjutkan ke langkah berikutnya	Nilai R yang merupakan komponen *signature*	[√] Diterima [ ] Ditolak

4	1-2-3-4-6-7-8-9-10-11-12-13-14-15-16-18-19-20-21-22-23-24-25-26-27-28-29	Menghitung nilai S, apabila berhasil maka melanjutkan ke langkah berikutnya	Nilai S yang merupakan komponen *signature*	[√] Diterima [ ] Ditolak
5	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-18-19-20-21-22-25-26-27-28-29	Verifikasi berhsil	Verifikasi berhasil	[√] Diterima [ ] Ditolak
6	1-2-1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-18-19-20-21-22-23-24-25-26-27-28-29	Menghitung nilai R apabila tidak sesuai kembali ke langkah sebelumnya	Menghitung nilai R apabila tidak sesuai kembali ke langkah sebelumnya	[√] Diterima [ ] Ditolak
7	1-2-3-4-5-4-5-6-7-8-9-10-11-12-13-14-15-16-18-	Menghitung nilai S apabila tidak sesuai kembali ke langkah sebelumnya	Menghitung nilai S apabila tidak sesuai kembali ke langkah sebelumnya	[√] Diterima [ ] Ditolak

	19-20-21-22-23-24-25-26-27-28-29			
8	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-18-19-20-21-22-23-24-25-26-27-22-23-24-25-26-27-28-29	Apabila perhitungan tidak sesuai maka kembali ke langkah pertama	Apabila perhitungan tidak sesuai maka kembali ke langkah pertama	[√] Diterima [ ] Ditolak

4.2.2.4 Pengujian Generate Key Pair Encryption

Pengujian *Generate Key Pair Encryption* berikut dimulai dengan merubah pseudocode menjadi *flow chart*, kemudian *flowchart* dirubah menjadi *flowgraph*.

1. *Source code generate key pair encryption*

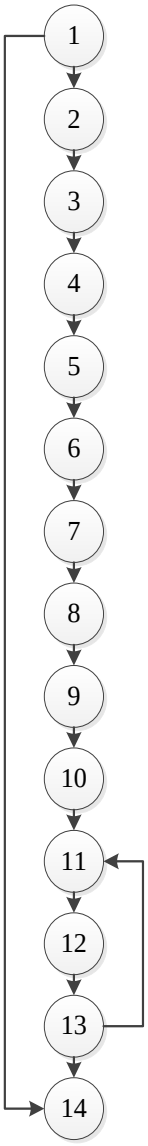
Tabel 4. 15 *Source Code Generate Key Pair Encryption*

```
1.  LinkedList kunciElGamal = new LinkedList();
2.          KurvaEliptik_Fp           ke           =           new
    KurvaEliptik_Fp(a,b,panjang_p/2);
3.
4.          // pilih alpha dan d
5.          Titik_KE_Fp alpha = ke.ambilSembarangTitik();
6.          BigInteger     d     =          new     BigInteger((new
    Random()).nextInt(panjang_p) ,new Random());
7.
8.          // hitung beta = d * alpha
9.          Titik_KE_Fp beta = ke.kaliSkalar(d, alpha);
10
11          kunciElGamal.add(new
```

```
    KunciPublikElGamalECC(ke,alpha,beta));
12          kunciElGamal.add(new KunciPrivatElgamal(d));
13          return kunciElGamal;
14      }
```

2. *Flowgraph generate key pair encryption*

Berikut tahap *source code* pada tabel 4.15 menjadi *flowgraph* seperti tampak pada gambar 4.13



Gambar 4. 13 *Flow Graph Generate Key Pair Encryption*

3. Tahap Pengujian

   a. $V_{(G)}$ = Edge – Node + 2

      = 15 – 14 + 2

      = 1 + 2

      = 3

   b. Independent Path

      Path 1 = 1-14

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

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

   c. Graph Matriks

      Tabel 4. 16 Graph Matriks Generate Key Pair Encryption

Node	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Sum
1		1												1	1
2			1												0
3				1											0
4					1										0
5						1									0
6							1								0
7								1							0
8									1						0
9										1					0
10											1				0
11												1			0
12													1		0
13											1			1	1
14															0
Sum+1															2+1=3

   d. Predicate Node

      V(G) = Jumlah node yg memiliki lebih dari 1 jalur + 1

      = 2 + 1

= 3

4. Tabel hasil pengujian

Tabel 4. 17 Tabel Hasil Pengujian Generate Key Encryption

NO	PATH	YANG DIHARAPKAN	HASIL YANG DI CAPAI	KESIMPULAN
1	1-14	Maju ke tahap berikutnya yaitu enkripsi file	Maju ke tahap berikutnya yaitu enkripsi file	[√] Diterima [ ] Ditolak
2	1-2-3-4-5-6-7-8-9-10-11-12-13-14	Berhasil melakukan generate key pair encryption	Berhasil melakukan generate key pair encryption	[√] Diterima [ ] Ditolak
3	1-2-3-4-5-6-7-8-9-10-11-12-13-14-11-12-13-14	Jika terjadi kesalahan maka kembali melakukan perhitungan kunci	Jika terjadi kesalahan maka kembali melakukan perhitungan kunci	[√] Diterima [ ] Ditolak

4.2.2.5 Pengujian *Encryption*

Pengujian *Encryption* berikut dimulai dengan merubah pseudocode menjadi *flow chart*, kemudian *flowchart* dirubah menjadi *flowgraph*.

1. *Source code encryption*

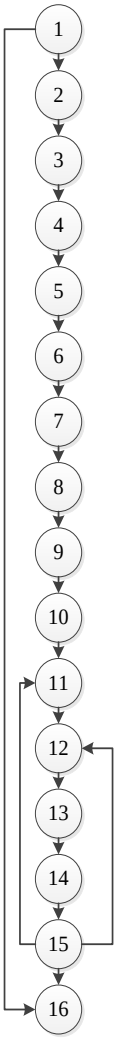
Tabel 4. 18 Source Code Encryption

```
1. KurvaEliptik_Fp kurvaeliptik = kunciPublik.getKurvaeliptik();
2.         Titik_KE_Fp alpha = kunciPublik.getAlpha();
3.         Titik_KE_Fp beta = kunciPublik.getBeta();
4.         //BigInteger beta = kunciPublik.getBeta();
5.
6.         int n = kurvaeliptik.getFp().getP().bitLength();
7.
8.         BigInteger r = new BigInteger((new
   Random()).nextInt(n-1),new Random());
9.
```

```
10
11          Titik_KE_Fp c1 = kurvaeliptik.kaliSkalar(r, alpha);
12          Titik_KE_Fp c2 = kurvaeliptik.kaliSkalar(r, beta);
13          c2 = kurvaeliptik.tambah(teksasli,c2);
14
15          return new TeksSandiElGamalECC(c1,c2);
16      }
```

2. Tahap perubahan *source code* menjadi *flowgraph*

Berikut tahap *souce code* pada tabel 4.18 menjadi *flowgraph* seperti pada gambar 4.14



Gambar 4. 14 Pengujian Encryption

5. Tahap Pengujian

   a. $V_{(G)}$ = Edge – Node + 2

      = 18– 16 + 2

      = 2 + 2

      = 4

   b. Independent Path

      Path 1 = 1-16

      Path 2 = 1-2-1-3-4-5-6-7-8-9-10-11-12-13-14-15-16

      Path 3 = 1-2-1-3-4-5-6-7-8-9-10-11-12-13-14-15-11-12-13-14-15-16

      Path 4 = 1-2-1-3-4-5-6-7-8-9-10-11-12-13-14-15-12-13-14-15-16

   c. Graph Matriks

Tabel 4. 19 Graph Matriks Encryption

Node	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Sum
1		1														1	1
2			1														0
3				1													0
4					1												0
5						1											0
6							1										0
7								1									0
8									1								0
9										1							0
10											1						0
11												1					0
12													1				0
13														1			0
14												1			1		1
15											1					1	1
16																	0

Sum + 1	3+1=4

d. Predicate Node

V(G) = Jumlah node yg memiliki lebih dari 1 jalur + 1

= 3 + 1

= 4

6. Tabel hasil pengujian

Tabel 4. 20 Tabel Hasil Pengujian Eckripsi

NO	PATH	YANG DIHARAPKAN	HASIL YANG DI CAPAI	KESIMPULAN
1	1-16	Berhasil melakukan enkripsi file	Berhasil melakukan enkripsi file	[√] Diterima [ ] Ditolak
2	1-2-1-3-4-5-6-7-8-9-10-11-12-13-14-15-16	Berhasil melakukan tahap-demi tahap enkripsi	Berhasil melakukan tahap demi tahap enkripsi	[√] Diterima [ ] Ditolak
3	1-2-1-3-4-5-6-7-8-9-10-11-12-13-14-15-11-12-13-14-15-16	Jika C1 tidak sesuai maka kembali ke langkah sebelumnya yaitu menghitung C1	Berhasil menghitung nilai C1	[√] Diterima [ ] Ditolak
4	1-2-1-3-4-5-6-7-8-9-10-11-12-13-14-15-12-13-14-15-16	Jika C2 tidak sesuai maka kembali ke langkah sebelumnya yaitu menghitung C2	Berhasil menghitung nilai C2	[√] Diterima [ ] Ditolak

4.2.2.6 Pengujian *Decryption*

Pengujian *Decryption* berikut dimulai dengan merubah *source code* menjadi *flowgraph*.

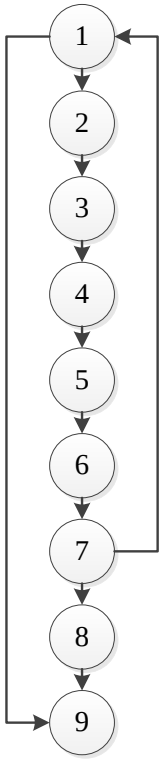
1. *Source code decryption*

Tabel 4. 21 *Source Code Decryption*

```
1.  Titik_KE_Fp c1 = tekssandi.getC1();
2.          Titik_KE_Fp c2 = tekssandi.getC2();
3.          BigInteger d = kunciPrivat.getD();
4.          KurvaEliptik_Fp             kurvaeliptik             =
    kuncipublik.getKurvaeliptik();
5.
6.          c1 = kurvaeliptik.kaliSkalar(d, c1);
7.          return kurvaeliptik.tambah(c2, kurvaeliptik.neg(c1));
8.
9.      }
```

2. Tahap perubahan *source code* menjadi *flowgraph*

Berikut tahap *souce code* pada tabel 4.21 menjadi *flowgraph* seperti pada gambar 4.15



Gambar 4. 15 Flowgraph Dekripsi

3. Tahap pengujian
   a. $V_{(G)}$ = Edge – Node + 2
      = 10– 9 + 2
      = 1 + 2
      = 3
   b. Independent Path
      Path 1 = 1-9
      Path 2 = 1-2-3-4-5-6-7-8-9
      Path 3 = 1-2-3-4-5-6-7-1-2-3-4-5-6-7-8-9
   c. Graph Matriks

Tabel 4. 22 Graph Matriks Dekripsi

Node	1	2	3	4	5	6	7	8	9	Sum
1		1							1	1
2			1							0
3				1						0
4					1					0
5						1				0
6							1			0
7	1							1		1
8									1	0
9										0
Sum+1										2+1=3

   d. Predicate Node
      V(G) = Jumlah node yg memiliki lebih dari 1 jalur + 1
      = 2 + 1
      = 3

4. Tabel hasil pengujian

Tabel 4. 23 Tabel Hasil Pengujian Dekripsi

NO	PATH	YANG DIHARAPKAN	HASIL YANG DI CAPAI	KESIMPULAN
1	1-9	Berhasil melakukan dekripsi file	Berhasil melakukan dekrispi file	[√] Diterima [ ] Ditolak
2	1-2-3-4-5-6-7-8-9	Berhasil melakukan tahap-demi tahap dekripsi	Berhasil melakukan tahap demi tahap dekripsi	[√] Diterima [ ] Ditolak
3	1-2-3-4-5-6-7-1-2-3-4-5-6-7-8-9	Jika C2 yaitu file yang terenkripsi tidak sesuai maka kembali ke langkah pertama	Berhasil menghitung nilai C2	[√] Diterima [ ] Ditolak

4.2.3 Pengujian *Black Box*

Pengujian *black box* bertujuan untuk mengukur kinerja dari perangkat lunak apakah fungsinya dapat berjalan dengan baik atau tidak. Berikut pengujian black box yang akan dilakukan

4.2.4 Kasus dan Hasil Pengujian

Berdasarkan rencana pengujian yang disusun, maka dilakukan pengujian seperti yang dicantumkan dibawah ini.

1. Pengujian *Compose Message*

Pengujian data benar *compose message* terlihat pada tabel 4.12

Tabel 4. 24 Pengujian Data Benar Compose Message

Kasus dan Hasil Uji (Data Benar)			
Aktivitas yang	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**

dilakukan			
Mengisikan alamat IP	Berhasil mengisi alamat IP	Berhasil mengisi alamt IP	[√] Diterima [ ] Ditolak
Membuat sepasang kunci *signature*, menekan tombol *generate*	Menampilkan kunci privat dan kunci publik	Menampilkan kunci privat dan kunci publik	[√] Diterima [ ] Ditolak
Memilih file, dengan menekan tombol *Browse*	Berhasil memilih file	Berhasil memilih file	[√] Diterima [ ] Ditolak
Membuat *signature* dengan menekan tombol *create*	Berhasil membuat *signature*	*Signature* dapat tampil	[√] Diterima [ ] Ditolak
Menbuat sepasang kunci enkripsi	Berhasil membuat sepasang kunci	Berhasil membuat sepasang kunci	[√] Diterima [ ] Ditolak
Memilih pesan yang akan di enkripsi, dengan menekan tombol “browse pesan”	Berhasil memilih pesan untuk di enkripsi	Berhasil memilih pesan untuk di enkripsi	[√] Diterima [ ] Ditolak
Meng-enkripsi pesan dengan menekan tombol encrypt	Berhasil enkripsi pesan	Berhasil enkripsi pesan	[√] Diterima [ ] Ditolak
Klik Tombol Send	Dokumen tersimpan di data storage	Dokumen disimpan dalam storage	[√] Diterima [ ] Ditolak
Klik Tombol	Dokumen tidak	Dokumen tidak	[√] Diterima

Cancel	jadi dikirim	jadi dikirim	[ ] Ditolak

Pengujian data salah *Compose Message* terlihat pada tabel 4.25

Tabel 4. 25 Pengujian Data Salah Compose Message

Kasus dan Hasil Uji (Data Salah)			
Aktivitas yang dilakukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Tidak memasukan alamat IP	Menampilkan pesan "Belum memasukan IP"	Menampilkan pesan "Belum memasukan IP"	[√] Diterima [ ] Ditolak
Tidak membuat sepasang kunci *signature*, lalu menekan tombol "sign"	Menampilkan pesan "Belum *generate* kunci *signature*"	Menampilkan pesan "Belum *generate* kunci"	[√] Diterima [ ] Ditolak
File kosong	Menampilkan pesan "Pilih pesan terlebih dahulu".	Menampilkan pesan "Pilih pesan terlebih dahulu".	[√] Diterima [ ] Ditolak
File tidak sesuai, atau file selain dokumen	Menampilkan pesan "File tidak sesuai".	Menampilkan pesan "File tidak sesuai".	[√] Diterima [ ] Ditolak
Tidak membuat sepasang kunci enkripsi, lalu menekan tombol "enkripsi"	Menampilkan pesan "Belum *generate* kunci enkripsi"	Menampilkan pesan "Belum *generate* kunci enkripsi"	[√] Diterima [ ] Ditolak
Menekan tombol "send" tapi belum	Menampilkan pesan "Tidak ada	Menampilkan pesan "Tidak ada	[√] Diterima [ ] Ditolak

melakukan proses *digital signature* dan enkripsi	file untuk dikirim"	file untuk dikirim"	

2. Pengujian *Inbox*

Pengujian data benar *Inbox* terlihat pada tabel 4.8

Tabel 4. 26 Pengujian Data Benar Inbox

Kasus dan Hasil Uji (Data Benar)			
Aktivitas yang dilakukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Memilih pesan yang ada pada tabel Inbox, kemudian menekan tombol "open"	Berhasil membuka pesan	Berhasil membuka pesan	[√] Diterima [ ] Ditolak
Memilih pesan yang ada pada tabel Inbox, kemudian menekan tombol "delete"	Berhasil menghapus pesan	Berhasil menghapus pesan	[√] Diterima [ ] Ditolak
Memilih pesan yang ada pada tabel Inbox, kemudian menekan tombol "cancel"	Berhasil keluar dari form Inbox	Berhasil keluar dari form Inbox	[√] Diterima [ ] Ditolak
Membuka pesan	Menampilkan	Menampilkan	[√] Diterima

pada tabel Inbox	form dekripsi pesan dan verifikasi *signature*	form dekripsi pesan dan verifikasi *signature*	[ ] Ditolak
Memasukan kunci publik enkripsi, kemudian menekan tombol *"*decrypt"	Pesan berhasil di dekripsi	Pesan berhasil di dekripsi	[√] Diterima [ ] Ditolak
Memasukan kunci privat *signature*, kemudian menekan tombol "verify"	*Signature* berhasil di verifikasi	*Signature* berhasil di verifikasi	[√] Diterima [ ] Ditolak

Pengujian data salah *Inbox* terlihat pada tabel 4.9

Tabel 4. 27 Pengujian Data Salah Inbox

Kasus dan Hasil Uji (Data Salah)			
Aktivitas yang dilakukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Pada form buka pesan, tidak memasukan kunci publik untuk dekrispi kemudian menekan tombol "decrypt"	Menampilkan pesan "belum memasukan kunci publik"	Menampilkan pesan "belum memasukan kunci publik"	[√] Diterima [ ] Ditolak
Pada form buka pesan, memasukan kunci privat yang salah untuk	Menampilkan pesan "kunci enkripsi tidak sesuai"	Menampilkan pesan "kunci enkripsi tidak sesuai"	[√] Diterima [ ] Ditolak

dekripsi kemudian menekan tombol "decrypt"			
Pada form buka pesan, memasukan kunci publik yang salah untuk verifikasi kemudian menekan tombol "verify"	Menampilkan pesan "kunci *signature* tidak sesuai"	Menampilkan pesan "kunci *signature* tidak sesuai"	[√] Diterima [ ] Ditolak

3. Pengujian *sent*

Tabel 4. 28 Pengujian Data Benar Sent

Kasus dan Hasil Uji (Data Benar)			
Aktivitas yang dilakukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Memilih pesan yang ada pada tabel Sent, kemudian menekan tombol "open"	Berhasil membuka pesan	Berhasil membuka pesan	[√] Diterima [ ] Ditolak
Memilih pesan yang ada pada tabel Sent, kemudian menekan tombol "delete"	Berhasil menghapus pesan	Berhasil menghapus pesan	[√] Diterima [ ] Ditolak
Memilih pesan yang	Berhasil keluar dari form Sent	Berhasil keluar dari form Sent	[√] Diterima [ ] Ditolak

ada pada tabel Sent, kemudian menekan tombol "cancel"			

4.2.2 Pengujian Beta

Pengujian Beta merupakan pengujian yang dilakukan secara objektif dimana diuji secara langsung ke lapangan yaitu instansi yang bersangkutan mengenai kepuasan pengguna dengan kandungan poin yaitu pemenuhan kebutuhan dari tujuan awal pembangunan aplikasi *digital signature* dan enkripsi tersebut. Pengujian beta dilakukan melalui sebuah teknik pengambilan data, yaitu melalui wawancara.

4.2.2.1 Wawancara Pengguna

Tempat wawancara yaitu PT. Len Industri (Persero). Adapun wawancara pengujian beta untuk Instansi tersebut yaitu :

1. Apakah aplikasi yang dibangun memudahkan pengguna dalam melakukan pengiriman dokumen?
2. Apakah aplikasi *digital signature* dan enkripsi ini cukup akurat dalam mengetahui keontetikan dan kerahasiaan dokumen yang dikirim?
3. Apakah anda kesulitan menggunakan aplikasi *digital signature* dan enkripsi ini?
4. Apakah aplikasi *digital signature* dan enkripsi ini memberikan keefisienan waktu dalam pengiriman dokumen antar bagian dalam perusahaan anda?
5. Apakah tampilan dari aplikasi *digital signature* dan enkripsi ini menarik?
6. Apakah aplikasi *digital signature* dan enkripsi ini sudah *user friendly*?

Dari wawancara tersebut, menghasilkan jawaban dari narasumber. Rincian hasil jawaban yang diperoleh dari wawancara kepada narasumber untuk pertanyaan pertama terdapat pada tabel 4.8.

Tabel 4. 29 Hasil Wawancara

1. Apakah aplikasi yang dibangun memudahkan pengguna dalam melakukan pengiriman dokumen?

Jawaban Narasumber	Aplikasi yang dibangun memudahkan dalam pengiriman dokumen, karena tinggal memasukan alamat IP dan mengikuti langkah-langkah membuat *digital signature* dan mengenkripsi pesan untuk kemudian dikirimkan ke alamat IP tujuan
2. Apakah aplikasi *digital signature* dan enkripsi ini cukup akurat dalam mengetahui keontetikan dan kerahasiaan dokumen yang dikirim?	
Jawaban Narasumber	Seperti fungsi yang ditawarkan, saya rasa aplikasi *digital signature* dan enkripsi ini bisa mengetahui keaslian/ otentikasi pengirim dan kerahasian dokumen yang dikirimkan. Tapi mungkin dibutuhkan pengujian lebih lanjut untuk membuktikan keakuratannya.
3. Apakah anda kesulitan menggunakan aplikasi *digital signature* dan enkripsi ini?	
Jawaban Narasumber	Pertama kali melihat aplikasi ini cukup bingung juga, tapi setelah dijelaskan saya mengerti maksud aplikasi ini seperti apa. Untuk tambahan mungkin tampilannya diperbaiki.
4. Apakah aplikasi *digital signature* dan enkripsi ini memberikan keefisienan waktu dalam pengiriman dokumen antar bagian dalam perusahaan anda?	
Jawaban Narasumber	Cukup efisien seperti halnya mengirim email, kalau lama mungkin biasa ada faktor-faktor teknik yang mempengaruhi.
5. Apakah tampilan dari aplikasi *digital signature* dan enkripsi ini menarik?	
	Untuk tampilan mungkin belum begitu bagus dan cenderung membosankan, tapi untuk fungsi-fungsi utamanya sudah jelas jadi tidak membuat bingung.
6. Apakah aplikasi *digital signature* dan enkripsi ini sudah *user friendly*?	
Jawaban Narasumber	Iya tidak memebingungkan, fungsi utamanya sudah jelas.

4.2.2.2 Kesimpulan Pengujian Beta

Dari pengujian beta yang telah dilakukan dari hasil wawancara di lapangan didapat kesimpulan bahwa aplikasi yang dibangun dinilai dapat membantu dalam pengiriman dokumen antar bagian unit kerja dengan memakai aplikasi *digital signature* dan enkripsi, sesuai kebutuhan, menjadi suatu sarana yang bermanfaat, mengefisienkan waktu, info yang disampaikan cepat, mudah dan akurat.