BAB 4

Implementasi dan Pengujian Sistem

4.1 Implementasi Sistem

Setelah sistem dianalisis dan didesain secara rinci, maka selanjutnya akan menuju tahap implementasi. Implementasi sistem bertujuan untuk menjelaskan tentang manual modul kepada pengguna yang akan menggunakan sistem, sehingga dapat memberikan masukan kepada pembangun sistem. Tahapan-tahapan implementasi tersebut berupa implementasi perangkat lunak, implementasi perangkat keras, implementasi antarmuka, implementasi program dan pengujian program.

4.1.1 Implementasi Perangkat Lunak

Implementasi perangkat lunak merupakan proses instalasi perangkat lunak, sehingga dapat beroperasi dengan benar. Berikut ini pada adalah kebutuhan perangkat lunak yang digunakan terlihat pada Tabel 4.1.

Tabel 4.1 Lingkungan Implementasi Perangkat Lunak

No	Perangkat Lunak	Keterangan
1.	Sistem Operasi	Microsoft Windows XP Professional SP3 atau lebih
2.	*Framework*	Microsoft .NET Framework 3.5
3.	Aplikasi	Doy-ENKOM

4.1.2 Implementasi Perangkat Keras

Implementasi perangkat keras merupakan realisasi dari analisis dan perancangan kebutuhan perangkat keras. Berikut ini adalah kebutuhan perangkat keras yang digunakan terlihat pada Tabel 4.2.

Tabel 4.2 Lingkungan Implementasi Perangkat Keras

No	Perangkat Keras	Spesifikasi
1.	Prosessor	AMD E350 1,6 GHz atau lebih
2.	Monitor	Monitor
3.	VGA	VGA Card On-Board
4.	Memori	Memori DDR1 512 Mb atau lebih
5.	Hard disk	Kapasitas bebas minimal 1 Gb
6.	*Optical Drive*	CD-Rom atau lebih
7.	*Keyboard*	*Keyboard*
8.	*Mouse*	*Mouse*

4.1.3 Implementasi Antarmuka Sistem

Implementasi antarmuka sistem merupakan realisasi desain tampilan antarmuka dari setiap menu yang telah dirancang sebelumnya kedalam bahasa pemrograman. Berikut ini adalah implementasi antarmuka sistem yang terlihat pada Tabel 4.3.

Tabel 4.3 Implementasi Antarmuka Sistem

Nama Menu	Deskripsi	Diterapkan Pada *File*
Login	Digunakan sebagai tampilan menu *login* dan pemvalidasi data masukan dari *login*.	Login.vb
Utama	Digunakan sebagai tampilan awal dan menampilkan sub menu dari aplikasi.	Utama.vb
Enkripsi *File*	Digunakan sebagai tampilan menu enkripsi *file* dan pengolah *file* yang akan dienkripsi.	Enkripsi.vb
Dekripsi *File*	Digunakan sebagai tampilan menu dekripsi *file* dan pengolah *file* yang akan didekripsi.	Dekripsi.vb
Kompresi *File*	Digunakan sebagai tampilan menu kompresi *file* dan	Kompresi.vb

Nama Menu	Deskripsi	Diterapkan Pada *File*
	pengolah *file* yang akan dikompresi.	
Dekompresi *File*	Digunakan sebagai tampilan menu dekompresi *file* dan pengolah *file* yang akan didekompresi.	Dekompresi.vb
Pengaturan	Digunakan sebagai tampilan menu ganti *password* dan pemvalidasi data masukan ganti *password login.*	GantiPwd.vb
Petunjuk	Digunakan sebagai tampilan menu petunjuk yang berisi informasi petunjuk penggunaan aplikasi.	Petunjuk.vb
Tentang	Digunakan sebagai tampilan menu tentang yang berisi informasi tentang aplikasi	Tentang.vb

4.2 Pengujian Sistem

Pengujian sistem merupakan hal terpenting yang bertujuan untuk menemukan kesalahan-kesalahan atau kekurangan-kekurangan pada perangkat lunak yang diuji. Pengujian bermaksud untuk mengetahui perangkat lunak yang dibuat sudah memenuhi kriteria yang sesuai dengan tujuan perancangan perangkat lunak tersebut.

Pengujian yang dilakukan ada dua tahapan yaitu pengujian *alpha* dan pengujian *beta.* Pengujian *alpha* digunakan untuk menguji sistem dengan metode yang digunakan adalah metode *white-box*. Pengujian *white-box* digunakan untuk meyakinkan semua perintah dan kondisi dieksekusi secara minimal. Pengujian *beta* digunakan untuk memeriksa tingkat keakuratan sistem, pada pengujian *beta* akan dilakukan uji performansi sistem.

4.2.1 Pengujian *Alpha*

Pengujian *alpha* digunakan untuk menguji aplikasi dengan berbagai kondisi, metode yang digunakan pada pengujian *alpha* adalah metode *white-box*. Pengujian *white-box* digunakan untuk meyakinkan semua perintah dan kondisi pada aplikasi Doy-ENKOM dieksekusi secara minimal.

Pengujian *white-box* yang pada aplikasi Doy-ENKOM menggunakan dua *tools* yaitu *flow graph* yang digunakan untuk menggambarkan alur dari algoritma dan *graph matrix* yang digunakan untuk menggenerasi *flow graph.* Adapun pengujian *white-box* pada aplikasi Doy-ENKOM adalah sebagai berikut:

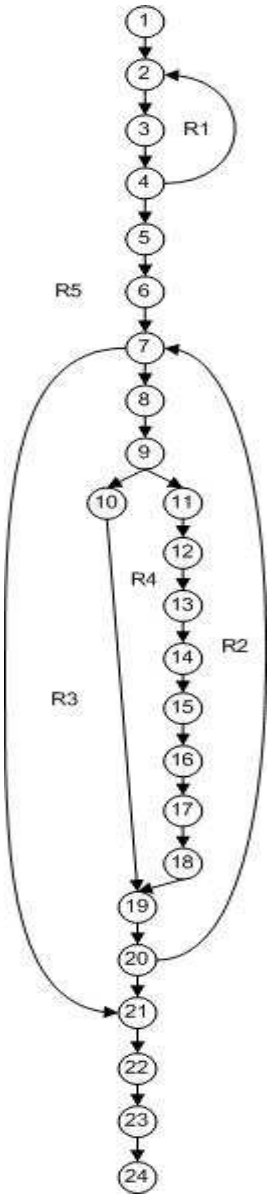
1. Pengujian Kompresi LZW

Berikut ini adalah algoritma kompresi LZW yang akan diuji:

```
1. iNextCode = 256
2. For i As Integer = 0 To TABLE_SIZE – 1
3. iaCode_Value(i) = -1
4. Next i
5. iStringCode = ReadByte()
6. iCharacter = ReadByte()
7. While iCharacter <> -1
8. iIndex = find_match(iStringCode, iCharacter)
9. If (iaCode_Value(iIndex) <> -1) Then
10.   iStringCode = iaCode_Value(iIndex)
11.   Else If (iNextCode <= MAX_CODE) Then
12.   iaCode_Value(iIndex) = iNextCode
13.   iNextCode += 1
14.   iaPrefix_Code(iIndex) = iStringCode
15.   baAppend_Character(iIndex) = Cbyte(iCharacter)
16.   End If
17.   output_code(iStringCode)
18.   iStringCode = iCharacter
19.   End If
20.   iCharacter = ReadByte()
21.   End While
22.   output_code(iStringCode)
23.   output_code(MAX_VALUE)
24.   output_code(0)
```

a. *Flow Graph* Kompresi LZW

Berikut ini adalah *flow graph* kompresi LZW yang terlihat pada Gambar 4.1.



Gambar 4.1 *Flow Graph* Kompresi LZW

Keterangan :

◯ = Menggambarkan Kondisi

→ = Menggambarkan Aksi

Dari Gambar 4.1 dapat dihitung *cyclomatic complexity* sebagai berikut:

Region = 5

V(G) = 27 *edge* – 24 *node* + 2 = 5

Jadi, *cyclomatic complexity* untuk Gambar 4.1 adalah 5.

Berdasarkan *cyclomatic complexity* tersebut, Gambar 4.1 memiliki 5 *path* yang terdiri dari:

Path 1: 1-2-3-4-2-3-4-5-6-7-8-9-10-19-20-7-21-22-23-24

Path 2: 1-2-3-4-2-3-4-5-6-7-8-9-11-12-13-14-15-16-17-18-19-20-7-21-22-23-24

Path 3: 1-2-3-4-2-3-4-5-6-7-21-22-23-24

Path 4: 1-2-3-4-2-3-4-5-6-7-8-9-10-19-20-7-8-9-11-12-13-14-15-16-17-18-19-20-7-21-22-23-24

Path 5: 1-2-3-4-5-6-7-21-22-23-24

b. *Graph Matrix* Kompresi LZW

Berikut ini adalah *graph matrix* dari kompresi LZW yang terlihat pada Tabel 4.4.

Tabel 4.4 *Graph Matrix* Kompresi LZW

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1		1																						
2			1																					
3				1																				
4		1			1																			
5						1																		
6							1																	
7								1													1			
8									1															
9										1	1													
10																			1					
11												1												
12													1											
13														1										
14															1									
15																1								
16																	1							
17																		1						
18																			1					
19																				1				
20							1														1			
21																						1		
22																							1	
23																								1
24																								

2. Pengujian Dekompresi LZW

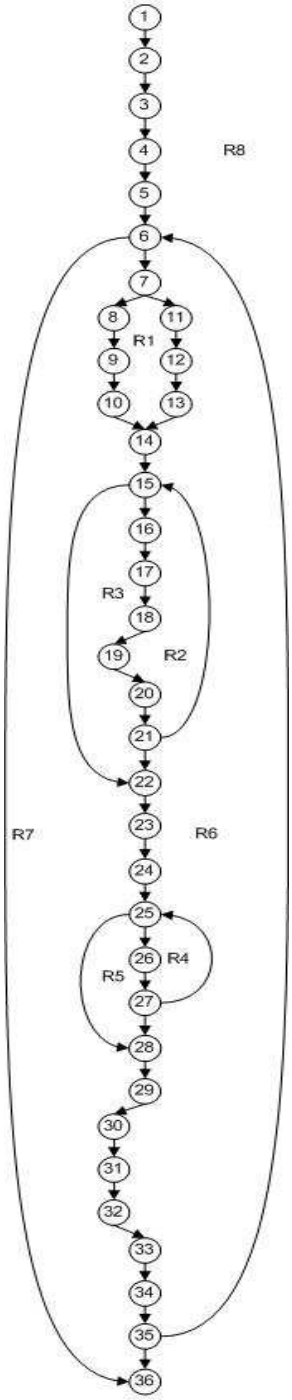
Berikut ini adalah algoritma dekompresi LZW yang akan diuji:

```
1. iNextCode = 256
2. iOldCode = input_code()
3. bCharacter = CType(iOldCode, Byte)
4. WriteByte(CByte(iOldCode))
5. iNewCode = input_code()
6. While (iNewCode <> MAX_VALUE)
7. If iNewCode >= iNextCode Then
8. baDecode_Stack(0) = bCharacter
9. i = 1
10.   iCurrCode = iOldCode
11.   Else
12.   i = 0
13.   iCurrCode = iNewCode
14.   End If
15.   While iCurrCode > 255
16.   baDecode_Stack(i) = baAppend_Character(iCurrCode)
17.   i = i + 1
18.   If i >= MAX_CODE Then
19.   Throw New ApplicationException("Fatal error during iCurrCode
   expansion.")
20.   End If
21.   iCurrCode = iaPrefix_Code(iCurrCode)
22.   End While
23.   baDecode_Stack(i) = CType(iCurrCode, Byte)
24.   bCharacter = baDecode_Stack(i)
25.   While i >= 0
```

```
26.   WriteByte(baDecode_Stack(i))
27.   i = i – 1
28.   End While
29.   If (iNextCode <= MAX_CODE) Then
30.   iaPrefix_Code(iNextCode) = iOldCode
31.   baAppend_Character(iNextCode) = bCharacter
32.   iNextCode += 1
33.   End If
34.   iOldCode = iNewCode
35.   iNewCode = input_code()
36.   End While
```

a. *Flow Graph* Dekompresi LZW

Berikut ini adalah *flow graph* dekompresi LZW yang terlihat pada Gambar 4.2.



Gambar 4.2 *Flow Graph* Dekompresi LZW

Keterangan :

◯ = Menggambarkan Kondisi

→ = Menggambarkan Aksi

Dari Gambar 4.2 dapat dihitung *cyclomatic complexity* sebagai berikut:

Region = 8

V(G) = 42 *edge* – 36 *node* + 2 = 8

Jadi, *cyclomatic complexity* untuk Gambar 4.2 adalah 8.

Berdasarkan *cyclomatic complexity* tersebut, Gambar 4.2 memiliki 8 *path* yang terdiri dari:

Path 1: 1-2-3-4-5-6-7-8-9-10-14-15-16-17-18-19-20-21-15-22-23-24-25-26-27-25-28-29-30-31-32-33-34-35-6-36

Path 2: 1-2-3-4-5-6-7-11-12-13-14-15-16-17-18-19-20-21-15-22-23-24-25-26-27-25-28-29-30-31-32-33-34-35-6-36

Path 3: 1-2-3-4-5-6-36

Path 4: 1-2-3-4-5-6-7-8-9-10-14-15-22-23-24-25-26-27-25-28-29-30-31-32-33-34-35-6-36

Path 5: 1-2-3-4-5-6-7-11-12-13-14-15-22-23-24-25-26-27-25-28-29-30-31-32-33-34-35-6-36

Path 6: 1-2-3-4-5-6-7-8-9-10-14-15-22-23-24-25-28-29-30-31-32-33-34-35-6-36

Path 7: 1-2-3-4-5-6-7-11-12-13-14-15-22-23-24-25-28-29-30-31-32-33-34-35-6-36

Path 8: 1-2-3-4-5-6-7-8-9-10-14-15-22-23-24-25-28-29-30-31-32-33-34-35-36

b. *Graph Matrix* Dekompresi LZW

Berikut ini adalah *graph matrix* dekompresi LZW yang terlihat pada Tabel 4.5.

Tabel 4.5 *Graph Matrix* Dekompresi LZW

	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	30	31	32	33	34	35	36
1		1																																		
2			1																																	
3				1																																
4					1																															
5						1																														
6							1																													1
7								1			1																									
8									1																											
9										1																										
10														1																						
11												1																								
12													1																							
13														1																						
14															1																					
15																1						1														
16																	1																			
17																		1																		
18																			1																	
19																				1																
20																					1															
21															1							1														
22																							1													
23																								1												
24																									1											
25																										1		1								
26																											1									
27																									1			1								
28																													1							
29																														1						
30																															1					
31																																1				
32																																	1			
33																																		1		
34																																			1	
35						1																														1
36																																				

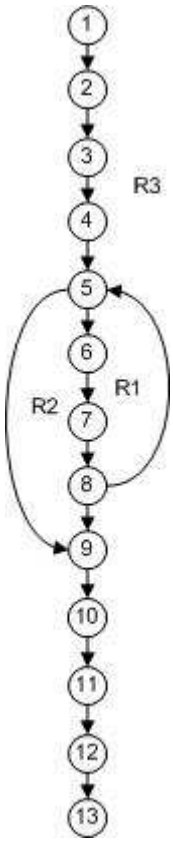
3. Pengujian Enkripsi Rijndael

Berikut ini adalah algoritma enkripsi rijndael yang akan diuji:

```
1. fsInput = New System.IO.FileStream(strInputFile, FileMode.Open,
   FileAccess.Read)
2. fsOutput = New System.IO.FileStream(strOutputFile,
   FileMode.OpenOrCreate, FileAccess.Write)
3. fsOutput.SetLength(0)
4. csCryptoStream = New CryptoStream(fsOutput,
   cspRijndael.CreateEncryptor(bytKey, bytIV),
   CryptoStreamMode.Write)
5. While lngBytesProcessed < lngFileLength
6. intBytesInCurrentBlock = fsInput.Read(bytBuffer, 0, 4096)
7. csCryptoStream.Write(bytBuffer, 0, intBytesInCurrentBlock)
8. lngBytesProcessed = lngBytesProcessed +
   CLng(intBytesInCurrentBlock)
9. End While
10.   csCryptoStream.Close()
11.   fsInput.Close()
12.   fsOutput.Close()
13.   fileOriginal.Delete()
```

a. *Flow Graph* Enkripsi Rijndael

Berikut ini adalah *flow graph* enkripsi rijndael yang terlihat pada Gambar 4.3.



Gambar 4.3 *Flow Graph* Enkripsi Rijndael

Keterangan :

◯ = Menggambarkan Kondisi

→ = Menggambarkan Aksi

Dari Gambar 4.3 dapat dihitung *cyclomatic complexity* sebagai berikut:

Region = 3

V(G) = 14 *edge* – 13 *node* + 2 = 3

Jadi, *cyclomatic complexity* untuk Gambar 4.3 adalah 3.

Berdasarkan *cyclomatic complexity* tersebut, Gambar 4.3 memiliki 3 *path* yang terdiri dari:

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

Path 2: 1-2-3-4-5-9-10-11-12-13

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

b. *Graph Matrix* Enkripsi Rijndael

Berikut ini adalah *graph matrix* enkripsi rijndael yang terlihat pada Tabel 4.6.

Tabel 4.6 *Graph Matrix* Enkripsi Rijndael

	1	2	3	4	5	6	7	8	9	10	11	12	13
1		1											
2			1										
3				1									
4					1								
5						1			1				
6							1						
7								1					
8					1				1				
9										1			
10											1		
11												1	
12													1
13													

4. Pengujian Dekripsi Rijndael

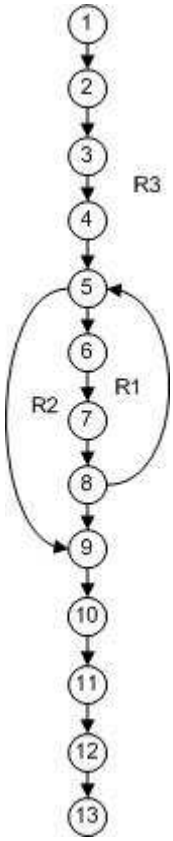
Berikut ini adalah algoritma dekripsi rijndael yang akan diuji:

```
1. fsInput = New System.IO.FileStream(strInputFile, FileMode.Open,
   FileAccess.Read)
2. fsOutput = New System.IO.FileStream(strOutputFile,
   FileMode.OpenOrCreate, FileAccess.Write)
3. fsOutput.SetLength(0)
4. csCryptoStream = New CryptoStream(fsOutput,
   cspRijndael.CreateDecryptor(bytKey, bytIV),
   CryptoStreamMode.Write)
5. While lngBytesProcessed < lngFileLength
```

```
6. intBytesInCurrentBlock = fsInput.Read(bytBuffer, 0, 4096)
7. csCryptoStream.Write(bytBuffer, 0, intBytesInCurrentBlock)
8. lngBytesProcessed = lngBytesProcessed +
   CLng(intBytesInCurrentBlock)
9. End While
10.   csCryptoStream.Close()
11.   fsInput.Close()
12.   fsOutput.Close()
13.   fileEncrypted.Delete()
```

a. *Flow Graph* Dekripsi Rijndael

Berikut ini adalah *flow graph* dekripsi rijndael yang terlihat pada Gambar 4.4.



Gambar 4.4 *Flow Graph* Dekripsi Rijndael

Keterangan :

◯ = Menggambarkan Kondisi

→ = Menggambarkan Aksi

Dari Gambar 4.4 dapat dihitung *cyclomatic complexity* sebagai berikut:

Region = 3

V(G) = 14 *edge* – 13 *node* + 2 = 3

Jadi, *cyclomatic complexity* untuk Gambar 4.4 adalah 3.

Berdasarkan *cyclomatic complexity* tersebut, Gambar 4.4 memiliki 3 *path* yang terdiri dari:

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

Path 2: 1-2-3-4-5-9-10-11-12-13

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

b. *Graph Matrix* Dekripsi Rijndael

Berikut ini adalah *graph matrix* dekripsi rijndael yang terlihat pada Tabel 4.7.

Tabel 4.7 *Graph Matrix* Dekripsi Rijndael

	1	2	3	4	5	6	7	8	9	10	11	12	13
1		1											
2			1										
3				1									
4					1								
5						1			1				
6							1						
7								1					
8					1				1				
9										1			
10											1		
11												1	
12													1
13													

4.2.2 Pengujian *Beta*

Pengujian *beta* dilakukan melalui dua tahap, yaitu wawancara pengguna dan pengujian performansi atau uji coba langsung kepada pengguna sistem untuk mencoba aplikasi yang baru dan hasil dari percobaan tersebut dicatat untuk mengetahui tingkat keakuratan dan kecepatan sistem.

4.2.2.1 Wawancara

Wawancara yang dilakukan menggunakan tiga buah pertanyaan terbuka mengenai aplikasi Doy-ENKOM kepada lima pengguna aplikasi di PT. Prospera Consulting Engineers. Berikut adalah hasil wawancara yang telah dilakukan.

1. Wawancara dengan Bapak Andik Suparto (Koordinator Keuangan dan Administrasi)
   a. Pertanyaan: “Menurut pendapat Anda, apakah fungsi enkripsi dan dekripsi dari aplikasi Doy-ENKOM sudah mampu untuk mengamankan data-data keuangan?”

      Jawaban : “Mampu, karena data-data yang sudah dikunci tidak bisa dibuka dengan paksa dan hanya bisa dibuka kembali oleh aplikasi ini.”

      Kesimpulan: fungsi enkripsi *file* dan dekripsi *file* sudah berjalan dengan baik dan sesuai dengan yang diharapkan.
   b. Pertanyaan: “Menurut pendapat Anda, apakah fungsi kompresi dan dekompresi dari aplikasi Doy-ENKOM sudah mampu untuk memampatkan data?”

      Jawaban: “Mampu dan hasilnya tidak kalah dengan aplikasi yang lain.”

Kesimpulan: fungsi kompresi dan dekompresi *file* sudah berjalan dengan baik dan sesuai dengan yang diharapkan.

c. Pertanyaan: “Menurut pendapat Anda, secara garis besar apakah aplikasi Doy-ENKOM mudah digunakan dan mudah dipelajari atau tidak?”

   Jawaban: “Mudah dan simpel.”

   Kesimpulan: aplikasi Doy-ENKOM mudah digunakan dan dipelajari.

2. Wawancara dengan Ibu Eliza Widiastuty (Koordinator RAB)

   a. Pertanyaan: “Menurut pendapat Anda, apakah fungsi enkripsi dan dekripsi dari aplikasi Doy-ENKOM sudah mampu untuk mengamankan data-data RAB?”

      Jawaban : “Mampu, karena data-data yang sudah dikunci tidak bisa dibuka kembali.”

      Kesimpulan: fungsi enkripsi *file* dan dekripsi *file* sudah berjalan dengan baik dan sesuai dengan yang diharapkan.

   b. Pertanyaan: “Menurut pendapat Anda, apakah fungsi kompresi dan dekompresi dari aplikasi Doy-ENKOM sudah mampu untuk memampatkan data?”

      Jawaban: “Mampu.”

      Kesimpulan: fungsi kompresi dan dekompresi *file* sudah berjalan dengan baik dan sesuai dengan yang diharapkan.

   c. Pertanyaan: “Menurut pendapat Anda, secara garis besar apakah aplikasi Doy-ENKOM mudah digunakan dan mudah dipelajari atau tidak?”

      Jawaban: “Mudah dan sangat sederhana.”

Kesimpulan: aplikasi Doy-ENKOM mudah digunakan dan dipelajari.

3. Wawancara dengan Bapak Kondang Wijayanto (Project Manager)
   a. Pertanyaan: “Menurut pendapat Anda, apakah fungsi enkripsi dan dekripsi dari aplikasi Doy-ENKOM sudah mampu untuk mengamankan data-data proyek?”

      Jawaban : “Mampu, karena kuncinya hanya Saya yang tahu.”

      Kesimpulan: fungsi enkripsi *file* dan dekripsi *file* sudah berjalan dengan baik dan sesuai dengan yang diharapkan.

   b. Pertanyaan: “Menurut pendapat Anda, apakah fungsi kompresi dan dekompresi dari aplikasi Doy-ENKOM sudah mampu untuk memampatkan data?”

      Jawaban: “Mampu dan hasilnya cukup memuaskan.”

      Kesimpulan: fungsi kompresi dan dekompresi *file* sudah berjalan dengan baik dan sesuai dengan yang diharapkan.

   c. Pertanyaan: “Menurut pendapat Anda, secara garis besar apakah aplikasi Doy-ENKOM mudah digunakan dan mudah dipelajari atau tidak?”

      Jawaban: “Sangat mudah.”

   Kesimpulan: aplikasi Doy-ENKOM mudah digunakan dan dipelajari.

4. Wawancara dengan Bapak Riyadi (Project Operational)
   a. Pertanyaan: “Menurut pendapat Anda, apakah fungsi enkripsi dan dekripsi dari aplikasi Doy-ENKOM sudah mampu untuk mengamankan data-data lapangan?”

Jawaban : “Mampu, karena data-data lapangan saya menjadi sangat aman.”

Kesimpulan: fungsi enkripsi *file* dan dekripsi *file* sudah berjalan dengan baik dan sesuai dengan yang diharapkan.

b. Pertanyaan: “Menurut pendapat Anda, apakah fungsi kompresi dan dekompresi dari aplikasi Doy-ENKOM sudah mampu untuk memampatkan data?”

Jawaban: “Mampu dan hasilnya juga cukup.”

Kesimpulan: fungsi kompresi dan dekompresi *file* sudah berjalan dengan baik dan sesuai dengan yang diharapkan.

c. Pertanyaan: “Menurut pendapat Anda, secara garis besar apakah aplikasi Doy-ENKOM mudah digunakan dan mudah dipelajari atau tidak?”

Jawaban: “Mudah.”

Kesimpulan: aplikasi Doy-ENKOM mudah digunakan dan dipelajari.

5. Wawancara dengan Bapak M. Wibowo T.L. (Koordinator IT)

a. Pertanyaan: “Menurut pendapat Anda, apakah fungsi enkripsi dan dekripsi dari aplikasi Doy-ENKOM sudah mampu untuk mengamankan *database*?”

Jawaban : “Mampu, karena dapat mengamankan data *backup* perusahaan.”

Kesimpulan: fungsi enkripsi *file* dan dekripsi *file* sudah berjalan dengan baik dan sesuai dengan yang diharapkan.

b. Pertanyaan: “Menurut pendapat Anda, apakah fungsi kompresi dan dekompresi dari aplikasi Doy-ENKOM sudah mampu untuk memampatkan data?”

   Jawaban: “Mampu dan hasilnya cukup.”

   Kesimpulan: fungsi kompresi dan dekompresi *file* sudah berjalan dengan baik dan sesuai dengan yang diharapkan.

c. Pertanyaan: “Menurut pendapat Anda, secara garis besar apakah aplikasi Doy-ENKOM mudah digunakan dan mudah dipelajari atau tidak?”

   Jawaban: “Mudah.”

   Kesimpulan: aplikasi Doy-ENKOM mudah digunakan dan dipelajari.

4.2.2.2 Pengujian Performansi

Pengujian performansi pada aplikasi Doy-ENKOM dibagi menjadi empat tahap, yaitu pengujian enkripsi *file*, pengujian dekripsi *file*, pengujian kompresi *file*, dan pengujian dekompresi *file*. Pada masing-masing tahap menggunakan sembilan *file* yang terdiri dari tiga *file* dokumen, dua *file* gambar, satu *file* aplikasi, dua *file* suara, dan satu *file* video. Berikut ini adalah hasil dari pengujian performansi pada aplikasi Doy-ENKOM:

1. Pengujian Performansi Enkripsi *File*

Pengujian performansi enkripsi *file* menggunakan sepuluh macam *file* yang berbeda jenisnya. Berikut ini adalah hasil pengujian performansi enkripsi *file* yang terlihat pada Tabel 4.8.

Tabel 4.8 Hasil Pengujian Performansi Enkripsi *File*

No.	Nama *File*	Rincian *File*	Ukuran *File* (KB)	Waktu Enkrip (detik)	Ukuran *File* Hasil (KB)	Rasio
1	08-AudioTrack 08.mp3	Suara	18.358	3,18	18.358	100%
2	Buku Panduan emonDAK SDA.docx	Teks, Gambar	21.512	3,24	21.512	100%
3	DAK 26 JANUARI 2012 SDA BANDUNG.pptx	Teks, Gambar	38.212	5,52	38.212	100%
4	emondak_bg3.psd	Gambar	19.574	2,76	19.574	100%
5	eMonDAK2012.mdb	Teks	21.532	3,17	21.532	100%
6	Marine02Cinematic.wav	Suara	35.131	5,14	35.131	100%
7	Slank.bmp	Gambar	26.228	3,78	26.228	100%
8	Solok-Padang traffic.flv	Video	22.893	3,24	22.893	100%
9	ymsgr1150_0192_us.exe	Aplikasi	18.920	2,69	18.920	100%

Berikut ini pada Tabel 4.9 merupakan hasil enkripsi *file* menggunakan algoritma *Blowfish.*

Tabel 4.9 Hasil Enkripsi *File* Menggunakan Algoritma *Blowfish*

No.	Nama *File*	Rincian *File*	Ukuran *File* (KB)	Waktu Enkrip (detik)	Ukuran *File* Hasil (KB)	Rasio
1	08-AudioTrack 08.mp3	Suara	18.358	1,73	18.359	100%
2	Buku Panduan emonDAK SDA.docx	Teks, Gambar	21.512	2,10	21.512	100%
3	DAK 26 JANUARI 2012 SDA BANDUNG.pptx	Teks, Gambar	38.212	3,46	38.212	100%
4	emondak_bg3.psd	Gambar	19.574	1,81	19.574	100%
5	eMonDAK2012.mdb	Teks	21.532	2,05	21.533	100%
6	Marine02Cinematic.wav	Suara	35.131	2,91	35.131	100%
7	Slank.bmp	Gambar	26.228	2,47	26.229	100%
8	Solok-Padang traffic.flv	Video	22.893	2,16	22.893	100%
9	ymsgr1150_0192_us.exe	Aplikasi	18.920	1,97	18.921	100%

Kesimpulan dari perbandingan Tabel 4.8 dan Tabel 4.9 adalah algoritma *Blowfish* memiliki waktu enkripsi yang lebih cepat tetapi ukuran *file* bertambah 1 KB untuk ekstensi *file* *.mp3, *.mdb, *.bmp, dan *.exe,

sedangkan algoritma *Rijndael* memiliki waktu enkripsi yang lebih lambat tetapi ukuran *file* tetap sama.

2. Pengujian Performansi Dekripsi *File*

Pengujian performansi dekripsi *file* menggunakan sepuluh macam *file* yang berbeda jenisnya. Berikut ini adalah hasil pengujian performansi dekripsi *file* yang terlihat pada Tabel 4.10.

Tabel 4.10 Hasil Pengujian Performansi Dekripsi *File*

No.	Nama *File*	Rincian *File*	Ukuran *File* (KB)	Waktu Dekrip (detik)	Ukuran *File* Hasil (KB)	Rasio
1	08-AudioTrack 08_mp3doykrip	Suara	18.359	2,73	18.358	100%
2	Buku Panduan emonDAK SDA_docx.doykrip	Teks, Gambar	21.512	3,10	21.512	100%
3	DAK 26 JANUARI 2012 SDA BANDUNG_pptx.doykrip	Teks, Gambar	38.212	5,67	38.212	100%
4	emondak_bg3_psd.doykrip	Gambar	19.574	2,80	19.574	100%
5	eMonDAK2012_mdb.doykrip	Teks	21.533	3,15	21.532	100%
6	Marine02Cinematic_wav.doykrip	Suara	35.131	4,99	35.131	100%
7	Slank_bmp.doykrip	Gambar	26.228	3,73	26.228	100%
8	Solok-Padang traffic_flv.doykrip	Video	22.893	3,26	22.893	100%
9	ymsgr1150_0192_us_exe.doykrip	Aplikasi	18.920	2,76	18.920	100%

Berikut ini pada Tabel 4.11 merupakan hasil dekripsi *file* menggunakan algoritma *Blowfish.*

Tabel 4.11 Hasil Dekripsi *File* Menggunakan Algoritma *Blowfish*

No.	Nama *File*	Rincian *File*	Ukuran *File* (KB)	Waktu Dekrip (detik)	Ukuran *File* Hasil (KB)	Rasio
1	08-AudioTrack 08.k2p	Suara	18.359	1	18.358	100%
2	Buku Panduan emonDAK SDA.k2p	Teks, Gambar	21.512	1,18	21.512	100%
3	DAK 26 JANUARI 2012 SDA BANDUNG.k2p	Teks, Gambar	38.212	2,02	38.212	100%
4	emondak_bg3.k2p	Gambar	19.574	1,09	19.574	100%
5	eMonDAK2012.k2p	Teks	21.533	1,29	21.532	100%
6	Marine02Cinematic.k2p	Suara	35.131	1,77	35.131	100%
7	Slank.k2p	Gambar	26.228	1,35	26.228	100%
8	Solok-Padang traffic.k2p	Video	22.893	1,22	22.893	100%
9	ymsgr1150_0192_us.k2p	Aplikasi	18.920	1,15	18.920	100%

Kesimpulan dari Tabel 4.10 dan Tabel 4.11 adalah algoritma *Rijndael* memiliki waktu dekripsi yang lebih lambat dibandingkan dengan algoritma *Blowfish*.

3. Pengujian Performansi Kompresi *File*

Pengujian performansi kompresi *file* menggunakan sepuluh macam *file* yang berbeda jenisnya. Berikut ini adalah hasil pengujian performansi kompresi *file* yang terlihat pada Tabel 4.12.

Tabel 4.12 Hasil Pengujian Performansi Kompresi *File*

No.	Nama *File*	Rincian *File*	Ukuran *File* (KB)	Waktu Kompres (detik)	Ukuran *File* Hasil (KB)	Rasio
1	08-AudioTrack 08.mp3	Suara	18.358	4,57	17.205	93,72 %
2	Buku Panduan emonDAK SDA.docx	Teks, Gambar	21.512	4,43	21.429	99,61 %
3	DAK 26 JANUARI 2012 SDA BANDUNG.pptx	Teks, Gambar	38.212	8,12	37.740	98,76 %
4	emondak_bg3.psd	Gambar	19.574	4,65	13.480	68,87 %
5	eMonDAK2012.mdb	Teks	21.532	3,21	1.757	8,16%

No.	Nama *File*	Rincian *File*	Ukuran *File* (KB)	Waktu Kompres (detik)	Ukuran *File* Hasil (KB)	Rasio
6	Marine02Cinematic.wav	Suara	35.131	9,28	28.980	82,49 %
7	Slank.bmp	Gambar	26.228	6,57	11.378	43,38 %
8	Solok-Padang traffic.flv	Video	22.893	5,45	22.736	99,31 %
9	ymsgr1150_0192_us.exe	Aplikasi	18.920	4,29	17.936	94,80 %

Berikut ini pada Tabel 4.13 merupakan hasil kompresi *file* menggunakan algoritma RAR.

Tabel 4.13 Hasil Kompresi *File* Menggunakan Algoritma RAR

No.	Nama *File*	Rincian *File*	Ukuran *File* (KB)	Waktu Kompres (detik)	Ukuran *File* Hasil (KB)	Rasio
1	08-AudioTrack 08.mp3	Suara	18.358	19	17.090	93,09 %
2	Buku Panduan emonDAK SDA.docx	Teks, Gambar	21.512	24	20.949	97,38 %
3	DAK 26 JANUARI 2012 SDA BANDUNG.pptx	Teks, Gambar	38.212	46	37.278	97,56 %
4	emondak_bg3.psd	Gambar	19.574	13	11.129	56,85 %
5	eMonDAK2012.mdb	Teks	21.532	4	1.019	4,73%
6	Marine02Cinematic.wav	Suara	35.131	14	17.602	50,10 %
7	Slank.bmp	Gambar	26.228	13	9.817	37,43 %
8	Solok-Padang traffic.flv	Video	22.893	26	22.676	99,05 %
9	ymsgr1150_0192_us.exe	Aplikasi	18.920	20	17.494	92,46 %

Kesimpulan dari Tabel 4.12 dan Tabel 4.13 adalah waktu kompresi menggunakan algoritma LZW lebih tinggi dari algoritma RAR, tetapi rasio kompresi algoritma LZW lebih besar dibandingkan algoritma RAR.

4. Pengujian Performansi Dekompresi *File*

Pengujian performansi dekompresi *file* menggunakan sepuluh macam *file* yang berbeda jenisnya. Berikut ini adalah hasil pengujian performansi dekompresi *file* yang terlihat pada Tabel 4.14.

Tabel 4.14 Hasil Pengujian Dekompresi *File*

No.	Nama *File*	Rincian *File*	Ukuran *File* (KB)	Waktu Dekompres (detik)	Ukuran *File* Hasil (KB)	Rasio
1	08-AudioTrack 08.mp3	Suara	17.205	1,92	18.358	100%
2	Buku Panduan emonDAK SDA.docx	Teks, Gambar	21.429	1,36	21.512	100%
3	DAK 26 JANUARI 2012 SDA BANDUNG.pptx	Teks, Gambar	37.740	2,51	38.212	100%
4	emondak_bg3.psd	Gambar	13.480	2,17	19.574	100%
5	eMonDAK2012.mdb	Teks	1.757	1,22	21.532	100%
6	Marine02Cinematic.wav	Suara	28.980	4,55	35.131	100%
7	Slank.bmp	Gambar	11.378	2,39	26.228	100%
8	Solok-Padang traffic.flv	Video	22.736	2,37	22.893	100%
9	ymsgr1150_0192_us.exe	Aplikasi	17.936	1,44	18.920	100%

Berikut ini pada Tabel 4.15 merupakan hasil dekompresi *file* menggunakan algoritma RAR.

Tabel 4.15. Hasil Dekompresi File Menggunakan Aplikasi WinRAR

No.	Nama *File*	Rincian *File*	Ukuran *File* (KB)	Waktu Dekompres (detik)	Ukuran *File* Hasil (KB)	Rasio
1	08-AudioTrack 08.mp3	Suara	17.090	0,30	18.358	100%
2	Buku Panduan emonDAK SDA.docx	Teks, Gambar	20.949	1	21.512	100%
3	DAK 26 JANUARI 2012 SDA BANDUNG.pptx	Teks, Gambar	37.278	1	38.212	100%
4	emondak_bg3.psd	Gambar	11.129	1	19.574	100%
5	eMonDAK2012.mdb	Teks	1.019	0,90	21.532	100%
6	Marine02Cinematic.wav	Suara	28.980	2	35.131	100%
7	Slank.bmp	Gambar	11.378	1	26.228	100%
8	Solok-Padang traffic.flv	Video	22.736	0,70	22.893	100%
9	ymsgr1150_0192_us.exe	Aplikasi	17.936	0,60	18.920	100%

Kesimpulan dari Tabel 4.14 dan Tabel 4.15 adalah waktu dekompresi menggunakan algoritma LZW lebih lambat dari algoritma RAR.

5. Kesimpulan Pengujian *Beta*

Berdasarkan hasil pengujian performansi di atas, maka diambil kesimpulan bahwa:

1. Enkripsi dan dekripsi menggunakan algoritma *Rijndael* untuk ukuran rentang *file* antara 15 MB-40 MB memiliki waktu proses antara 2 detik-5 detik.
2. Kompresi dan dekompresi menggunakan algoritma LZW untuk ukuran rentang *file* antara 15 MB-40 MB memiliki waktu proses antara 3 detik-9 detik kompres dan 1 detik-4 detik dekompres serta memiliki rasio kompresi antara 8%-99%.