BAB V
IMPLEMENTASI

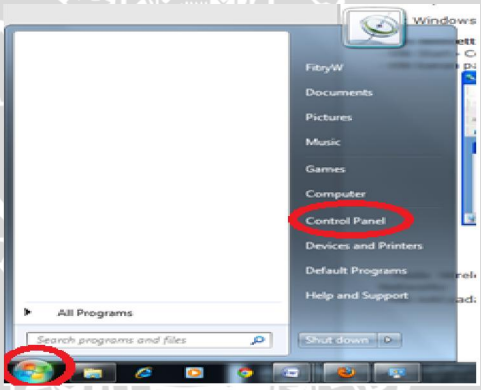
5.1 Konfigurasi Jaringan *Ad-Hoc*

Hal pertama yang perlu dilakukan pada mode *ad-hoc* adalah membuat sebuah nama jaringan atau SSID yang nantinya digunakan untuk menghubungkan PC yang satu dengan lainnya. Dibutuhkan minimal 2 PC, PC pertama digunakan untuk memegang nama SSID, sedangkan PC lainnya untuk melihat atau mendengar *broadcast* dari PC pertama agar dapat saling terkoneksi.

5.1.1 Pemegang SSID

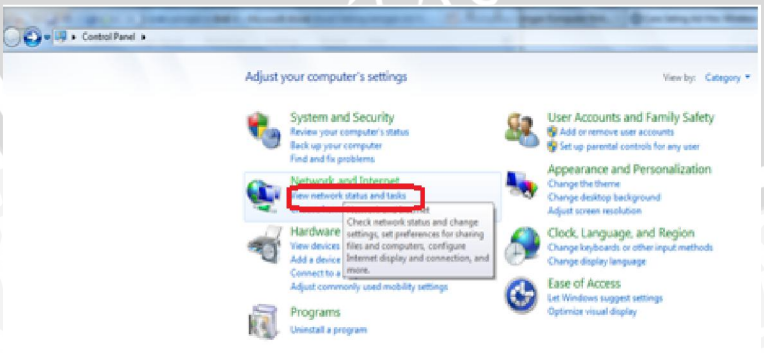
Adapun langkah-langkahnya sebagai berikut :

- Klik *Start* > *Control Panel*



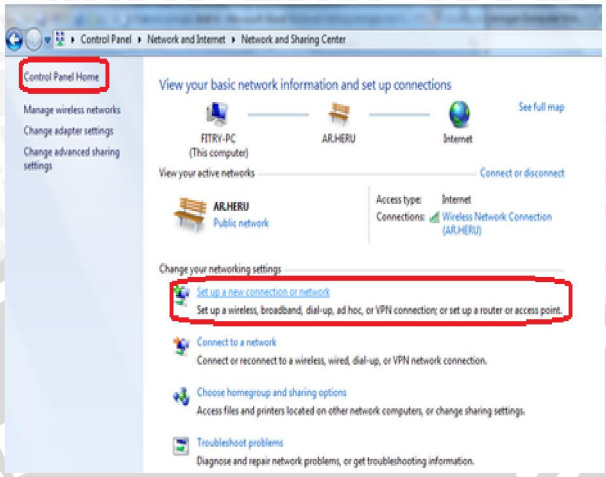
Gambar 5.1 Menu *Start*

- Pada *Network and Internet*, pilih *View network status and task*



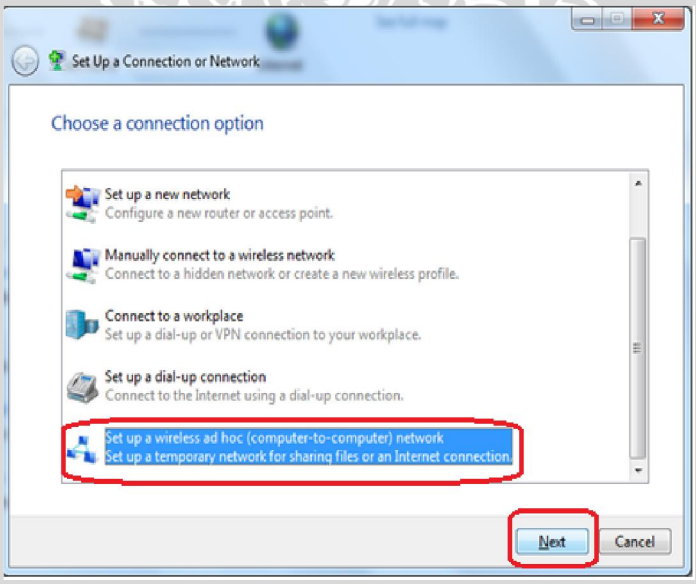
Gambar 5.2 Tampilan *Control Panel Home*

- Pilih *Set up a new connection or network.*



Gambar 5.3 Tampilan *Network and Sharing Center*

- Muncul kotak dialog memilih jenis koneksi, pilih *Set up wireless ad hoc (computer-to-computer) network,* kemudian *next.*

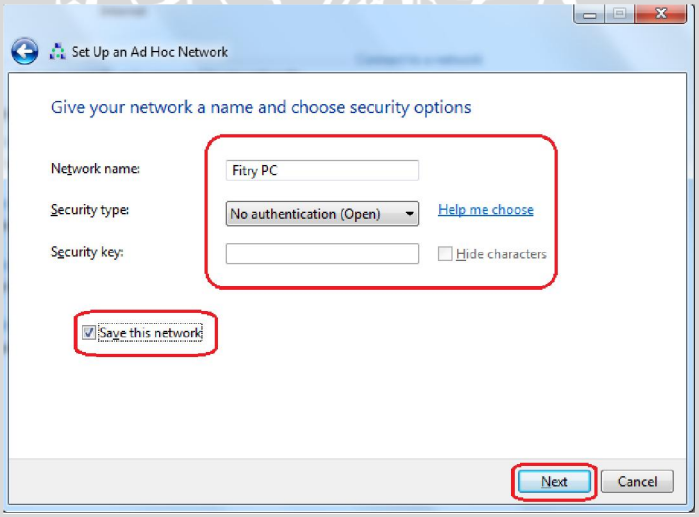


Gambar 5.4 *Set Up a Connection or Network*

- Klik *next* untuk membuat *ad hoc.*

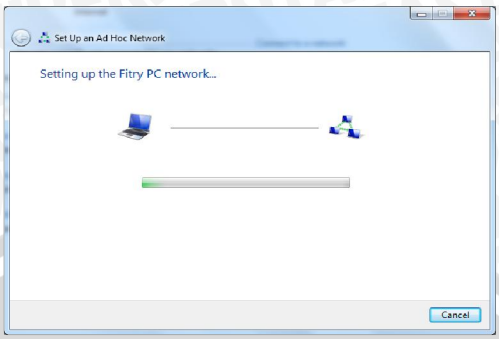
Gambar 5.5 *Set Up a Wireless Ad Hoc Network*

- Muncul *form* seperti dibawah ini, isikan nama pada *network name* misalkan Fitry PC, *security type* pilih *No authentication (open)* hal ini dikarenakan program aplikasi ini telah memiliki password untuk enkripsi data. Centang pada pilihan *Save this Network,* klik *Next.*



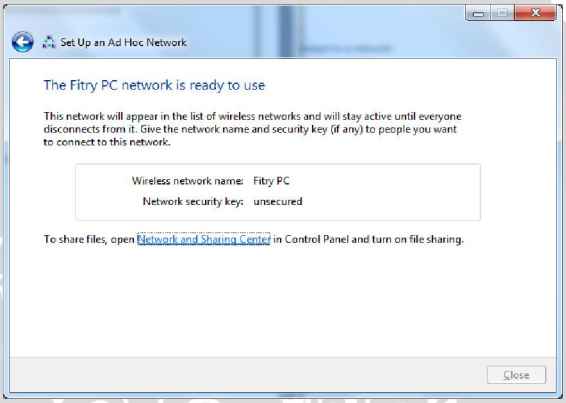
Gambar 5.6 Men*setting* SSID

- PC akan melakukan konfigurasi terhadap jaringan



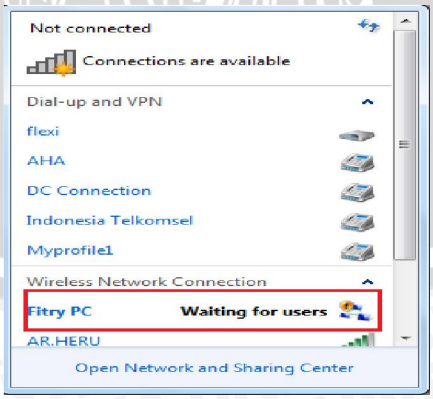
Gambar 5.7 Konfigurasi Jaringan

- Jaringan *Ad Hoc* dengan SSID Fitry PC telah siap digunakan



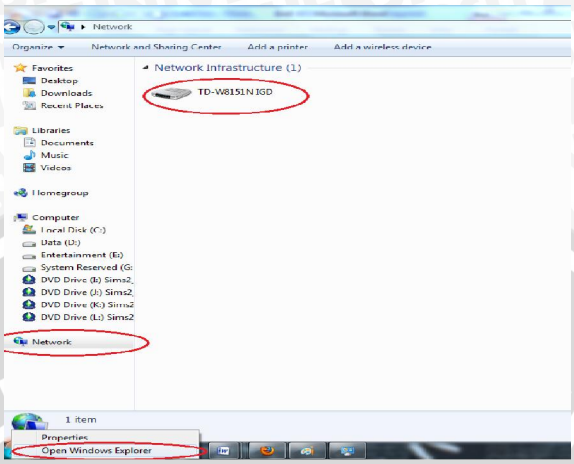
Gambar 5.8 SSID Fitry PC Digunakan

- *Check Connection* pada jaringan apakah sudah bisa termasuk ke jaringan ad hoc sekitar, dengan cara *Start* > Pilih *Connect to a Network*



Gambar 5.9 Cek Koneksi Fitry PC

- Mengecek siapa saja yang terhubung dengan jaringan kita adalah dengan klik kanan pada *Start > Open Window Explorer > Network*



Gambar 5.10 Jaringan Terhubung Fitry PC

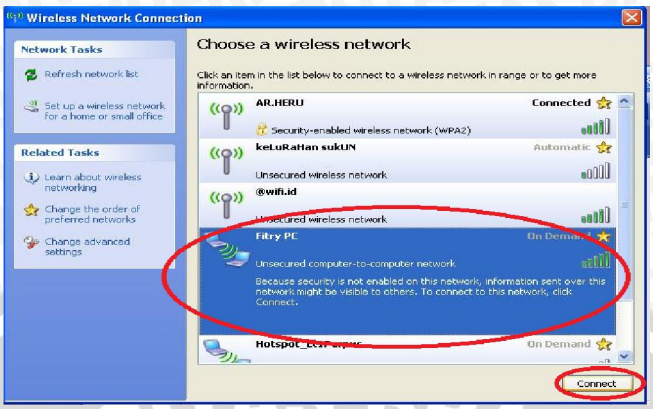
5.1.2 Menerima *Broadcast*

Langkah-langkahnya sebagai berikut :

- Untuk bergabung dalam jaringan, maka PC 2 harus melakukan koneksi ke PC 1 dengan menyebutkan SSID yang sesuai yaitu Fitry PC. Langkahnya adalah Klik *Start > Connect to > Wireless Network Connection.*

Gambar 5.11 Koneksi Terhadap Jaringan A*d hoc*

- Muncul kotak dialog, pilih Firy PC > *Connect.*

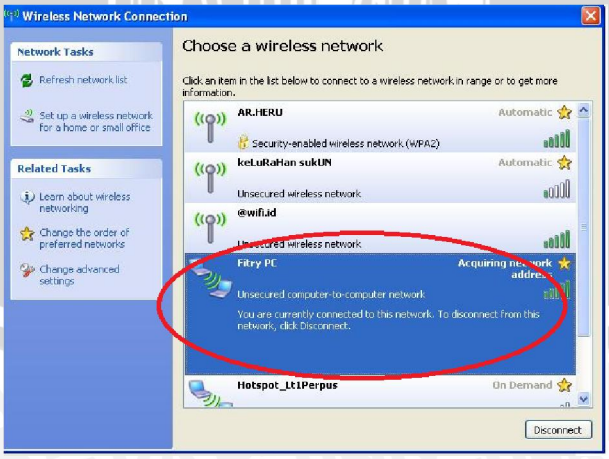


Gambar 5.12 Memilih Koneksi

- PC sedang memproses koneksi dengan Fitry PC.

Gambar 5.13 Proses Koneksi Terhadap Fitry PC

- PC telah terhubung dengan Fitry PC.



Gambar 5.14 PC Telah Terhubung Fitry PC

5.2 Implementasi Rancangan Algoritma

Implementasi algoritma yang dibahas pada implementasi skripsi ini yaitu implementasi algoritma enkripsi dan dekripsi data AES 128 bit dan RC4 128 bit serta *pseudocode* untuk program penerima serta pengirim.

5.2.1 Implementasi Enkripsi AES 128 Bit dan RC4 128 Bit

Proses enkripsi dilakukan dengan memasukkan file *plaintext* dan *password*. Jika proses enkripsi berhasil dilakukan file akan langsung terenkripsi, sedangkan jika gagal akan ada pemberitahuan kesalahan.

5.2.1.1 Implementasi Enkripsi AES 128 Bit

AES 128 ini merupakan algoritma *block cipher* dengan menggunakan sistem permutasi dan substitusi (P-Box dan S-Box). Kemudian akan dilakukan 4 transformasi sebagai berikut sebanyak 9 kali yaitu SubBytes, ShiftRows, MixColumns dan AddRoundKey. Adapun *sourcecode*-nya dijabarkan dalam gambar 5.15.

```
1.  public void Cipher(byte[] input, byte[] output)
2.          {
3.              // blok dalam state selalu [4,4]
4.              this.state = new byte[4, besar_block];
5.
6.      1       // state = input, copy nilai dari input ->
7.  state
8.              for (int i = 0; i < (4 * besar_block); ++i)
9.              {
10.         2       this.state[i % 4, i / 4] = input[i];
11.             }
12.
13.     /************************************************
14.     * Pemrosesan enkripsi mengubah state yang berasal *
15.     * dari input menjadi output                       *
16.     ************************************************/
17.
18.             // putaran awal
19.             AddRoundKey(0);     3
20.
```

```
21.             // putaran utama sebanyak jumlah putaran - 1
22.             // 128 bit: 10 round, putaran utama: 9  round
23.             for (int round = 1; round <= (jumlah_round -
24. 1); round++)
25.             {
26.                 SubBytes();
27.                 ShiftRows();
28.                 MixColumns();
29.                 AddRoundKey(round);
30.             }
31.
32.             // putaran akhir
33.             SubBytes();
34.             ShiftRows();
35.             AddRoundKey(jumlah_round);
36.
37.             // masukkan hasil pengkodean input yang
38. tersimpan dalam state ke output
39.             for (int i = 0; i < (4 * besar_block); ++i)
40.             {
41.                 output[i] = this.state[i % 4, i / 4];
42.             }
43.         }
```

4 5 6 5 9x 7 8

Gambar 5.15 Tampilan *Sourcecode Method Cipher* AES 128 Bit

(Sumber: Implementasi)

Penjelasan *sourcecode* proses *login* pada gambar 5.16 yaitu:

1. Baris 1 adalah *method cipher* (enkripsi per blok)
2. Baris 4 merupakan blok dalam *state*.
3. Baris 8-11 mengkopi nilai dari input -> *state*.
4. Baris 19 merupakan putaran awal proses enkripsi.
5. Baris 23-29 merupakan putaran utama, untuk AES 128 bit maka putaran sebanyak 9 kali putaran.
6. Baris 33-35 merupakan putaran terakhir.
7. Baris 39-41 memasukkan hasil pengkodean input yang tersimpan dalam state ke output.

5.2.1.2 Implementasi Enkripsi RC4 128 bit

RC4 merupakan salah satu jenis *stream cipher*, yaitu memproses unit atau input data pada satu saat. Unit atau data pada umumnya sebuah byte atau bahkan kadang kadang bit (*byte* dalam hal RC4). RC4 128 bit digunakan untuk menginisialisasi table sepanjang 128 bit. Tabel tersebut digunakan untuk menghasilkan *pseudo random bit.* Kemudian *pseudo random bit* yang memproses XOR dengan *plaintext* untuk menghasilkan *ciphertext.* Masing-masing elemen dalam table saling ditukarkan minimal sekali. RC4 mempunyai sebuah S-Box, $S_0,S_1,\ldots,S_{255}$, yang berisi permutasi dari bilangan 0 sampai 255, dan permutasi merupakan fungsi dari kunci dengan panjang yang variabel. Terdapat dua indeks yaitu *i* dan *j*, yang diinisialisasi dengan bilangan nol. Indeks *i* digunakan untuk memastikan bahwa suatu elemen berubah, sedangkan indeks *j* akan memastikan bahwa suatu elemen dapat berubah secara random.

Adapun *sourcecode*-nya dijabarkan dalam gambar 5.16.

```
1.  public static void Enkripsi(ref Byte[] bytes, Byte[] key)
2.          {
3.              // inisialisasi internal state 128 bit
4.              Byte[] s = new Byte[128];
5.      1       Byte[] k = new Byte[128];
6.              Byte temp;
7.              int i, j;
8.
9.              // input, menyimpan key dalam key byte array
10.             for (i = 0; i < 128; i++)        a
11.     2       {
12.                 s[i] = (Byte)i;          b
13.                 k[i] = key[i % key.GetLength(0)]; //memilih
14. kunci yang akan digunakan
15.             }
16.
17.     /************************************************
18.      * Pemrosesan enkripsi mengubah state yang berasal *
19.      * dari input menjadi output                       *
20.     ************************************************/
21.
22.             // pengacakan pada table state
```

```
23.                 j = 0;
24.                 for (i = 0; i < 128; i++)
25.                 {
26.                     j = (j + s[i] + k[i]) % 128;
27.                     temp = s[i];
28.                     s[i] = s[j];
29.                     s[j] = temp;
30.                 }
31.                 i = j = 0;
32.                 for (int x = 0; x < bytes.GetLength(0); x++)
33. //memanggil kunci yang akan digunakan
34.                 {
35.
36.                 // Generate pseudorandom key
37.                     i = (i + 1) % 128;
38.                     j = (j + s[i]) % 128;
39.                     temp = s[i];
40.                     s[i] = s[j];
41.                     s[j] = temp;
42.
43.                 // masukkan pengkodean input yang tersimpan
44. dalam state ke output
45.                     int t = (s[i] + s[j]) % 128;
46.                     bytes[x] ^= s[t];
                    }
                }
```

3 4 5 6 7

Gambar 5.16 Tampilan *Sourcecode Method Cipher* RC4 128 Bit

(Sumber: Implementasi)

Penjelasan *sourcecode* proses *login* pada gambar 5.17 yaitu:

1. Baris 1 adalah *method cipher* (enkripsi per byte)
2. Baris 4-7 inisialisasi internal *state* 128 bit.
3. Baris 10-14 mengkopi nilai dari input -> menyimpan *key* dalam *key byte array.*
4. Baris 22-32 pengacakan pada table *state*.
5. Baris 35-39 *generate pseudorandom key*

6. Baris 43-46 memasukkan hasil pengkodean *input* yang tersimpan dalam *state* ke *output*.

5.2.2 Implementasi Dekripsi AES 128 Bit dan RC4 128 Bit

Proses dekripsi dilakukan dengan memasukkan file *ciphertext* dan *password*. Jika proses dekripsi berhasil dilakukan file akan langsung terdekripsi berubah menjadi *plainteks*, sedangkan jika gagal akan ada pemberitahuan kesalahan.

5.2.2.1 Implementasi Dekripsi AES 128 Bit

```
1.  public void Decipher(byte[] input, byte[] output)
2.          {
3.
4.              // blok dalam state selalu [4,4]
5.              this.state = new byte[4, besar_block];
6.
7.              // state = input, copy nilai dari input ->
8.  state
9.              for (int i = 0; i < (4 * besar_block); ++i)
10.             {
11.                 this.state[i % 4, i / 4] = input[i];
12.             }
13.
14.     /*************************************************
15.      * Pemrosesan dekripsi mengubah state yang berasal *
16.      * dari input menjadi output                       *
17.      * ----------------------------------------------- *
18.      * Proses yang dilakukan mempunyai urutan yang     *
19.      * merupakan kebalikan dari fungsi Cipher dan      *
20.      * menggunakan fungsi invers dari tiap fungsi yang *
21.      * dipanggil pada fungsi Cipher                    *
22.     *************************************************/
23.
24.             // putaran awal
25.             AddRoundKey(jumlah_round);
26.
```

1 2 3

```
27.             // putaran utama sebanyak jumlah putaran - 1
28.             // 128 bit: 10 round, putaran utama: 9  round
29.              for (int round = jumlah_round - 1; round >=
30. 1; --round)                                  4          5
31.             {
32.                 InvShiftRows();
33.       6         InvSubBytes();
34.                 AddRoundKey(round);                      9x
35.                 InvMixColumns();
36.             }
37.
38.             // putaran akhir
39.       5     InvShiftRows();
40.             InvSubBytes();
41.             AddRoundKey(0);          7
42.
43.             // masukkan hasil pengkodean input yang
44. tersimpan dalam state ke output
45.             for (int i = 0; i < (4 * besar_block); ++i)
46.       8     {
47.                 output[i] = this.state[i % 4, i / 4];
48.             }
49.         }
```

Gambar 5.17 Tampilan *Sourcecode Method Decipher* AES 128 bit

(Sumber: Implementasi)

Penjelasan implementasi algoritma pada proses dekripsi gambar 5.18 yaitu:

1. Baris 1 adalah *method Decipher* (dekripsi per blok).
2. Baris 5 merupakan blok dalam state.
3. Baris 9-12 mengkopi nilai dari input -> state.
4. Baris 25 merupakan putaran awal proses enkripsi.
5. Baris 31-38 merupakan putaran utama, untuk AES 128 bit maka putaran sebanyak 9 kali putaran.
6. Baris 41-43 merupakan putaran terakhir.
7. Baris 47-51 masukkan hasil pengkodean input yang tersimpan dalam state ke output.

5.2.2.2 Implementasi Dekripsi RC4 128 bit

Dekripsi pada algoritma RC4 sama dengan enkripsinya, sehingga hanya ada satu fungsi yang dijalankan. Fungsi tersebut terdiri dari inisialisasi S-Box, menyimpan *key* dalam *key bit array,* permutasi untuk S-Box, *Generate Pseudorandom byte stream.*

5.2.3 Implementasi Algoritma Untuk Penerima

5.2.3.1 Koneksi Penerima

```
1.      public class Penerima : System.MarshalByRefObject ,
2.  InterfacePenerima
3.      {
4.          public static System.Windows.Forms.TextBox Logger
5.  = null;
6.          public static System.Windows.Forms.ListView
7.  daftarFileDiterima = null;
8.          public void AddLog(string text)
9.          {
10.             if (Logger.InvokeRequired)
11.             {
12.                 Action<string> invoker = new
13. Action<string>(AddLog);
14.                 Logger.Invoke(invoker, text);
15.             }
16.             else
17.             {
18.                 Logger.Text +=Environment.NewLine + text;
19.             }
20.         }
21.         public void Connect(string user)
22.         {
23.             if (Logger != null)
24.             {
25.                 if (user == null || user == "")
26.                     user = "127.0.0.1";
27.                 if (Logger.InvokeRequired)
28.                 {
```

```
29.                     Action<string> invoker = new
30. Action<string>(Connect);
31.                     Logger.Invoke(invoker,user);
32.                 }
33.                 else
34.                 {
35.                     Logger.Text += string.Format("{0}>{1}
36. terkoneksi pada [{2}].", Environment.NewLine, user,
37. DateTime.Now.ToShortTimeString());
38.                 }
39.             }
40.         }
41.         public void Disconnect(string user)
42.         {
43.             if (Logger != null)
44.             {
45.                 if (user == null || user == "")
46.                     user = "127.0.0.1";
47.
48.                 if (Logger.InvokeRequired)
49.                 {
50.                     Action<string> invoker = new
51. Action<string>(Disconnect);
52.                     Logger.Invoke(invoker,user);
53.                 }
54.                 else
55.                 {
56.                     Logger.Text += string.Format("{0}>{1}
57. koneksi terputus pada [{2}].", Environment.NewLine, user,
58. DateTime.Now.ToShortTimeString());
59.                 }
60.             }
61.         }
```

Gambar 5.18 Tampilan *Sourcecode* Koneksi Penerima

(Sumber: Implementasi)

5.2.3.2 Penerima *File*

```
1.          public void PostData(string user, byte[] data)
2.          {
3.              if (user != null && user != "127.0.0.1")
4.              {
5.                  if (PostedData != null)
6.                      PostedData(user, data);
7.              }
8.          }
9.          public event PostedDataHandler PostedData;
10.         public void Upload(string user,List<UploadData>
11. files, string password, bool isAES)
12.         {
13.             if (!System.IO.Directory.Exists("Share"))
14. System.IO.Directory.CreateDirectory("Share");
15.             foreach (UploadData file in files)
16.             {
17.                 //Variabel untuk dekripsi
18.                 String fileAsli = "";
19.                 String fileBackup = "";
20.                 System.IO.File.WriteAllBytes("Share\\" +
21. file.Filename, file.File);
22.                 //Dekripsi file sisi penerima
23.                 fileAsli = "Share\\" + file.Filename;
24.                 fileBackup = file + ".bak";
25.                 Enkripsi.dekripsi(fileAsli, fileBackup,
26. password, isAES);
27.                 File.Copy(fileBackup, fileAsli, true);
28.                 File.Delete(fileBackup);
29.                 //--------
30.                 AddLog(string.Format("> File: {0} telah
31. diupload pada {1}. oleh
32. {2}",file.Filename,DateTime.Now.ToShortTimeString(),user))
33. ;
34.                 refreshList();
35.             }
36.             if (user != null && user!="127.0.0.1")
37.             {
```

```
38.                 if (Update != null)
39.                     Update(user);
40.             }
41.         }
42.         public void refreshList()
43.         {
44.             if (daftarFileDiterima.InvokeRequired)
45.             {
46.                 MethodInvoker invoker = new
47. MethodInvoker(refreshList);
48.                 daftarFileDiterima.Invoke(invoker);
49.             }
50.             else
51.             {
52.                 try
53.                 {
54.                     daftarFileDiterima.Items.Clear();
55.                     daftarFileDiterima.SuspendLayout();
56.                     List<FileInfo> files = new
57. List<FileInfo>();
58.                     GetFiles(out files);
59.                     daftarFileDiterima.SuspendLayout();
60.                     foreach (FileInfo file in files)
61.                     {
62.                         ListViewItem item = new
63. ListViewItem((daftarFileDiterima.Items.Count +
64. 1).ToString());
65.                         item.SubItems.Add("PC Penerima");
66. item.SubItems.Add(file.Filename.Split('\\')[1]);
67. item.SubItems.Add(file.Size.ToString());
68. daftarFileDiterima.Items.Add(item);
69.                     }
70.                     daftarFileDiterima.ResumeLayout();
71.                 }
72.                 catch (Exception ex)
73.                 {
74.                     MessageBox.Show(ex.Message, "File
75. Transfer", MessageBoxButtons.OK, MessageBoxIcon.Error);
76.                 }
```

```
77.             }
78.         }
79.         public void Download(string user,string filename,
80. out byte[] file)
81.         {
82.             file = new byte[1];
83.             if (!System.IO.Directory.Exists("Share"))
84. System.IO.Directory.CreateDirectory("Share");
85.             foreach (string the in
86. System.IO.Directory.GetFiles("Share"))
87.             {
88.                 if(the.Contains(filename))
89.                 if (System.IO.File.Exists(the))
90.                 {
91.                     file =
92. System.IO.File.ReadAllBytes(the);
93.                     AddLog(string.Format("> File: {0}
94. telah didownload pada {1}. oleh {2}",(new
95. System.IO.FileInfo(the)).Name,DateTime.Now.ToShortTimeStri
96. ng(),user));
97.                     break;
98.                 }
99.             }
100.            if (file.Length == 1)
101.                file = null;
102.        }
103.        public void GetFiles(out List<FileInfo> files)
104.        {
105.            if (!System.IO.Directory.Exists("Share"))
106.System.IO.Directory.CreateDirectory("Share");
107.            List<FileInfo > list = new List<FileInfo>();
108.            foreach (string file in
109.System.IO.Directory.GetFiles("Share"))
110.            {
111.                list.Add(new FileInfo(file, ((new
112.System.IO.FileInfo(file).Length) / 1024)));
113.            }
114.            files = list;
115.        }
```

```
116.        public event UpdateHandler Update;
117.    }
118.}
```

Gambar 5.19 Tampilan *Sourcecode* Penerima *File*

(Sumber: Implementasi)

5.2.4 Implementasi Algoritma Untuk Pengirim

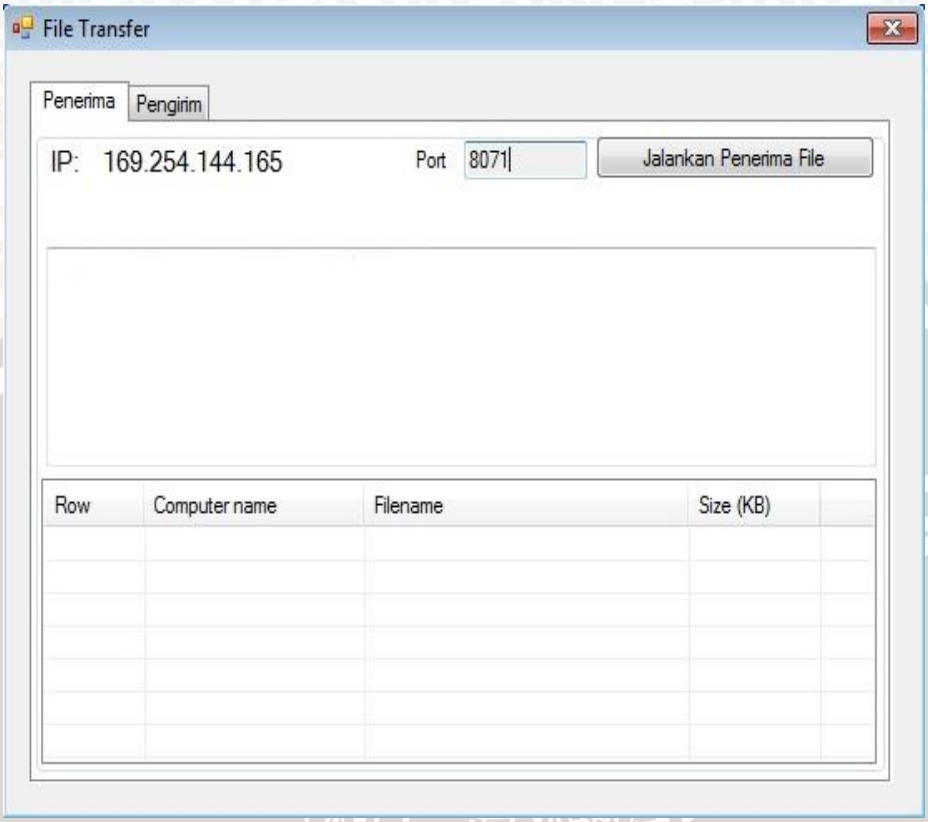
```
1.     public class DataKirim : PostedData
2.     {
3.         public event EventHandler RefreshList;
4.         public event EventHandler RefreshListPenerima;
5.         public  override  void  ImplementedPostData(string
6. user, byte[] data)
7.         {
8.             if (!user.Equals(MachineInfo.GetJustIP()))
9.                 return;
10.         }
11.         public   override   void   ImplementedUpdate(string
12. user)
13.         {
14.             if (user.Equals(MachineInfo.GetJustIP()))
15.                 return;
16.             if (RefreshList != null)
17.                 RefreshList(this, null);
18.             if (RefreshListPenerima != null)
19.                 RefreshListPenerima(this, null);
20.         }
21.     }
22. }
```

Gambar 5.20 Tampilan *Sourcecode* Pengirim

(Sumber: Implementasi)

5.3 Tampilan Aplikasi

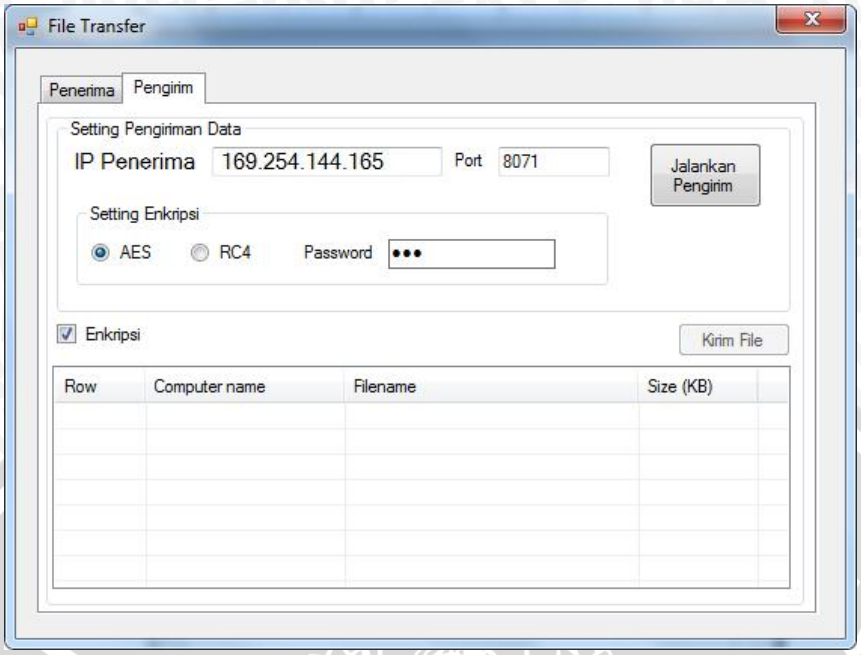
Tampilan aplikasi pada program *transfer file* dapat dilihat pada gambar 5.21 dan gambar 5.22 berikut ini.



Gambar 5.21 Tampilan Aplikasi Penerima
(Sumber : Implementasi)

Keterangan gambar:

1. IP : IP yang digunakan oleh penerima.
2. *Port* : *Port* yang digunakan untuk aplikasi ini.
3. Jalankan Penerima: Untuk menggunakan aplikasi penerima pada transfer file.



Gambar 5.22 Tampilan Aplikasi Pengirim
(Sumber : Implementasi)

Keterangan gambar:

1. IP Penerima : IP PC penerima yang dipakai.
2. *Port* : *Port* yang telah ditentukan oleh penerima.
3. *Password* : Kunci yang ditetapkan oleh pengirim.
4. Jalankan Pengirim : Untuk menggunakan aplikasi pengirim pada transfer *file.*
5. Enkripsi : Tanpa proses enkripsi, hilangkan *checklist*
6. Kirim File : Untuk mengirimkan *file* ke PC penerima.

Program aplikasi ini mempunyai 2 buah menu utama yaitu menu untuk penerima dan pengirim. Apabila pada menu pengirim IP dan *port* diisikan sesuai dengan penerima, maka program tersebut dapat dijalankan pada PC. Untuk mengisikan *port*, sebaiknya memperhatikan *port* yang telah digunakan. Hal ini untuk menghindari adanya tabrakan program apabila keduanya dijalankan dalam satu PC. Sedangkan untuk IP penerima, jika tidak terkoneksi pada jaringan manapun, maka IP 127.0.0.1 (*default*). *Password* secara *default* akan berisi 123 dan port *default* akan berisi 8071.