BAB IV
IMPLEMENTASI

Bab ini menjelaskan implementasi dari perancangan pada bab 3. Implementasi ini meliputi lingkungan implementasi, implementasi perangkat keras dan implementasi perangkat lunak.

4.1. Lingkungan Implementasi

Lingkungan implementasi yang akan dijelaskan dalam subbab ini adalah lingkungan implementasi perangkat keras dan perangkat lunak yang digunakan.

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan sistem adalah sebagai berikut :

- CPU : Intel *Core* i3 522, 3.2 Ghz
- RAM : 4 Gb DDR3
- *Harddisk* : 640 Gb
- Lain : Keyboard, mouse, display.

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan sistem ini adalah :

- Sistem Operasi : Microsoft Window 7 Sp.1
- *Framework* : .Net Framework 4.0
- Script : C#
- *Compiler* : Microsoft Visual Studio 2010
- *Image Editor* : Adobe Photoshop CS3

4.2. Implementasi Proses

Berdasarkan analisis dan perancangan proses yang terdapat dalam bab 3, maka pada subbab ini akan dijelaskan implementasi proses-proses tersebut.

4.2.1. Implementasi Proses Parameter Input

Pada implementasi bertujuan untuk melakukan validasi terhadap input dari user baik inputan citra maupun kunci. Implementasi proses parameter input diperlihatkan pada Gambar 4.1 sampai dengan Gambar 4.3

```
OpenFileDialog OFDialog = newOpenFileDialog();
            OFDialog.Title = "Select Cover Image File";
            OFDialog.Filter = "Bitmap File (*.Bmp) | *.bmp";
StringBuilder sb = newStringBuilder();
if (OFDialog.ShowDialog() == DialogResult.OK)
            {

                Text = OFDialog.FileName;
Bitmap bmp = newBitmap(OFDialog.FileName);
                pbOri.Image = bmp;
                sb.AppendLine("Informasi :");
                sb.AppendLine(" + File : " + OFDialog.FileName);
                sb.AppendLine();
                sb.AppendLine(" + Width : " + bmp.Width.ToString() + "
pixel");
                sb.AppendLine(" + Height : " + bmp.Height.ToString() + "
pixel");
                btnProcess.Enabled = true;
                lbLog.Items.Add(DateTime.Now.ToString() + " : Load image
file "+ OFDialog.FileName);
                FileName = OFDialog.FileName ;

long[] histData = GetHistogram(bmp);
Series series = this.chart1.Series[0] ;
                series.Points.Clear();
for (int i = 0; i < histData.Length; i++)
                {
                    series.Points.Add(histData[i]);
                }
                lbLog.Items.Add(DateTime.Now.ToString() + " : Generating
histogram  data");
            }
            expandablePanel2.Expanded = true;
            lbInfo1.Text = sb.ToString();
```

Gambar 4.1 Proses Pemilihan Citra

Pada proses Gambar 4.1, perangkat lunak melakukan proses pengambilan citra digital dari berkas, dan kemudian menampilkan informasi lokasi serta dimensi dari citra yang dipilih, yang kemudian ditampilkan.

```
long[] myHistogram = newlong[256];

for (int i = 0; i < picture.Size.Width; i++)
for (int j = 0; j < picture.Size.Height; j++)
```

```
                {
                    System.Drawing.Color c = picture.GetPixel(i, j);

long Temp = 0;
                    Temp += c.R;
                    Temp += c.G;
                    Temp += c.B;

                    Temp = (int)Temp / 3;
                    myHistogram[Temp]++;
                }

return myHistogram;
```

Gambar 4.2 Proses Pembuatan Historgram

Pada proses Gambar 4.2, histogram diambil dari data piksel dari citra, yang kemudian diproses untuk menghasilkan visualisasi histogram.

```
int keyFix = (txtKey.Value > Mw ? txtKey.Value % Mw : txtKey.Value);
            keyFix = (keyFix > Nh ? keyFix % Nh : keyFix);
//init key
for (int i = 0; i < Mw; i++)
                Kc[i] = Color.FromArgb(keyFix  );
for (int j = 0; j < Nh; j++)
                Kr[j] = Color.FromArgb(keyFix  );
```

Gambar 4.3 Proses Pembentukan Kunci

Pada proses Gambar 4.3, kunci dihasilkan dari input user. Kemudian kunci inputan diproses untuk menghasilkan matrik K_C dan K_R.

4.2.2 Implementasi Proses Enkripsi

Proses enkripsi berfungsi untuk melakukan pengacakan informasi dari citra yang telah diinputkan kedalam bentuk citra terenkripsi. Implementasi proses enkripsi diperlihatkan pada Gambar 4.4 sampai dengan Gambar 4.8

```
if (txtKey.Value <= 10) return;
Bitmap picOri = newBitmap(FileName);
            Io = newColor[picOri.Size.Width, picOri.Size.Height] ;
            Kc = newColor[picOri.Size.Width ];
            Kr = newColor[picOri.Size.Height];
            Mw = picOri.Size.Width;
            Nh = picOri.Size.Height;
```

Gambar 4.4 Proses Inisialisasi Parameter

```
//pembacaan pixel
          lbLog.Items.Add(DateTime.Now.ToString() + " : Reading images
pixel ");
for (int i = 0; i < Mw; i++)
for (int j = 0; j < Nh; j++)
                {
                    System.Drawing.Color c = picOri.GetPixel(i, j);
                    Io[i, j] = c;
                }
```

Gambar 4.5 Proses Pembacaan Piksel

```
while (ITER < ITERmax)
                {
if (DEBUG) Console.WriteLine("ITERSI :" + ITER);

                    progressBarX1.Value = ITER;
Int64[] Ai = newInt64[Nh];
//Row
if (DEBUG) Console.WriteLine("Proses tiap-tiap baris pixel :");
for (int i = 0; i < Nh; i++)
                    {
                        Ai[i] = 0;
if (DEBUG) Console.Write("Ai["+i+"]  = "   );


for (int j = 0; j < Mw; j++)
                        {
                            Ai[i] += Io[j, i].ToArgb();
if (DEBUG) Console.Write(Io[j, i].ToArgb()+"+\t");
                        }

Int64 Mai = Ai[i] % 2;
if (DEBUG) Console.WriteLine(" ="+Ai[i] + "+\t");
if (DEBUG) Console.WriteLine("Mai =" + Mai + "+\t");

Color[] tmp = newColor[Mw];

if (DEBUG) Console.WriteLine("Nilai Sebelum = \t");
for (int k = 0; k < Mw; k++)
                        {
                            tmp[k] = Io[k, i];
if (DEBUG) Console.Write(String.Format("0x{0:X}\t", Io[k, i].ToArgb()));
                        }
if (DEBUG) Console.WriteLine( );
if (Mai == 0)
                        {

                            DownOrRightCirculantShift(tmp, Kr[i].ToArgb());
if (DEBUG) Console.WriteLine("Mai = 0 ,Sehingga Geser Kanan " +
Kr[i].ToArgb() + "\t");

                        }
else
                        {
                            UpOrLeftCirculantShift(tmp, Kr[i].ToArgb());
```

```
if (DEBUG) Console.WriteLine("Mai <> 0 ,Sehingga Geser Kiri " +
Kr[i].ToArgb() + "\t");
                    }
for (int k = 0; k < Mw; k++)
                    {
                        Io[k, i] = tmp[k];
if (DEBUG) Console.Write(String.Format("0x{0:X}\t", Io[k, i].ToArgb()));
                    }
if (DEBUG) Console.WriteLine();
                }

//Column
if (DEBUG) Console.WriteLine("Proses tiap-tiap  kolom :");
Int64[] Bi = newInt64[Mw];
for (int i = 0; i < Mw; i++)
                {
                    Bi[i] = 0;
if (DEBUG) Console.Write("Bi[" + i + "]  = ");
for (int j = 0; j < Nh; j++)
                    {
                        Bi[i] += Io[i, j].ToArgb();
if (DEBUG) Console.Write(Io[j, i].ToArgb() + "+\t");
                    }
Int64 Mbi = Bi[i] % 2;
if (DEBUG) Console.WriteLine(" =" + Bi[i] + "+\t");
if (DEBUG) Console.WriteLine("Mbi =" + Mbi + "+\t");

Color[] tmp = newColor[Nh];
if (DEBUG) Console.WriteLine("Nilai Sebelum = \t");
for (int k = 0; k < Nh; k++)
                    {
                        tmp[k] = Io[i, k];
if (DEBUG) Console.Write(String.Format("0x{0:X}\t", Io[k, i].ToArgb()));
                    }

if (Mbi == 0)
                    {
                        DownOrRightCirculantShift(tmp, Kc[i].ToArgb());
if (DEBUG) Console.WriteLine("Mbi = 0 ,Sehingga Geser Turun " +
Kr[i].ToArgb() + "\t");

                    }
else
                    {
                        UpOrLeftCirculantShift(tmp, Kc[i].ToArgb());
if (DEBUG) Console.WriteLine("Mbi <> 0 ,Sehingga Geser Atas " +
Kr[i].ToArgb() + "\t");

                    }
for (int k = 0; k < Nh; k++)
                    {
                        Io[i, k] = tmp[k];
if (DEBUG) Console.Write(String.Format("0x{0:X}\t", Io[k, i].ToArgb()));
                    }
if (DEBUG) Console.WriteLine();
                }

if (DEBUG) Console.WriteLine("Nilai Pixel Sebelum :");
```

```
Color[,] I1 = newColor[picOri.Size.Width, picOri.Size.Height];
for (int i = 0; i < Mw; i++)
                    {
for (int j = 0; j < Nh; j++)
                        {
                            I1[i, j] = Io[i, j];
if (DEBUG) Console.Write(String.Format("0x{0:X}\t", I1[i, j].ToArgb()));
                        }
if (DEBUG) Console.WriteLine();
                    }

//Row
if (DEBUG) Console.WriteLine("Proses tiap-tiap baris pixel :");
for (int i = 0; i < Nh; i++)
                    {

for (int j = 0; j < Mw; j++)
                        {

if (2 * j - 1 < Mw && 2 * j - 1 >= 0)
                            {
if (DEBUG) Console.WriteLine("I1[2 * " + j + " - 1, " + i + "] = " + Io[2 *
j - 1, i].ToArgb() + " xor " + Kc[j].ToArgb() + "\t");
                                I1[2 * j - 1, i] = Color.FromArgb(Io[2 * j
- 1, i].ToArgb() ^ Kc[j].ToArgb());


                            }
if (2 * j < Mw)
                            {
if (DEBUG) Console.WriteLine("I1[2 * " + j + ", " + i + "] = " + Io[2 * j,
i].ToArgb() + " xor " + Kc[Kc.Length - j - 1].ToArgb() + "\t");
                                I1[2 * j, i] = Color.FromArgb(Io[2 * j,
i].ToArgb() ^ Kc[Kc.Length - j - 1].ToArgb());
                            }
                        }
                    }
if (DEBUG) Console.WriteLine("Proses tiap-tiap kolom pixel :");
//column
for (int i = 0; i < Mw; i++)
                    {

for (int j = 0; j < Nh; j++)
                        {

if (2 * j - 1 < Nh && 2 * j - 1 >= 0)
                            {
if (DEBUG) Console.WriteLine("I1[" + i + ", 2 * " + j + " - 1] = " + Io[i,
2 * j - 1 ].ToArgb() + " xor " + Kr[j].ToArgb() + "\t");
                                I1[i, 2 * j - 1] = Color.FromArgb(Io[i, 2 *
j - 1].ToArgb() ^ Kr[j].ToArgb());
                            }
if (2 * j < Nh - 1)
                            {
if (DEBUG) Console.WriteLine("I1[" + i + ", 2 * " + j + "] = " + Io[i, 2 *
j].ToArgb() + " xor " + Kr[Kr.Length - j - 1].ToArgb() + "\t");
                                I1[i, 2 * j] = Color.FromArgb(Io[i, 2 *
j].ToArgb() ^ Kr[Kr.Length - j - 1].ToArgb());
```

```
                        }
                    }
                }

if (DEBUG) Console.WriteLine("Nilai Pixel Sebelum :");
for (int i = 0; i < Mw; i++)
                {
for (int j = 0; j < Nh; j++)
                    {
                        Io[i, j] = I1[i, j];
if (DEBUG) Console.Write(String.Format("0x{0:X}\t", Io[i, j].ToArgb()));
                    }
if (DEBUG) Console.WriteLine();
                }
                ITER++;
            }
```

Gambar 4.6 Proses Enkripsi

```
      Color[] temp = newColor[places];
Array.Copy(array, 0, temp, 0, places);
Array.Copy(array, places, array, 0, array.Length - places);
Array.Copy(temp, 0, array, array.Length - places, places);
return array;
```

Gambar 4.7 Proses Rotasi Piksel Kanan atau Bawah

Pada proses Gambar 4.7, digunakan untuk melakukan rotasi baris piksel dari baris kiri kearah kanan dan juga dari posisi atas menuju bawah.

```
      Color[] temp = newColor[array.Length - places];

Array.Copy(array, 0, temp, 0, array.Length - places);
Array.Copy(array, array.Length - places, array, 0, places);
Array.Copy(temp, 0, array, places, array.Length - places);
return array;
```

Gambar 4.8 Proses Rotasi Piksel Kiri atau Atas

Pada proses Gambar 4.8, digunakan untuk melakukan rotasi baris piksel dari baris kanan kearah kiri dan juga dari posisi bawah menuju atas.

4.2.3 Implementasi Proses Dekripsi

Proses dekripsi berfungsi untuk melakukan proses pengembalian informasi dari citra yang telah diinputkan kedalam bentuk citra terdekripsi. Implementasi proses dekripsi diperlihatkan pada Gambar 4.9 sampai dengan Gambar 4.13

```
if (txtKey.Value <= 10) return;
```

```
Bitmap picOri = newBitmap(FileName);
          Io = newColor[picOri.Size.Width, picOri.Size.Height] ;
          Kc = newColor[picOri.Size.Width ];
          Kr = newColor[picOri.Size.Height];
          Mw = picOri.Size.Width;
          Nh = picOri.Size.Height;
```

Gambar 4.9 Proses Inisialisasi Parameter

```
      //pembacaan pixel
                lbLog.Items.Add(DateTime.Now.ToString() + " : Reading
      images pixel ");
      for (int i = 0; i < Mw; i++)
      for (int j = 0; j < Nh; j++)
                      {
                          System.Drawing.Color c = picOri.GetPixel(i, j);
                          Io[i, j] = c;
                      }
```

Gambar 4.10 Proses Pembacaan Piksel

```
      while (ITER < ITERmax)
                {
                    progressBarX1.Value = ITER;
if (DEBUG) Console.WriteLine("Nilai pixel sebelum :");
Color[,] I1 = newColor[picOri.Size.Width, picOri.Size.Height];
for (int i = 0; i < Mw; i++)
                    {
for (int j = 0; j < Nh; j++)
                        {
                            I1[i, j] = Io[i, j];
if (DEBUG) Console.Write(String.Format("0x{0:X}\t", I1[i, j].ToArgb()));
                        }
if (DEBUG) Console.WriteLine();
                    }

if (DEBUG) Console.WriteLine("Proses tiap-tiap :");
////column
for (int i = 0; i < Mw; i++)
                    {

for (int j = 0; j < Nh; j++)
                        {


if (2 * j - 1 < Nh && 2 * j - 1 >= 0)
                            {
if (DEBUG) Console.WriteLine("I1[" + i + ", 2 * " + j + " - 1] = " + Io[i,
2 * j - 1].ToArgb() + " ^ " + Kr[j].ToArgb() + "\t");
                                I1[i, 2 * j - 1] = Color.FromArgb(Io[i, 2 *
j - 1].ToArgb() ^ Kr[j].ToArgb());
                            }
if (2 * j < Nh - 1)
                            {
if (DEBUG) Console.WriteLine("I1[" + i + ", 2 * " + j + "] = " + Io[i, 2 *
j].ToArgb() + " ^ " + Kr[Kr.Length - j - 1].ToArgb() + "\t");
```

```
                                I1[i, 2 * j] = Color.FromArgb(Io[i, 2 *
j].ToArgb() ^ Kr[Kr.Length - j - 1].ToArgb());
                            }
                        }
                    }
if (DEBUG) Console.WriteLine("Proses tiap-tiap baris pixel :");
//Row
for (int i = 0; i < Nh; i++)
                    {

for (int j = 0; j < Mw; j++)
                        {


if (2 * j - 1 < Mw && 2 * j - 1 >= 0)
                            {
if (DEBUG) Console.WriteLine("I1[2 * " + j + " - 1, " + i + "] = " + Io[2 *
j - 1, i].ToArgb() + " ^ " + Kc[j].ToArgb() + "\t");
                                I1[2 * j - 1, i] = Color.FromArgb(Io[2 * j
- 1, i].ToArgb() ^ Kc[j].ToArgb());

                            }
if (2 * j < Mw)
                            {
if (DEBUG) Console.WriteLine("I1[2 * " + j + ", " + i + "] = " + Io[2 * j,
i].ToArgb() + " ^ " + Kc[Kc.Length - j - 1].ToArgb() + "\t");
                                I1[2 * j, i] = Color.FromArgb(Io[2 * j,
i].ToArgb() ^ Kc[Kc.Length - j - 1].ToArgb());
                            }

                        }
                    }


if (DEBUG) Console.WriteLine("Nilai Pixel Sebelum :");
for (int i = 0; i < Mw; i++)
                    {
for (int j = 0; j < Nh; j++)
                        {
                            Io[i, j] = I1[i, j];
if (DEBUG) Console.Write(String.Format("0x{0:X}\t", Io[i, j].ToArgb()));
                        }
if (DEBUG) Console.WriteLine();
                    }


// Column
if (DEBUG) Console.WriteLine("Proses tiap-tiap  kolom :");
Int64[] Bi = newInt64[Mw];
for (int i = 0; i < Mw; i++)
                    {
                        Bi[i] = 0;
if (DEBUG) Console.Write("Bi[" + i + "]  = ");
for (int j = 0; j < Nh; j++)
                        {
                            Bi[i] += Io[i, j].ToArgb();
if (DEBUG) Console.Write(Io[j, i].ToArgb() + "+\t");
                        }
```

```
Int64 Mbi = Bi[i] % 2;
if (DEBUG) Console.WriteLine(" =" + Bi[i] + "+\t");
if (DEBUG) Console.WriteLine("Mbi =" + Mbi + "+\t");

Color[] tmp = newColor[Nh];
if (DEBUG) Console.WriteLine("Nilai Sebelum = \t");
for (int k = 0; k < Nh; k++)
                        {
                            tmp[k] = Io[i, k];
if (DEBUG) Console.Write(String.Format("0x{0:X}\t", Io[k, i].ToArgb()));
                        }

if (Mbi == 0)
                        {

if (DEBUG) Console.WriteLine("Mbi = 0 ,Sehingga Geser Atas " +
Kr[i].ToArgb() + "\t");
                            UpOrLeftCirculantShift(tmp, Kc[i].ToArgb());
                        }
else
                        {
if (DEBUG) Console.WriteLine("Mbi <> 0 ,Sehingga Geser Turun " +
Kr[i].ToArgb() + "\t");
                            DownOrRightCirculantShift(tmp, Kc[i].ToArgb());


                        }
for (int k = 0; k < Nh; k++)
                        {

if (DEBUG) Console.Write(String.Format("0x{0:X}\t", Io[k, i].ToArgb()));
                            Io[i, k] = tmp[k];
                        }
if (DEBUG) Console.WriteLine();
                    }


Int64[] Ai = newInt64[Nh];
//Row
if (DEBUG) Console.WriteLine("Proses tiap-tiap baris pixel :");
for (int i = 0; i < Nh; i++)
                    {
                        Ai[i] = 0;
if (DEBUG) Console.Write("Ai[" + i + "]  = ");
for (int j = 0; j < Mw; j++)
                        {
                            Ai[i] += Io[j, i].ToArgb();
if (DEBUG) Console.Write(Io[j, i].ToArgb() + "+\t");
                        }
Int64 Mai = Ai[i] % 2;
if (DEBUG) Console.WriteLine(" =" + Ai[i] + "+\t");
if (DEBUG) Console.WriteLine("Mai =" + Mai + "+\t");

Color[] tmp = newColor[Mw];
if (DEBUG) Console.WriteLine("Nilai Sebelum = \t");
for (int k = 0; k < Mw; k++)
                        {
```

```
if (DEBUG) Console.Write(String.Format("0x{0:X}\t", Io[k, i].ToArgb()));
                            tmp[k] = Io[k, i];
                        }

if (Mai == 0)
                        {

if (DEBUG) Console.WriteLine("Mai = 0 ,Sehingga Geser Kiri " +
Kr[i].ToArgb() + "\t");
                            UpOrLeftCirculantShift(tmp, Kr[i].ToArgb());
                        }
else
                        {

if (DEBUG) Console.WriteLine("Mai <> 0 ,Sehingga Geser Kanan " +
Kr[i].ToArgb() + "\t");
                            DownOrRightCirculantShift(tmp, Kr[i].ToArgb());
                        }
for (int k = 0; k < Mw; k++)
                        {

if (DEBUG) Console.Write(String.Format("0x{0:X}\t", Io[k, i].ToArgb()));
                            Io[k, i] = tmp[k];
                        }
if (DEBUG) Console.WriteLine();
                    }



                    ITER++;
                }
```

Gambar 4.11 Proses Dekripsi

```
        Color[] temp = newColor[places];
Array.Copy(array, 0, temp, 0, places);
Array.Copy(array, places, array, 0, array.Length - places);
Array.Copy(temp, 0, array, array.Length - places, places);
return array;
```

Gambar 4.12 Proses Rotasi Piksel Kanan atau Bawah

Pada proses Gambar 4.12, digunakan untuk melakukan rotasi baris piksel dari baris kiri kearah kanan dan juga dari posisi atas menuju bawah.

```
        Color[] temp = newColor[array.Length - places];

Array.Copy(array, 0, temp, 0, array.Length - places);
Array.Copy(array, array.Length - places, array, 0, places);
Array.Copy(temp, 0, array, places, array.Length - places);
return array;
```

Gambar 4.13 Proses Rotasi Piksel Kiri atau Atas

Pada proses Gambar 4.13, digunakan untuk melakukan rotasi baris piksel dari baris kanan kearah kiri dan juga dari posisi bawah menuju atas.

4.2.4 Implementasi Proses Analisis

Proses analisis berfungsi untuk melakukan proses perhitungan dari analisis korelasi, *MSE* dan *PSNR*. Dimana hasil perhitungan tersebut digunakan untuk melakukan analisis terhadap kualitas dari citra terdekripsi jika dibandingkan dengan citra yang asli. Implementasi proses dekripsi diperlihatkan pada Gambar 4.14 sampai dengan Gambar 4.15

```
        double[] ori = newdouble[Mw * Nh];
double[] eks = newdouble[Mw * Nh];

int idx = 0;
for (int i = 0; i < Mw; i++)
for (int j = 0; j < Nh; j++)
                {
                    ori[idx] = img1[i, j].ToArgb();
                    idx++;
                }
            idx = 0;
for (int i = 0; i < Mw; i++)
for (int j = 0; j < Nh; j++)
                {
                    eks[idx] = img2[i, j].ToArgb();
                    idx++;
                }


double temp = 0;
for (int x = 0; x < ori.Length; x++)
            {
                temp += Math.Pow(ori[x] - eks[x], 2);
            }

            MSE = temp / ori.Length;

if (MSE != 0)
            {
                PSNR = 10 * Math.Log10(Math.Pow(255, 2) / MSE);
            }
else
            {
                PSNR = 100;
            }
```

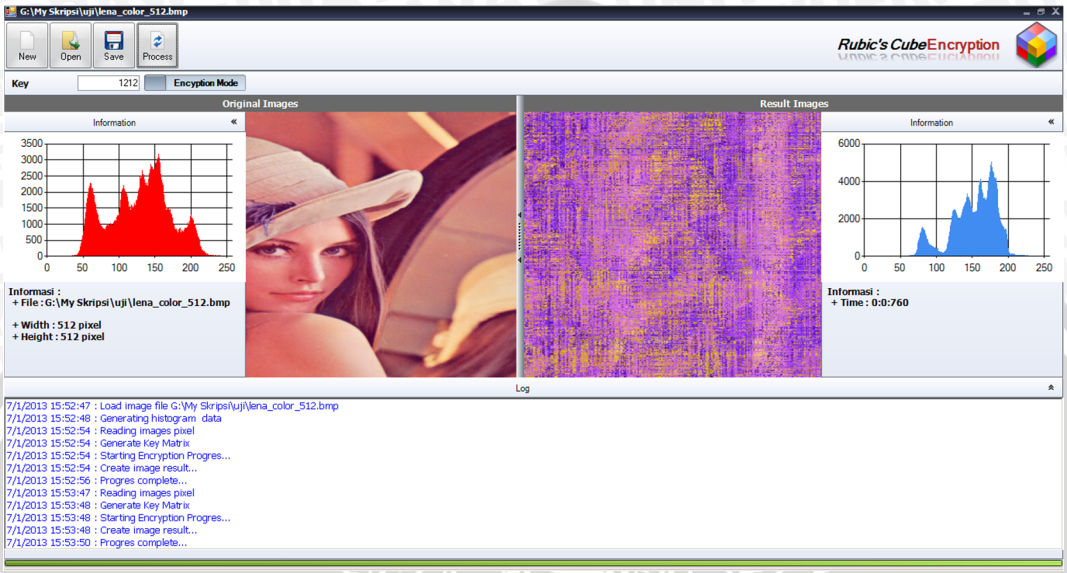
Gambar 4.14 Proses Perhitungan MSE dan PSNR

```
      double[] array1 = newdouble[Mw * Nh];
double[] array2 = newdouble[Mw * Nh];
int idx = 0;
for (int i = 0; i < Mw; i++)
for (int j = 0; j < Nh; j++){
                    array1[idx] =  img1[i, j].ToArgb();
                    idx ++;
                }
            idx = 0;
for (int i = 0; i < Mw; i++)
for (int j = 0; j < Nh; j++){
                    array2[idx] =  img2[i, j].ToArgb();
                    idx ++;
                }


double[] array_xy = newdouble[array1.Length];
double[] array_xp2 = newdouble[array1.Length];
double[] array_yp2 = newdouble[array1.Length];

for (int i = 0; i < array1.Length; i++)
                array_xy[i] = array1[i] * array2[i];

for (int i = 0; i < array1.Length; i++)
                array_xp2[i] = Math.Pow(array1[i], 2.0);

for (int i = 0; i < array1.Length; i++)
                array_yp2[i] = Math.Pow(array2[i], 2.0);
double sum_x = 0;
double sum_y = 0;

foreach (double n in array1)
                sum_x += n;
foreach (double n in array2)
                sum_y += n;

double sum_xy = 0;
foreach (double n in array_xy)
                sum_xy += n;

double sum_xpow2 = 0;
foreach (double n in array_xp2)
                sum_xpow2 += n;

double sum_ypow2 = 0;
foreach (double n in array_yp2)
                sum_ypow2 += n;

double Ex2 = Math.Pow(sum_x, 2.00);
double Ey2 = Math.Pow(sum_y, 2.00);

double Correl = (array1.Length * sum_xy - sum_x * sum_y) /
Math.Sqrt((array1.Length * sum_xpow2 - Ex2) * (array1.Length * sum_ypow2 -
Ey2));
```

Gambar 4.15 Proses Perhitungan Analisis Korelasi

4.2.5 Implementasi Antarmuka

Sesuai dari perancangan halaman antamuka pada Gambar 3.6, hasil dari implementasi halaman atarmuka diperlihatkan pada Gambar 4.16.



Gambar 4.16 Halaman Atarmuka

Pada halaman antarmuka Gambar 4.16, terdapat empat buah tombol navigasi, yakni tombol *new, open, save* dan *process*. Tombol *new* digunakan untuk memulai proses baru dan membersihkan tampilan. Tombol *open* digunakan untuk memilih citra yang akan digunakan pada berkas. Tombol *save* digunakan untuk menyimpan citra hasil proses dekripsi atau enkripsi. Tombol *process* digunakan untuk memulai proses, baik proses enkripsi ataupun dekripsi.