BAB IV
PERANCANGAN DAN IMPLEMENTASI

Bab ini menjelaskan mengenai langkah-langkah yang akan dilakukan untuk merancang dan mengimplementasikan aplikasi sistem enkripsi *file* citra 2 dimensi menggunakan algoritma kriptografi ElGamal. Perancangan dan implementasi dikerjakan dengan beberapa tahap. Meliputi pembangkitan kunci privat dan kunci publik, proses enkripsi, dan proses dekripsi.

4.1 Representasi Pixel

Untuk mengimplementasikan algoritma ElGamal, perlu diperhatikan representasi pixel yang dienkripsi maupun didekripsi. Untuk algoritma tersebut menggunakan representasi pixel yang sama. Sebuah citra berwarna merupakan objek 2D yang tersusun dari pixel-pixel. Proses enkripsi dan dekripsi dengan cara menelusuri setiap pixel secara iteratif sehingga dapat mengakses seluruh pixel. Kemudian untuk setiap pixel di dapat warna yang di representasikan sebagai RGB masing masing 1 byte. Sebagai contoh, warna putih total memiliki nilai red=255, green=255, dan blue=255.

Agar pixel yang diakses dapat di terapkan sebagai *message* maka dari pixel yang didapat dibentuk menjadi sebuah bilangan *integer*. Bilangan yang dimaksud didapat dari mengubah nilai red, green, dan blue pada sebuah pixel menjadi hexadesimal 8 bit untuk setiap nilainya. Masing masing nilai kemudian di *append* 1 sama lain dimulai dari red, green, kemudian blue. Setelah mendapatkan hexadesimal 24 bit maka segera dikonversikan menjadi bilangan integer, bilangan inilah yang nantinya menjadi *message* untuk diterapkan dalam sebuah algoritma ElGamal.

Sebagai contoh diberikan sebuah citra berwarna dengan ukuran 512 x 512 pixel. Kemudian diambil pixel ke (0,0) dan didapat pixel dengan warna yang putih (red=255,green=255,blue=255). Masing – masing di konversi menjadi hexadesimal 8 bit dan nilainya menjadi red=FF,green=FF, dan blue=FF. Ketiga hexadesimal yang didapat kemudian digabung menjadi

satu dimulai dari red, green, blue dan menjadi pixel =FFFFFF. Hexadesimal pixel lalu dikonversi menjadi bilangan *integer*. Pixel putih mempunyai *integer* = 16777215. Bilangan tersebut yang nantinya sebagai message yang akan di proses. Cara sebaliknya dapat digunakan untuk membentuk sebuah pixel berwarna dari hasil enkripsi. Terdapat beberapa properti algoritma elgamal antara lain :

1. Bilangan prima, p (tidak rahasia)
2. Bilangan acak, g (g < p, *grayscale*) (tidak rahasia)
3. Bilangan acak, r(r< p, kunci privat) (rahasia)
4. $z = g^r \bmod p$ (kunci publik,RRGGBB) (tidak rahasia) (4-1)
5. m atau f(x,y) (*plain*citra, *grayscale*) (rahasia)
6. a dan b (a dan b adalah *cipher*citra,RRGGBB) (tidak rahasia)

Pada prinsipnya proses dekripsi melakukan hal yang sama dengan ketika mengenkripsi sebuah pixel, yang membedakan adalah algoritma dekripsinya saja. Pixel yang ingin di dekripsi harus diubah dalam bentuk yang dapat di masukkan ke dalam algoritma kemudian hasilnya dibentuk ulang menjadi sebuah pixel berwarna atau *grayscale*.

4.2 Perancangan Program

Pada bagian perancangan aplikasi yang akan dibuat menggunakan bahasa pemrograman *Microsoft visual studio C# 2010* dan aplikasi yang dibuat nantinya dapat melakukan hal-hal sebagai berikut :

1. Menentukan nilai kunci yang meliputi variabel p, g, r, dan z. Dimana nilai variabel p adalah nilai bilangan prima random. Variabel r adalah nilai random *integer* dengan ketentuan 1 < r < p-2 .g dan z adalah nilai pixel suatu citra.
2. Proses enkripsi dengan menyusun nilai-nilai intensitas sesuai blok-blok pada pixel citra. Nilai-nilai ini yang disebut nilai m (*plain*citra *grayscale*). nilai m harus masih berada didalam range 0 sampai p – 1. Kemudian memilih bilangan acak *k*, yang dalam hal ini $1 \leq k \leq p - 2$. Setiap blok dienkripsi dengan rumus.

$$a = g^k \bmod p \quad (4\text{-}2)$$

$$b = z^k m \mod p \quad (4\text{-}3)$$

3. Proses dekripsi digunakan kunci privat r dan p untuk mendekripsi a dan b menjadi *plain*citra m dengan persamaan.

$$(a^r)^{-1} = a^{p-1-r} \bmod p \quad (4\text{-}4)$$

$$m = b/a^r \bmod p = b(a^r)^{-1} \bmod p \quad (4\text{-}5)$$

4.2.1 Konversi Citra ke *Grayscale*

Pengambilan *plain*citra berasal dari citra hasil *scanner* atau berupa citra *digital* dengan format citra yang dapat berupa *jpeg*, *jpg*, *png* dan *bmp*. Setelah *plain*citra di *load* kedalam program maka langkah selanjutnya adalah melakukan konversi citra *true color* ke dalam bentuk *grayscale*.

Pada pengubahan sebuah gambar menjadi *grayscale* dapat dilakukan dengan cara mengambil semua *pixel* pada gambar kemudian warna tiap *pixel* akan diambil informasi mengenai 3 warna dasar yaitu merah, biru dan hijau, ketiga warna dasar ini akan dijumlahkan sesuai algoritma citra *grayscale*. Nilai yang didapatkan dari penjumlahan tadi inilah yang akan dipakai untuk memberikan warna pada *pixel* gambar sehingga warna menjadi *grayscale*, tiga warna dasar dari sebuah *pixel* akan di*set* dengan nilai tersebut. Secara matematis, perhitungan citra *grayscale* ditunjukkan pada rumus (4-6).

$$Grayscale = 0.30R + 0.559G + 0.11B \quad (4\text{-}6)$$

Dengan R adalah merah, G adalah hijau dan B adalah biru. Programnya seperti ditunjukkan pada gambar 4.1.

```
for (int x = 0; x < sourceBmp.Width; x++)
{
for (int y = 0; y < sourceBmp.Height; y++)
      {
Color originalColor = sourceBmp.GetPixel(x, y);
int intValue = (int)((originalColor.R * .30) + (originalColor.G * .59) +
                            (originalColor.B * .11));
  grayscaleBmp.SetPixel(x, y, Color.FromArgb(intValue, intValue, intValue));
      }
}
```

Gambar 4.1 Program *Grayscale*

4.2.2 Mendapatkan Nilai Intensitas Pixel

Untuk mendapatkan nilai intensitas suatu pixel citra, dibutuhkan code unsafe agar program dapat mengakses langsung menuju memori. Mengenai kinerja dan fleksibilitas, dengan menggunakan pointer maka program dapat mengakses data dan memanipulasinya. Programnya seperti ditunjukkan pada gambar 4.2.

```
 privateBitmap sourceBmp;
   Citra cit = newCitra();
 Bitmap gambar = newBitmap(pictureBoxG.Image);
Bitmap grayscaleBmp = newBitmap(sourceBmp.Width, sourceBmp.Height);
   BitmapData bmCitra = gambar.LockBits(newRectangle(0, 0, gambar.Width, gambar.Height),
                                        ImageLockMode.ReadWrite,
                                        PixelFormat.Format24bppRgb);

        int stride = bmCitra.Stride; //Lebar satu baris data pixel suatu image
        System.IntPtr Scan0 = bmCitra.Scan0; //Data array fixed pada memory
        unsafe
        {
            byte* p = (byte*)(void*)Scan0;
            int nOffset = stride - bmCitra.Width * 3;
            for (int b = 0; b < bmCitra.Height; b++)
            {
                for (int c = 0; c < bmCitra.Width; c++)
                {
                int g = cit.GetPixel(bmCitra, b, c); //mendapatkan nilai pixel
                //manipulasi pixel citra
                p += 3;
                }
              p += nOffset;
            }
        }
        dataGridView1.DataSource = tabel;
        gambar.UnlockBits(bmCitra);
```

Gambar 4.2 Program untuk mendapatkan nilai intensitas pixel

4.2.3 Proses *Generate Key*

Pada proses *generate key* atau pembangkitan kunci dilakukan beberapa langkah antara lain :

1. Tentukan bilangan prima p secara acak dan bilangan *random* r. Berikut programnya pada gambar 4.3.

```
int[] myy = newint[] { 16777139,16777141,16777153,16777183,16777199,16777213};
Random ran = newRandom();
int mynum = myy[ran.Next(0, myy.Length)];
            labelprima.Text = Convert.ToString(mynum);
            labelprima.Visible = true;
            #region Bilangan_Random_r
Random rnd = newRandom();
int r = rnd.Next(3, Convert.ToInt32(labelprima.Text)-1);
            label6.Text = Convert.ToString(r);
            label6.Visible = true;
            #endregion
            labelprima.Visible = true;
            buttonSaveZ.Enabled = true;
              buttonSaveG.Enabled = true;
              buttonload.Enabled = true;
```

Gambar 4.3 Program bilangan random prima p dan bilangan random r

2. *Load* suatu citra dan ubah citra kedalam bentuk *grayscale* dimana citra ini adalah nilai dari variabel g. Berikut programnya pada gambar 4.4.

```
DialogResult dr = openFileDialog1.ShowDialog();
if (dr == DialogResult.OK)
            {
sourceBmp = newBitmap(openFileDialog1.FileName);
Bitmap grayscaleBmp = newBitmap(sourceBmp.Width, sourceBmp.Height);
for (int x = 0; x < sourceBmp.Width; x++)
                {
for (int y = 0; y < sourceBmp.Height; y++)
                    {
Color originalColor = sourceBmp.GetPixel(x, y);
int intValue = (int)((originalColor.R * .30) + (originalColor.G * .59) +
                                              (originalColor.B * .11));
                        grayscaleBmp.SetPixel(x, y, Color.FromArgb(intValue,
                                               intValue, intValue));
                    }
                }
                pictureBoxG.Image = grayscaleBmp;
                System.IO.FileInfo info =
newSystem.IO.FileInfo(openFileDialog1.FileName);
            }
```

Gambar 4.4 Program *load* citra g

3. Hitung nilai z dengan persamaan (4-1). Untuk menghitung nilai z, kunci publik g dipangkatkan dengan kunci privat r dan dimoduluskan bilangan prima p. Dimana g adalah suatu citra *grayscale*. Seperti program yang terdapat pada lampiran 1.

4.2.4 Proses Enkripsi

Pada proses *generate key* didapat kunci publik p, g, dan z yang digunakan untuk mengenkripsi *plain*citra. Dimana terdapat langkah-langkah proses enkripsi antara lain :

1. *Load plain*citra dan ubah citra kedalam bentuk *grayscale.* Berikut Programnya pada gambar 4.5.

```
        try
            {
Bitmap grayscaleBmp = newBitmap(sourceBmp.Width, sourceBmp.Height);
int Prim = Convert.ToInt32(textBoxVarP.Text);
for (int x = 0; x < sourceBmp.Width; x++)
                {
for (int y = 0; y < sourceBmp.Height; y++)
                    {
Color originalColor = sourceBmp.GetPixel(x, y);
int intValue = (int)((originalColor.R * .30) + (originalColor.G * .59) +
                                                (originalColor.B * .11));
                        grayscaleBmp.SetPixel(x, y, Color.FromArgb(intValue,
                                                intValue, intValue));
                    }
                }
                pictureBoxGrayscale.Image = grayscaleBmp;
            }
catch (Exception)
            {
                buttonProcced.Enabled = false;
            }
```

Gambar 4.5 Program konversi *plain*citra kedalam bentuk *grayscale*

2. Input bilangan prima yang didapat dari proses *generate key* dan dapatkan nilai *random* dengan variabel k dengan syarat 1<k<p-2. Seperti pada gambar 4.6.

```
                BigInteger Bil = Convert.ToInt32(textBoxVarP.Text);
if (Bil < 255 || Bil > 16777215)
                {
MessageBox.Show("Input Range (255 - 16777215)");
                }
else
                {
                    #region Bilangan_Random_k
Random rnd = newRandom();
BigInteger k = rnd.Next(20, 257);
                    labelK.Text = Convert.ToString(k);
                    labelK.Visible = true;
                    buttonLoadG.Enabled = true;
                    buttonProcced.Enabled = true;
                    #endregion
                }
```

Gambar 4.6 Program input *random* nilai k

3. *Load* citra g dan z yang didapat dari proses *generate key.*
4. Hitung a dan b seperti pada persamaan (4-2) dan (4-3). Implementasi program terdapat pada lampiran 2 dan 3.

4.2.5 Proses Dekripsi

Untuk mendekripsi a dan b digunakan bilangan prima p dan kunci privat r, sehingga *plain*citra m diperoleh kembali dengan

persamaan (4-4) dan (4-5). Dengan demikian *plain*citra dapat ditemukan kembali dari pasangan *cipher* a dan b. Untuk mendapatkan *plain*citra, pertama *Load cipher*citra a dan b. kemudian *input* bilangan prima p dan nilai kunci privat r. Setelah itu hitung m berdasarkan nilai-nilai intensitas pixel. Seperti pada *source code* lampiran 4.

4.3 Implementasi Sistem

Algoritma ElGamal adalah sebuah algoritma untuk kriptografi kunci publik.Pada umumnya algoritma ElGamal hanya diterapkan untuk data berupa teks saja. Dengan memanipulasi suatu nilai-nilai intesitas pixel citra. Maka algoritma ini dapat diimplementasikan ke *file* citra 2 dimensi.

4.4 Lingkungan Implementasi

Untuk mengimplementasikan aplikasi sistem enkripsi *file* citra 2 dimensi menggunakan algoritma kriptografi ElGamal, dibutuhkan suatu perangkat pendukungantara lain:

a. Perangkat Keras (laptop)

   Spesifikasi :

   Processor : Intel® Core™ 2 Duo CPU T6670 2.20GHz

   Memory RAM : 2.00 GB

   VGA : On-board

b. Perangkat Lunak :

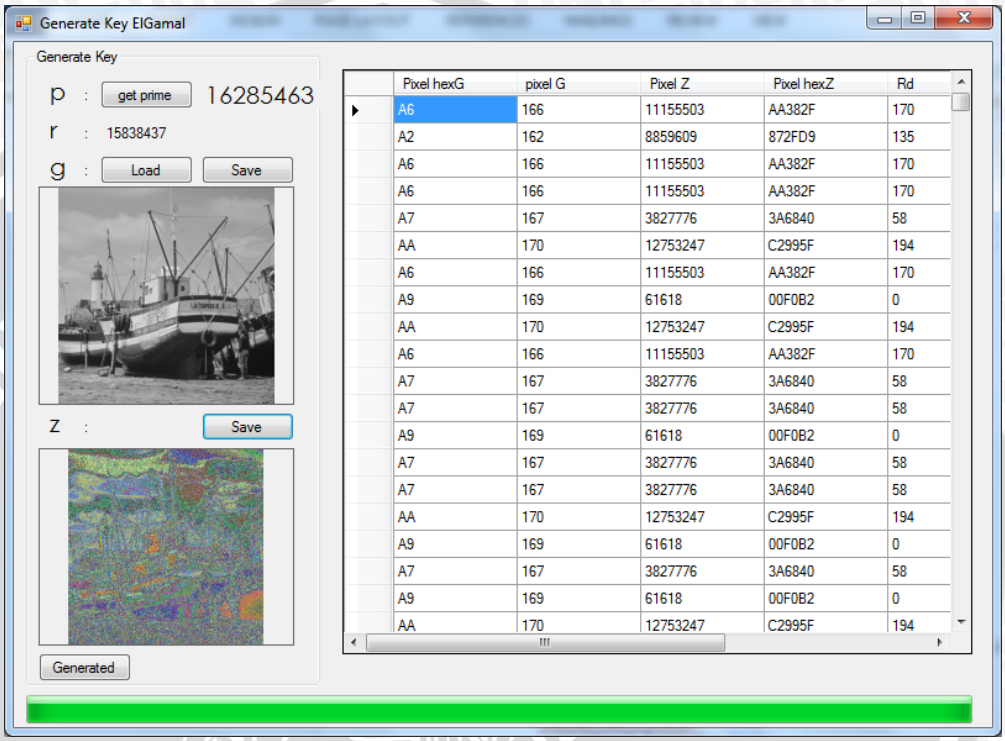
   1. *Operating systemWindows* 7.
   2. *Software Microsoft Visual C#* 2010.

4.5 Implementasi *Interface* (Antarmuka)

Program aplikasi sistem enkripsi *file* citra 2 dimensi menggunakan algoritma kriptografi ElGamal ini didisain untuk mengenkripsi dan mendekripsi citra sesuai dengan langkah-langkah yang berurutan. Hal ini bertujuan agar pemakai tidak

mengalami kesulitan dalam pengoprasian program ini. Dalam hal ini program *interface* terbagi menjadi tiga secara terpisah yaitu *generate key*, enkripsi, dan dekripsi.

4.5.1 *Interface generate key ElGamal*



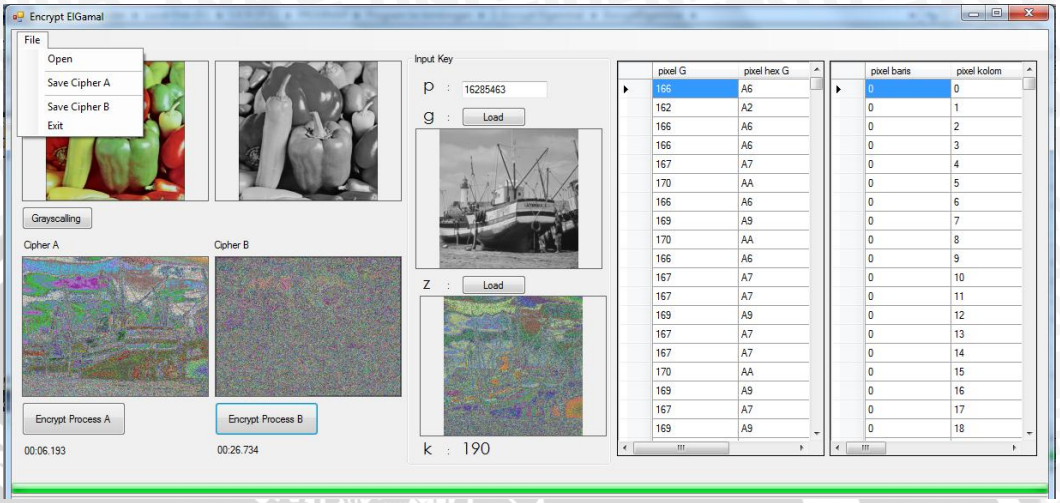
Gambar 4.7 Disain *user interfacegenerate key*

Pada disain gambar 4.7 terdapat 4 tombol yaitu :

1. Tombol *get prime* : tombol digunakan untuk mendapatkan bilangan prima secara acak dan nilai r (nilai kunci privat) secara acak.
2. Tombol *load* pada g digunakan untuk *input* citra dan konversi citra ke *grayscale* yang akan ditampilkan pada *pictureBox* g.
3. Tombol *save* pada g digunakan untuk menyimpan citra pada *pictureBox* g begitu pula dengan tombol *save* pada z digunakan untuk menyimpan citra pada *pictureBox* z.
4. Tombol *generated* yaitu tombol untuk menghitung nilai z dan mengubah nilai tersebut menjadi citra RRGGBB dan ditampilkan pada *pictureBox* z.

Kemudian terdapat *DataGridView* sebagai rincian informasi nilai intesitas citra.

4.5.2 *InterfaceEncrypt ElGamal*



Gambar 4.8 Desain *user interface* enkripsi

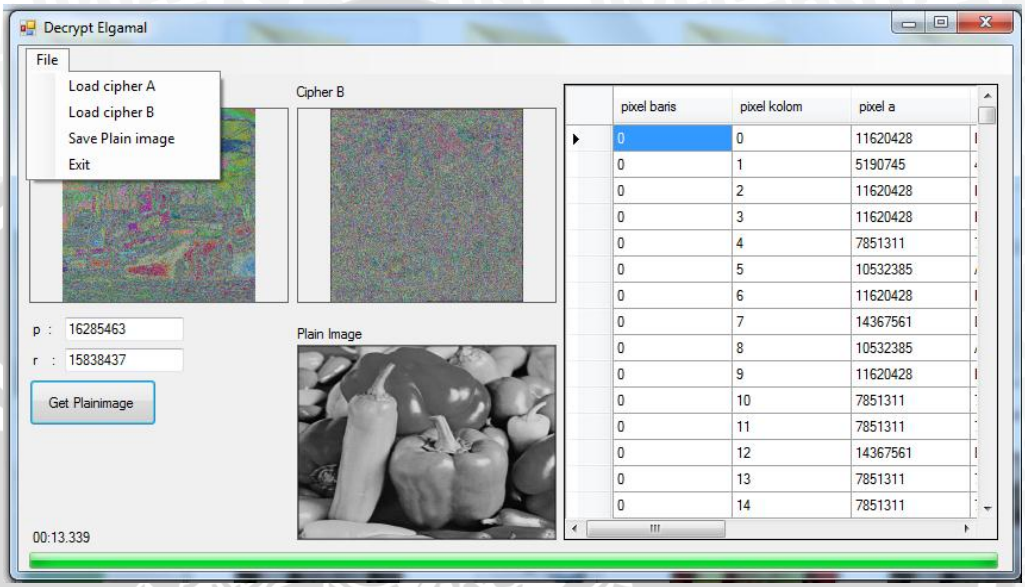
Pada disain *interface* gambar 4.8 terdapat 4 tombol dan *menu* pada *menu strip file* yaitu :

1. Tombol *Grayscaling* untuk mengubah citra true color (RGB) menjadi citra *grayscale* pada *pictureBox grayscale.*
2. Tombol *load* pada g dan z sebagai *input* citra g dan z dari *generate key,* kemudian ditampilkan pada *pictureBox* g dan z.
3. Tombol *Encrypt Process A* adalah tombol untuk menghitung nilai *cipher* a dan menjadikannya citra warna RRBBGG, kemudian ditampilkan pada *pictureBox Cipher A*.
4. Tombol *Encrypt Process B* adalah tombol untuk menghitung nilai *cipher* b dan menjadikannya citra warna RRBBGG, kemudian ditampilkan pada *pictureBox Cipher B*.
5. *Menu File Open* untuk menginput *file plain*citra yang akan di proses.
6. *Menu Save cipher A*untuk menyimpan citra pada *pictureBox cipher A.*
7. *Menu Save cipher B* untuk menyimpan citra pada *pictureBox cipher B.*

8. *Menu Exit* untuk keluar dari program.

Kemudian terdapat juga *DataGridView* sebagai rincian informasi nilai intesitas citra.

4.5.3 *Interface Decrypt ElGamal*



Gambar 4.9Disain *user interface* dekripsi

Pada *interface* gambar 4.9 terdapat beberapa *menu* pada *menu strip file*, diantaranya:

- *Save Plain image* : untuk melakukan penyimpanan hasil *plain*citra pada *pictureBox Plain Image*, berupa citra *grayscale*
- *Load cipher A* : untuk melakukan *load cipher A* ke *pictureBox Cipher B* yang didapat dari proses enkripsi.
- *Load cipher B* : untuk melakukan *load cipher B* ke *pictureBox Cipher B* yang didapat dari proses enkripsi.
- *Exit* : digunakan untuk keluar dari program.

Kemudian terdapat tombol *Get Plain image* untuk menghitung nilai m atau hasil *plain* citra dengan bentuk citra *grayscale* dan ditampilkan pada *pictureBox Plain Image*. Terdapat juga *DataGridView* sebagai rincian informasi nilai intesitas citra.