Autentikasi Citra Menggunakan *Digital Fragile Watermarking* Dengan Skema SVD (*Singular Value Decomposition*)

Samsu Sempena

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Jalan Ganesha 10,Bandung, Jawa Barat
e-mail: samsu.sempena@yahoo.com

ABSTRAK

Perkembangan teknologi dewasa ini mendorong penyebaran konten digital yang semakin banyak dan cepat. Namun, tentunya ancaman terhadap keaslian konten digital imenjadi isu tersendiri. Oleh karenanya, timbul kebutuhan untuk dapat memastikan keaslian (autentikasi) konten digital seperti citra, suara, dan video. Solusinya dengan menggunakan watermarking yaitu penyusupan data (watermark) berupa text, logo, suara, atau yang lainnya.

Watermarking merupakan aplikasi steganografi yang selain dapat digunakan sebagai metode autentikasi data digital,juga dapat digunakan untuk perlindungan hak cipta, pencegahan penggandaan, memonitor penyebaran data, dan juga *fingerprinting*.

Pada makalah ini, saya membahas secara umum mengenai teknik watermarking beserta jenis-jenisnya dan secara khusus membahas mengenai watermarking pada konten digital bertipe citra(*image*) menggunakan teknik *digital fragile watermarking* dengan skema SVD (*Singular Value Decomposition*) pada domain spasial.

Metode ini memiliki beberapa keunggulan yaitu citra yang telah disisipi watermark memiliki kualitas sangat baik, peka terhadap perubahan sekecil apapun, dan tidak membutuhkan citra asli untuk melakukan autentikasi citra.

Kata kunci: *fragile,watermarking*, autentikasi,SVD

1. PENDAHULUAN

Watermarking ada sejak 700 tahun lalu. Pada akhir abad 13, pabrik kertas di Fabriano, Italia, membuat kertas yang diberi watermark atau tanda air dengan cara menekan bentuk cetakan gambar atau tulisan pada kertas yang baru setengah jadi. Ketika kertas kering maka terbentuklah kertas yang berwatermark.

Ide watermarking pada data digital dikembangkan di Jepang tahun 1990 dan di Swiss tahun 1993. Digital watermarking semakin berkembang seiring dengan semakin meluasnya penggunaan internet yang mendukung penyebaran konten digital seperti video,citra,dan suara.

Watermarking merupakan proses penyisipan data pada objek multimedia yang dapat dideteksi atau diekstrak kemudian untuk keperluan autentikasi maupun hak cipta dari objek digital tersebut. Data digital dapat berupa citra, suara, atau video.

Watermarking memiliki kemiripan dengan steganografi karena watermarking memang merupakan aplikasi dari steganografi. Namun perbedaannya adalah, pada steganografi informasi rahasia disembunyikan dalam media digital dimana media penampung tidak berarti apa-apa, sebaliknya pada watermarking justru media digital tersebut yang dilindungi kepemilikannya dan keasliannya.

Tabel 1. Beberapa tipe watermarking

Kriteria	Jenis
Persepsibilitas	visible/audible, invisible/inaudible
Ketahanan	fragile, semi-fragile, robust
Kebutuhan data asli untuk ekstraksi	blind, informed
Media yang disisipkan	text,audio,video,image
Metode pemrosesan	spatial,spectral

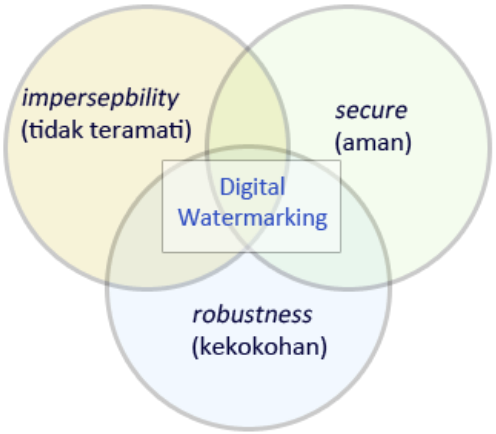
Metode digital fragile watermarking dengan *Singular Value Decomposition* (SVD) menggunakan watermarking dengan jenis berikut :

Tabel 2. Tipe watermarking yang digunakan

Kriteria	Jenis
Persepsibilitas	invisible
Ketahanan	fragile
Kebutuhan data asli untuk ekstraksi	blind
Media yang disisipkan	image
Metode pemrosesan	spatial

Watermarking yang baik harus memiliki beberapa kriteria berikut :

1. *imperceptible*
   watermark tidak dapat dipersepsi secara visual/auditori karena watermark tidak boleh merusak kualitas media host.
2. *robustness*
   kokoh terhadap manipulasi yang ditujukan untuk merusak atau menghapus watermark. Dapat berjenis fragile,semifragile,maupun robust. Jika fragile,maka faktor robustnessnya adalah watermark rusak sekalipun hanya perubahan kecil yang diperbuat pada gambar
3. *secure*
   hanya pihak yang punya otoritas dapat mengakses watermark.



Gambar 1. Kriteria watermarking yang baik

Beberapa serangan pada watermarking yang menunjukkan seberapa tangguh watermarking tersebut :

1. Konversi digital ke analog ke digital (misalnya : *printing* dan *scanning*)
2. Re-sampling
3. Kompresi
4. *Dithering*
5. Rotasi (walaupun hanya 1 derajat)
6. Translasi
7. *Cropping*
8. *Scaling,dll*

2. METODE SVD

2.1 Singular Value Decomposition

Singular value atau *s-numbers* merupakan operator kompak T pada Hilbert space yang didefinisikan sebagai eigenvalue dari operator $\sqrt{T^*T}$ (T^*merupakan adjoint dari T dan akar diambil dari operator sense).

Singular value merupakan bilangan real nonnegative dan biasanya dinyatakan dengan urutan menurun.

Sebuah matriks dapat didekomposisi dalam bentuk USV,dengan U dan V adalah matriks orthogonal dan S adalah matriks diagonal dengan singular value pada diagonalnya. Hal inilah yang disebut singular value decomposition.

$$\mathbf{A}_{nxp} = \mathbf{U}_{nxn}\,\mathbf{S}_{nxp}\,\mathbf{V}^{\mathbf{T}}_{pxp}$$

Sebagai contoh, misalkan matriks

$$A = \begin{bmatrix} 2 & 4 \\ 1 & 3 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

maka,

$$AA^T = \begin{bmatrix} 2 & 4 \\ 1 & 3 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 2 & 4 & 0 & 0 \\ 1 & 3 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 20 & 14 & 0 & 0 \\ 14 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = W$$

dicari eigenvaluenya :

$$\begin{bmatrix} 20-\lambda & 14 & 0 & 0 \\ 14 & 10-\lambda & 0 & 0 \\ 0 & 0 & -\lambda & 0 \\ 0 & 0 & 0 & -\lambda \end{bmatrix} \mathbf{x} = (W - \lambda I)\mathbf{x} = 0$$

Dari |W-λI|=0 didapatkan 4 buah eigenvalue :

$$\lambda = 0, \lambda = 0,$$
$$\lambda = 15 + \sqrt{221.5} \sim 29.883,$$
$$\lambda = 15 - \sqrt{221.5} \sim 0.117$$

Kemudian didapatkan persamaan berikut dengan mensubstitusikan eigen value pertama:

19.883 x1 + 14 x2 = 0
14 x1 + 9.883 x2 = 0
x3 = 0
x4 = 0

Solusi yang memenuhi persamaan di atas x1 = -0.58,x2 = 0.82, dan x3 = x4 =0 (solusi ini menjadi kolom kedua matriks U)

Dari eigenvalue yang lain didapatkan

-9.883 x1 + 14 x2 = 0
14 x1 - 19.883 x2 = 0
x3 = 0
x4 = 0

Solusi yang memenuhi persamaan tersebut x1 = 0.82,x2 = -0.58,dan x3 = x4 = 0. Solusi ini menjadi kolom pertama matriks U

$$U = \begin{bmatrix} 0.82 & -0.58 & 0 & 0 \\ 0.58 & 0.82 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Demikian pula $A^T.A$ akan mengisi kolom dari matriks V, maka dilakukan analisis serupa :

$$A^T.A = \begin{bmatrix} 2 & 4 & 0 & 0 \\ 1 & 3 & 0 & 0 \end{bmatrix} \begin{bmatrix} 2 & 4 \\ 1 & 3 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Didapatkanlah :

$$V = \begin{bmatrix} 0.40 & -0.91 \\ 0.91 & 0.40 \end{bmatrix}$$

Akhirnya,S merupakan akar kuadrat dari AA^T atau A^TA dan didapatkan :

$$S = \begin{bmatrix} 5.47 & 0 \\ 0 & 0.37 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Matriks S ini merupakan singular value dari matriks asal A, dan elemen diagonal dari matriks singular value ini terurut menurun.

Bukti :

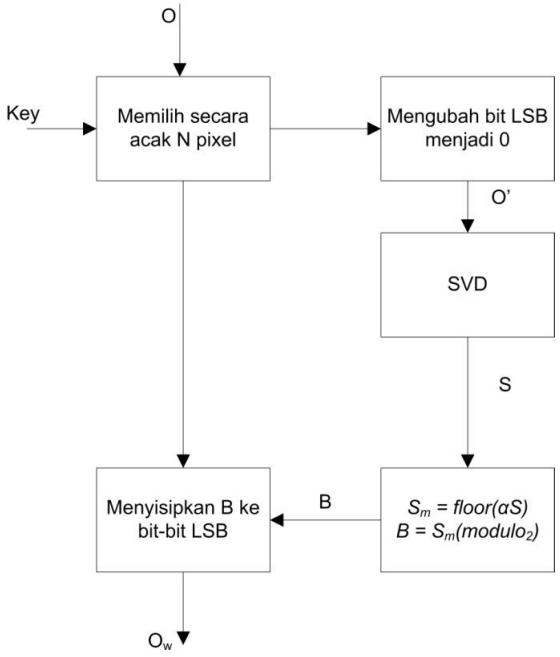
$\mathbf{A}=\mathbf{USV}^T$ and $\mathbf{A}^T=\mathbf{VSU}^T$
$\mathbf{A}^T\mathbf{A} = \mathbf{VSU}^T\mathbf{USV}^T$
$\mathbf{A}^T\mathbf{A} = \mathbf{VS}^2\mathbf{V}^T$
$\mathbf{A}^T\mathbf{AV} = \mathbf{VS}^2$

2.2 Penyisipan watermark

Singular values diubah ke dalam bentuk binary bitnya menggunakan aritmatika modulo. Binary bit dan data autentikasi disisipkan ke *Least Significant Bit* (LSB) dari citra asli. Pixel yang diubah dipilih secara acak dari citra asli. Keuntungan skema ini adalah :

1. Perubahan apapun pada citra berwatermark dapat dideteksi (seperti kompresi, filtering,scaling, pengubahan nilai pixel, cropping, dan menggunakan kunci yang salah)
2. Kualitas dari citra berwatermark sangat baik karena hanya sangat sedikit bit autentikasi yang disisipkan
3. Tidak membutuhkan citra asli untuk melakukan autentikasi citra, karena data yang disisipkan berasal dari citra asli asal.

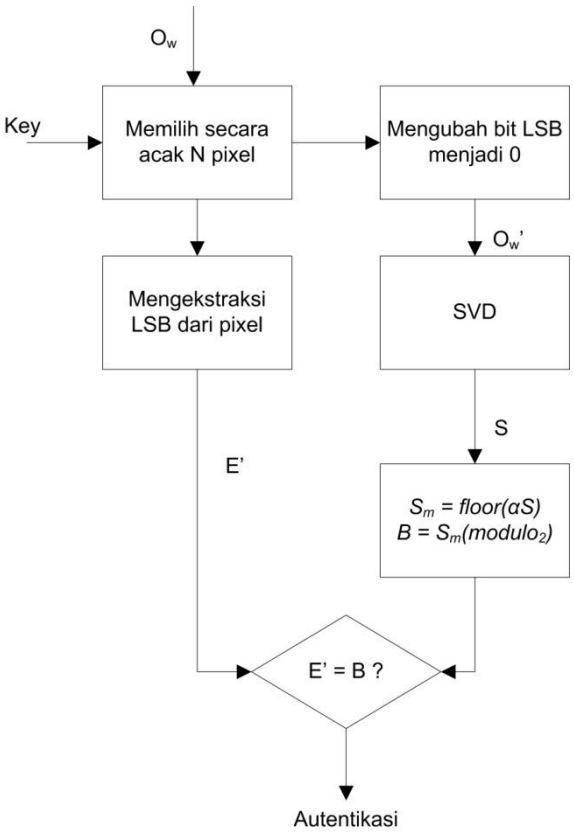
Ide dasarnya dari metode SVD adalah menyisipkan data yang diekstrak dari citra.



Gambar 2. Flowchart penyisipan watermark

Ket :
Key = kunci SVD
O = citra asli (*original image*) berukuran MxN
O' = citra dengan bit LSB telah diset 0
O_w = citra dengan watermark
S = hasil pembangkitan SVD dari key
B = bit-bit watermarking untuk autentikasi

2.3 Ekstraksi dan Verifikasi Watermark



Gambar 3. Flowchart autentikasi watermark

Ket :

Key = kunci SVD
O = citra asli (*original image*) berukuran MxN
O' = citra dengan bit LSB telah diset 0
O_w = citra dengan watermark
S = hasil pembangkitan SVD dari key
B = bit-bit watermarking untuk autentikasi

3. PENGUJIAN

3.1 Aplikasi SVD watermarker

Untuk menguji metode watermarking ini, saya mengembangkan sebuah program aplikasi sederhana untuk menambahkan watermark dengan metode ini pada citra. Program ini dibuat dengan bahasa pemrograman C# dan dapat dioperasikan dengan Microsoft .NET Framework 3.5.

Aplikasi ini terbagi atas beberapa komponen utama :

1. Library matrix menggunakan BlueBit .NET matrix library 5.0 untuk melakukan perhitungan SVD
2. Modul MatriksCitra yang menyediakan fitur untuk operasi pada matriks pixel gambar
3. Modul utility untuk pemilihan pixel secara random berdasarkan key
4. Modul penyisipan watermark
5. Modul pengecekan watermark

Modul MatriksCitra

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Drawing;
using System.Drawing.Imaging;
using Bluebit.MatrixLibrary;

/*
* Kode MatriksCitra.cs bertujuan untuk menyimpan
pixel dari image dalam bentuk matriks
*/
namespace SVDWatermark
{
public class MatriksCitra
{
        public static int RED = 0;
        public static int GREEN = 1;
        public static int BLUE = 2;
        public static int PixelSize = 3;
        public int[][,] matriks; //baris x kolom
        public double[,] matriksRGB;

        public int m;   //baris pada matriks
        public int n;   //kolom pada matriks

        public MatriksCitra(Image img)
        {
            matriks = new int[3][,];
            m = img.Height;
            n = img.Width;
            matriksRGB = new double[m, n];
            for (int x = 0; x < 3; x++)
                matriks[x] = new int[m, n];
            toMatrix(img);
        }

        public MatriksCitra(int Width,int Height)
        {
            matriks = new int[3][,];
            m = Height;
            n = Width;
            matriksRGB = new double[m, n];
            for (int x = 0; x < 3; x++)
                matriks[x] = new int[m, n];
        }

        public Matrix hitungSVD() {
            try
        {
             for (int y = 0; y < m; y++)
                for (int x = 0; x < n; x++)
                matriksRGB[y, x] =
                matriks[BLUE][y, x] * 65536 +
                matriks[GREEN][y, x] * 256 +
                matriks[RED][y, x];

                SVD mySVD = new SVD(new
                Matrix(matriksRGB));
```

```
            return mySVD.S;
        }
    catch { return null; }
}

/// meload gambar ke dalam matriks dengan
pemisahan R,G,B
public void toMatrix(Image img)
{
    Bitmap bm = (Bitmap)img;
    BitmapData bmd = bm.LockBits(new
    Rectangle(0, 0, img.Width, img.Height),
    System.Drawing.Imaging.ImageLockMode.Read
    Only, PixelFormat.Format24bppRgb);

    unsafe
    {
       byte* p = (byte*)(void*)bmd.Scan0;
       int nOffset = bmd.Stride - bmd.Width *
       PixelSize;

       for (int y = 0; y < bmd.Height; y++)
       {
            for (int x=0; x < bmd.Width; x++)
            {
            matriks[BLUE][y, x] = p[0];
            matriks[GREEN][y, x] = p[1];
            matriks[RED][y, x] = p[2];
            matriksRGB[y, x] = p[0] * 65536 +
       p[1] * 256 + p[2];
            p += PixelSize;
            }
            p += nOffset;
       }

    }
    bm.UnlockBits(bmd);
}


/// membuat gambar dari matriks yang ada
public Image toImage()
{
    Image img = new Bitmap(n, m);
    Bitmap bm = (Bitmap)img;
    BitmapData bmd = bm.LockBits(new
    Rectangle(0, 0, img.Width, img.Height),
    System.Drawing.Imaging.ImageLockMode.Writ
    eOnly, PixelFormat.Format24bppRgb);
    int PixelSize = 3;

    unsafe
    {
       byte* p = (byte*)(void*)bmd.Scan0;
       int nOffset = bmd.Stride - bmd.Width *
       PixelSize;

       for (int y = 0; y < bmd.Height; y++)
       {
            for (int x=0; x < bmd.Width; x++)
            {
            p[0] = (byte)matriks[BLUE][y, x];
            p[1] = (byte)matriks[GREEN][y, x];
            p[2] = (byte)matriks[RED][y, x];
            p += PixelSize;
          }
          p += nOffset;
      }
    }
    bm.UnlockBits(bmd);
    return img;
      }
    }
}
```

Modul utility

```
public class Utility
{
   public static Random getRandom(string
originalKey)
   {
      int randomseed = 0;
      string key = originalKey;
      for (int i = 0; i < key.Length; i++)
        randomseed += Convert.ToInt16(key[i]);
      Random rnd = new Random(randomseed);
      return rnd;
   }
}
```

Modul penyisipan watermark

(hanya kode untuk menyisipkan watermark yang disertakan)
Bit autentikasi disisipkan pada bit LSB dari pixel yang terpilih oleh fungsi random berdasarkan key dari pengguna.

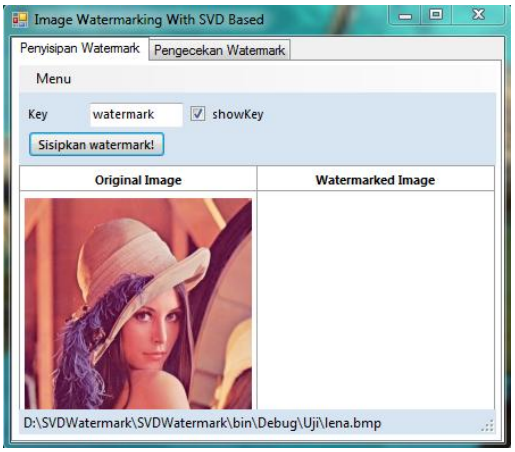
```
/// fungsi untuk menambahkan watermark ke gambar
void embedWatermark(string Key, ref MatriksCitra
originalImage)
{
   /*mendapatkan kelas random untuk key*/
   Random rnd=Utility.getRandom(textBoxKey.Text);

   int N = originalImage.m > originalImage.n ?
   originalImage.n : originalImage.m;
   int[] selectedPixels = new int[N];
   for (int i = 0; i < N; i++)
   {
        /*memilih sejumlah pixel dari gambar
        berdasarkan key*/
        selectedPixels[i] =
        rnd.Next(originalImage.m *
        originalImage.n - 1);
       /*membuat bit LSB dari pixel terpilih
       menjadi 0*/
       int m=selectedPixels[i]/ originalImage.n;
        int n=selectedPixels[i]% originalImage.n;

        if(originalImage.matriks[MatriksCitra.RED
        ][m, n] % 2 != 0)

                originalImage.matriks[MatriksCitra
                .RED][m, n] -= 1;
       }

       *menghitung nilai SVD dari gambar*/
       Matrix S = originalImage.hitungSVD();
       Vector SV = S.Diagonal();

       /*menyisipkan data dari image berdasarkan
       nilai SVD ke citra*/
       for (int i = 0; i < N; i++)
       {
        int m=selectedPixels[i]/originalImage.n;
        int n=selectedPixels[i]%originalImage.n;
        if (Convert.ToInt32(SV[i] % 2) == 1)
        {
originalImage.matriks[MatriksCitra.RED][m,n]+= 1;
```

```
        }
      }
  }
```

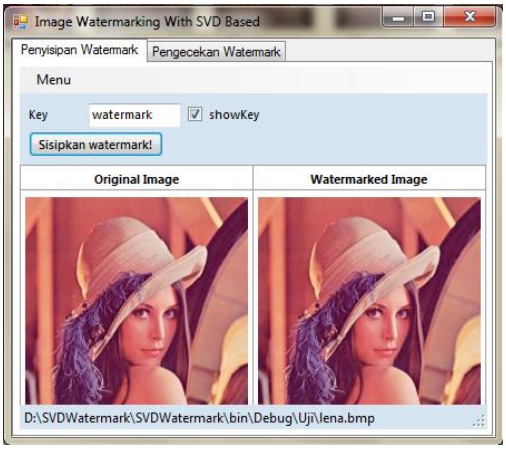
Modul pengecekan watermark
(hanya kode untuk pengecekan watermark yang disertakan)

```
/// fungsi untuk membandingkan nilai SVD dari
data yang terekstrak dengan data yang disisipkan
void checkWatermark(string originalKey,ref
MatriksCitra watermarkedImage)
{
  /*1. mendapatkan key yang disisipkan*/
  Random rnd = Utility.getRandom(originalKey);
  int N = watermarkedImage.m > watermarkedImage.n
    ? watermarkedImage.n : watermarkedImage.m;
   int[] selectedPixels = new int[N];

  for (int i = 0; i < N; i++)
  {
    /*2. memilih sejumlah pixel dari gambar
    berdasarkan key*/
    selectedPixels[i] =
    rnd.Next(watermarkedImage.m *
    watermarkedImage.n - 1);
    /*3. mendapatkan bit LSB dari pixel yang
    terpilih sekaligus mengeset LSB pixel
    terpilih menjadi 0*/
    int m = selectedPixels[i]/watermarkedImage.n;
    int n = selectedPixels[i]%watermarkedImage.n;
    selectedPixels[i] =
    watermarkedImage.matriks[MatriksCitra.RED][m,
    n] % 2;
    if(watermarkedImage.matriks[MatriksCitra.RED]
    [m, n] % 2 != 0)
        watermarkedImage.matriks[MatriksCitra.RED
        ][m, n] -= 1;
    }
    //selectedPixel sekarang berisi data
    autentikasi yang diekstrak dari gambar
    /*4. mendapatkan data watermark dengan
    metode SVD dari gambar*/
    Matrix S = watermarkedImage.hitungSVD();
    Vector SV = S.Diagonal();

    /*5. membandingkan keduanya*/
    int diff = 0;   //jumlah bit LSB berbeda
    for (int i = 0; i < N; i++)
    {
        if (selectedPixels[i] !=
        Convert.ToInt32(SV[i] % 2)%2) diff++;
    }
    if (diff == 0)
        MessageBox.Show("Gambar belum
        dimodifikasi!");
    else
        MessageBox.Show("Gambar telah
        dimodifikasi \natau tidak
        berwatermark\natau kunci yang dimasukkan
        salah! \n[beda " + diff + " LSB]");
}
```

Berikut ini akan dilakukan pengujian watermarking terhadap citra Lena.bmp. Pengujian dilakukan terhadap citra dengan format BMP 24 bit.

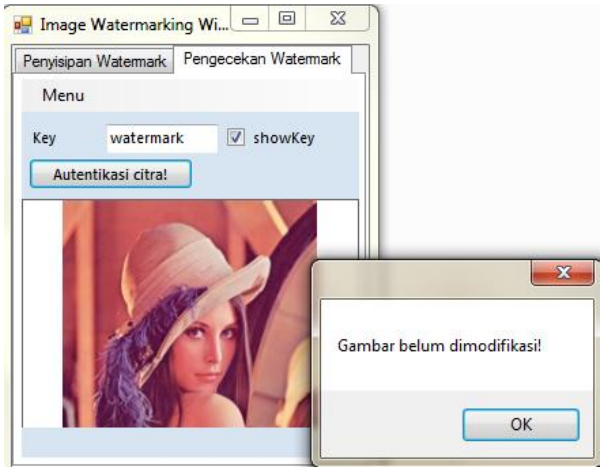


Gambar 4. Citra original Lena

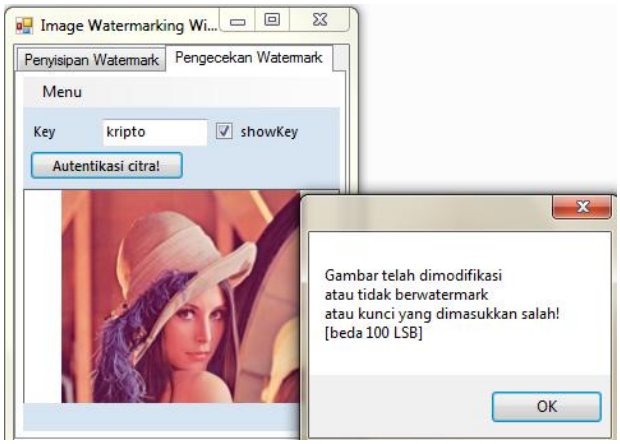


Gambar 5. Citra Lena sebelum dan sesudah disisipkan watermark

Dapat diamati bahwa kualitas citra berwatermark sangat bagus karena maksimal hanya sejumlah min(M,N) bit dari LSB (*Least Significant Bit*) citra berukuran M*N pixel 24 bit yang berubah nilainya.

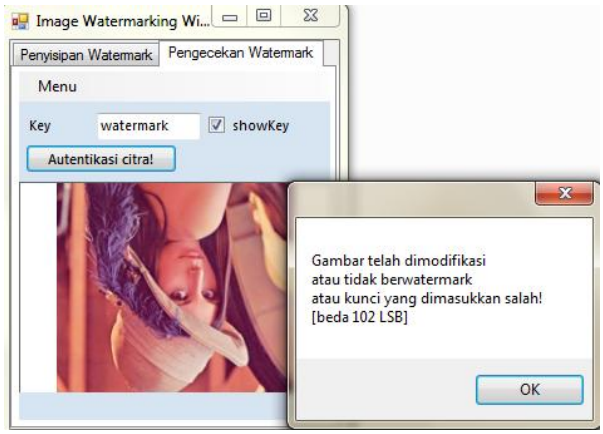


Gambar 6. Autentikasi citra lena yang telah diwatermark

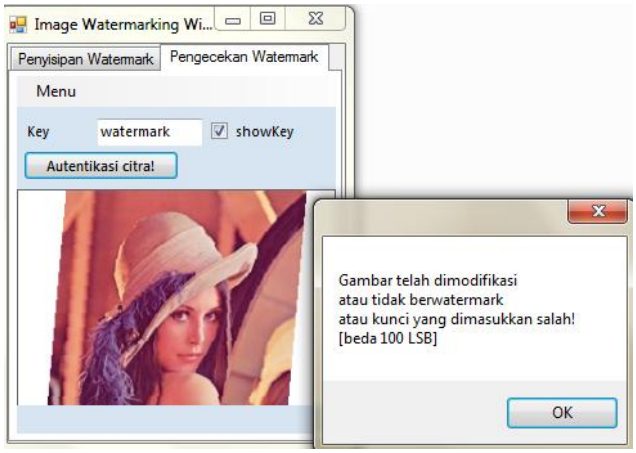


Gambar 7. Autentikasi gagal karena kunci salah

Gambar 8. Autentikasi gagal karena citra Lena telah ditambahkan tanda silang



Gambar 9. Autentikasi gagal karena citra Lena dirotasi 180º



Gambar 10. Autentikasi gagal karena citra Lena mengalami transformasi geometri

Pada prinsipnya, sekecil apapun perubahan pada citra akan terdeteksi. Alasannya perubahan satu pixel saja dapat mengubah singular value dari citra yang digunakan sebagai data autentikasi citra.

3.2 Pengembangan lebih lanjut

1. Pada citra bertipe yang disimpan dengan pengkompresian dibutuhkan penanganan khusus karena bit LSB nya mungkin hilang atau menjadi inkonsisten. Hal ini dapat ditangani dengan menggunakan skema SVD pada ranah transform.

2. Penggunaan SVD dapat dilakukan dengan lokalisasi sehingga dapat dilihat perubahan terjadi pada bagian mana dari citra.

4. KESIMPULAN

Citra dapat dinyatakan dengan suatu nilai yang unik dikenal dengan singular value. Nilai ini sangat sensitive terhadap modifikasi dan dapat digunakan untuk mengautentikasi citra tersebut.

Skema SVD sudah cukup baik untuk memberikan watermark karena telah berhasil mengidentifikasi perubahan pada citra berwatermark yang sangat kecil sekalipun, memiliki kualitas citra berwatermark yang tinggi, serta tidak membutuhkan citra asli untuk autentikasi.

REFERENSI

[1] Cahyana,T.Basaruddin,Danang Jaya, Teknik Watermarking Citra Berbasis SVD. 2007.

[2] http://web.mit.edu/be.400/www/SVD/Singular_Value_Decomposition.htm. Diakses 24 Maret 2010.

[3] Munir, Rinaldi, "Diktat Kuliah IF3058 Kriptografi", Program Studi Teknik Informatika, ITB,Bandung, 2006.

[4] Shang-Cheal Byun, Sang-Kwang Lee, Ahmed H. Tewlick. A SVD-Based Fragile Watermarking Scheme for Image Authentication. 2003.