Pemisahan Warna dengan Pengacakan Pixel untuk Enkripsi sebagai Modifikasi Kriptografi Visual

Bobby H. Suryanaga - 13508022[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]if18022@if.itb.ac.id

***Abstraksi*— Saat ini, foto digital sangat banyak tersebar karena memiliki kepraktisan dan kemudahan dalam melakukan pencetakan, berbagi dengan orang lain, pengambilan gambar, pengeditan gambar, dan lain-lain. Oleh karena itu, diperlukan suatu mekanisme untuk dapat menyimpannya dan membagikannya dengan aman.**

Kriptografi visual adalah suatu cara khusus untuk mengenkripsi gambar/citra dengan membagi gambar tersebut menjadi sejumlah bagian. Pada kriptografi visual, bagian-bagian yang dipisahkan ini, jika ditumpukkan akan menghasilkan gambar aslinya kembali, meskipun tidak sama persis dengan gambar awal sebelum dienkripsi karena adanya noise.

Proses pengacakan pixel dilakukan dengan menggunakan sebuah kunci, sehingga untuk mendapatkan citra awal, seseorang perlu memiliki ketiga bagian citra dan kunci yang benar untuk mendekripsikannya. Kunci yang digunakan untuk melakukan proses enkripsi dan dekripsi sama. Kunci yang dimasukkan berupa string.

***Kata kunci*—kriptografi visual, pemisahan warna, pengacakan pizel.**

I. PENDAHULUAN

Pada saat ini, teknologi internet dan dijital sudah sangat berkembang. Akses internet dengan mudah didapatkan di mana-mana. Setiap orang dapat menyimpan berkas-berkas di internet, termasuk gambar.

Gambar yang disimpan di internet merupakakan gambar dijital. Semakin berkembangnya teknologi menyebabkan dengan mudahnya orang mengambil foto dijital, baik dengan kamera handphone atau dengan kamera dijital. dan saat ini, foto yang diambil dari kamera handphone dapat dengan langsung diupload ke situs jejaring social yang diinginkan.

Penyebaran gambar dijital setelah masuk ke internet sangat cepat. Hal ini membawa dampak yang baik dan buruk. Di sisi positif, kita dapat berbagi foto dengan teman-teman dan kerabat kita dengan mudah, saling mengomentari foto-foto tersebut. Di sisi negatif, banyak orang yang tidak bertanggung jawab memanfaatkan cepatnya penyebaran tersebut sebagai sarana dalam menyebarkan konten yang tidak sesuai.

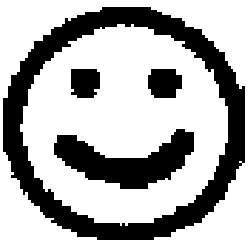
Untuk mencegah hal tersebut, kita perlu melakukan perlindungan terhadap foto-foto yang kita anggap tidak pantas untuk disebarluaskan. Salah satu caranya adalah dengan menggunakan kriptografi visual.

II. KRIPTOGRAFI VISUAL

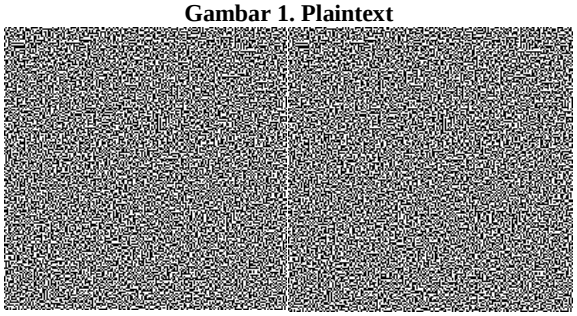
Kriptografi visual diperkenalkan oleh Moni Naor dan Adi Shamir dalam jurnal Eurocrypt'94. Metode kriptografi ini digunakan khusus untuk enkripsi gambar/citra.

Proses enkripsi dilakukan dengan membagi citra menjadi sejumlah bagian (share). Proses dekripsi gambar dilakukan dengan menumpuk sejumlah citra bagian sehingga menghasilkan citra yang mirip atau sama dengan citra aslinya. Proses dekripsi tersebut tidak membutuhkan komputasi.

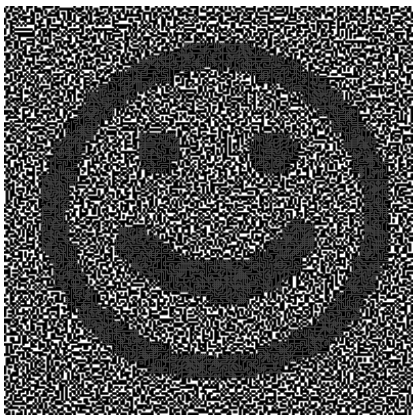
Contohnya adalah :



Gambar 1. Plaintext



Gambar 2. Ciphertext



Gambar 3. Tumpukan share 1 & share 2

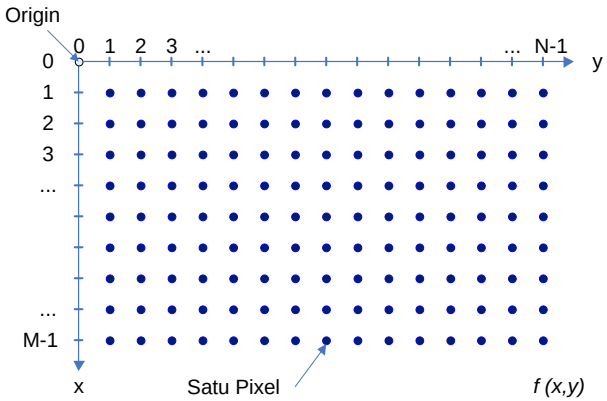
Kriptografi visual diaplikasikan pada citra digital. Citra digital memiliki definisi sebagai berikut.

fungsi larik dua dimensi f(x,y)

x, y : koordinat spasial

f : intensitas warna

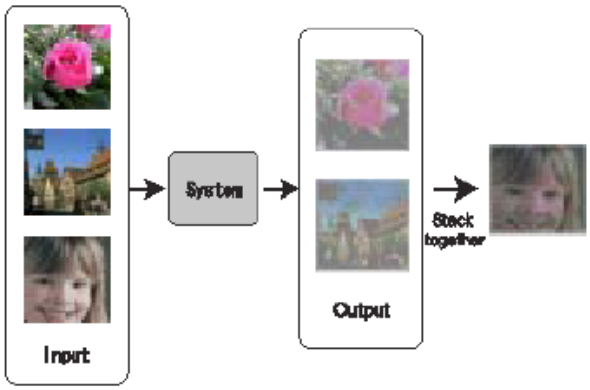
Citra digital terdiri atas pixel-pixel. Pixel merupakan elemen pada citra digital yang memiliki lokasi (x,y) dan nilai f(x,y). Nama lain pixel: picture elements, image elements, pels.[1]



Gambar 4. Representasi citra digital

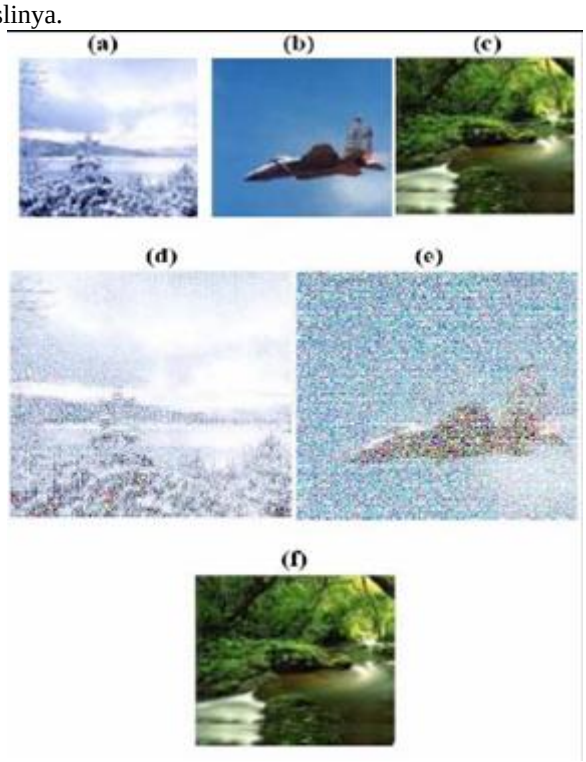
Kelemahan dari kriptografi visual adalah memiliki hasil dekripsi yang tidak sama dengan citra aslinya, citra hasil dekripsi mengandung noise, dan share tidak memiliki makna sehingga dapat menimbulkan kecurigaan bahwa gambar tersebut memiliki pesan rahasia.

Untuk menghilangkan kecurigakan, digunakan steganografi untuk menyimpan share. Cover digunakan untuk menyimpan share. Cover yang telah disisipi share disebut dengan camouflage.



Gambar 5. Steganografi pada Kriptografi Visual

Selain itu, terdapat pula metode yang diperkenalkan Chang dan Yu pada tahun 2000. Metode ini dapat menghasilkan hasil dekripsi yang sama dengan citra aslinya.



Gambar 6. Kriptografi visual Chang dan Yu, (a) cover1, (b) cover2, (c) plaintext, (d) camouflage1, (e) camouflage2, (f) hasil dekripsi.

III. Enkripsi Menggunakan Pemisahan Warna dan Pengacakan Pixel

Pada metode kriptografi visual menggunakan pemisahan warna pada makalah ini, citra digital tidak dibagi menjadi subpixel, melainkan unsur warna pembentuk citra disebarkan ke tiga buah citra share. Unsur warna merah, hijau, dan biru disebarkan ke tiga buah citra enkripsi. Satu citra share tidak hanya mengandung sebuah unsur warna saja, melainkan ketiga unsur warna tergantung pixelnya.

Sebagai contoh, misalkan pixel pada posisi (0,0), unsur warna merah pixel tersebut disimpan pada citra share

pertama, unsur warna hijau disimpan pada citra share kedua, dan unsur warna biru disimpan ke citra share ketiga. Namun pada pixel dengan posisi (1,0), unsur warna merah tersimpan tidak pada citra share pertama, melainkan pada citra share ketiga, unsur warna hijau pada citra share pertama, dan unsur warna biru pada citra share ketiga.

Unsur-unsur warna pada pixel (0,0) dari citra awal tidak disimpan ke pixel (0,0) citra share, melainkan diacak menggunakan sebuah kunci yang dimasukkan pengguna.

Proses pengacakan pixel serupa dengan penggunaan seed pada steganografi. Dari kata kunci yang dimasukkan dilakukan penghitungan untuk mendapatkan seed.

Berikut ini adalah pseudo code dari algoritma yang digunakan untuk melakukan enkripsi dan dekripsi.

```
procedure Encrypt(string Key)
{prosedur untuk menghasilkan tiga buah share dari sebuah citra(plain) dengan menggunakan kunci Key
Masukan: Key
I.S.: plain terdefinisi
F.S.: cipher1, cipher2, cipher3 (3 buah share) terdefinisi}
Deklarasi:
index, i, j: integer
B1, B2, B3: bitmap(plain.Width, plain.Height)
Algoritma:
Array of Point P ← pseudoRandomize(Key, plain.Width, plain.Height, plain.Width * plain.Height);
index ← 0;
for j ← 0 to plain.Height - 1
  for i ← 0 to plain.Width - 1
    Color c ← plain.GetPixel(i, j);
    if (i mod 3 = 0)
      B1.SetPixel(P[index].X, P[index].Y, Color.FromArgb(c.R, (i / 2 * c.G) mod 255, (j / 2 * c.R) mod 255))
      B2.SetPixel(P[index].X, P[index].Y, Color.FromArgb((i / 2 * c.R) mod 255, c.G, (j / 2 * c.B) mod 255))
      B3.SetPixel(P[index].X, P[index].Y, Color.FromArgb((i / 2 * c.B) mod 255, (j / 2 * c.G) mod 255, c.B))
    else if (i mod 3 = 1)
      B3.SetPixel(P[index].X, P[index].Y, Color.FromArgb(c.R, (i / 2 * c.B) mod 255, (j / 2 * c.G) mod 255))
      B1.SetPixel(P[index].X, P[index].Y, Color.FromArgb((i / 2 * c.G) mod 255, c.G, (j / 2 * c.R) mod 255))
      B2.SetPixel(P[index].X, P[index].Y, Color.FromArgb((i / 2 * c.R) mod 255, (j / 2 * c.B) mod 255, c.B))
    else
      B2.SetPixel(P[index].X, P[index].Y, Color.FromArgb(c.R, (i / 2 * c.B) mod 255, (j / 2 * c.G) mod 255))
      B3.SetPixel(P[index].X, P[index].Y, Color.FromArgb((i / 2 * c.B) mod 255, c.G, (j / 2 * c.R) mod 255))
      B1.SetPixel(P[index].X, P[index].Y, Color.FromArgb((i / 2 * c.R) mod 255, (j / 2 * c.G) mod 255, c.B))
    endif
    index++
  endfor
endfor
cipher1 ← B1;
cipher2 ← B2;
cipher3 ← B3;

procedure Decrypt(string Key)
{prosedur untuk menghasilkan palin dari 3 buah share dengan menggunakan kunci Key
Masukan: Key
I.S.: cipher1, cipher2, cipher3 terdefinisi
F.S.: plain terdefinisi}
Deklarasi:
index, i, j: integer
Algoritma:
array of Point P ← pseudoRandomize(Key, cipher1.Width, cipher1.Height, cipher1.Width * cipher1.Height)
index ← 0
for j ← 0 to cipher1.Height - 1
  for i ← 0 to cipher1.Width - 1
    Color c1 ← cipher1.GetPixel(P[index].X, P[index].Y)
    Color c2 ← cipher2.GetPixel(P[index].X, P[index].Y)
    Color c3 ← cipher3.GetPixel(P[index].X, P[index].Y)
    if (i mod 3 = 0)
      plain.SetPixel(i, j, Color.FromArgb(c1.R, c2.G, c3.B))
    else if (i mod 3 = 1)
      plain.SetPixel(i, j, Color.FromArgb(c3.R, c1.G, c2.B))
```

```
    else
        plain.SetPixel(i,          j,
Color.FromArgb(c2.R, c3.G, c1.B))
      endif
    index++
    endfor
endfor

function isArrPenuh(array of Point
arrP, int angkaKosong) --> boolean
{fungsi untuk mengembalikan apakan
array of point arrP telah penuh
angkaKosong mewakili nilai integer
yang diinisiasi pada saat instansiasi
Point pada array of point
Masukan: arrP, angkaKosong
Keluaran: boolean}
Deklarasi:
i : integer
Algoritma:
for i ← 0 to arrP.Length - 1
  if ((arrP[i].X = angkaKosong) and
(arrP[i].Y = angkaKosong))
    → false
  endif
endfor
→ true

function pseudoRandomize(String key,
int imgWidth, int imgHeight, int
jmlPixel) → array of Point
{fungsi yang mengembalikan urutan
posisi pixel yang diacak berdasarkan
Key dari masukan tinggi citra dalam
pixel imgHeight, lebar citra dalam
pixel imgWidth, jumlah pixel citra
jmlPixel
Masukan: key, imgWidth, imgHeight,
jmlPixel
Keluaran: array of Point}
Deklarasi:
hasil: array of Point[jmlPixel]
pTemp, pTempTemp: Point
seed, i, j: integer
matrix: array of array of boolean
Algoritma:
pTemp ← new Point(0, 0)
pTempTemp ← new Point(-1, -1)
seed ← 0
for j ← 0 to imgHeight - 1
  for i ← 0 to imgWidth - 1
    matrix[i, j] ← false
  endfor
endfor
for i ← 0 to key.Length - 1
  seed += (int)key[i];
endfor
for i ← 0 to hasil.Length - 1
  hasil[i] ← new Point(-1, -1)
endfor
for offset ← 0 to hasil.Length - 1
  pTemp.X ← (seed * offset) mod
imgWidth
  pTemp.Y ← (seed * (offset + 2)) mod
imgHeight
  while (matrix[pTemp.X, pTemp.Y] and
(!isArrPenuh(hasil, -1))) do
    pTempTemp.X ← pTemp.X
    pTempTemp.Y ← pTemp.Y
    if (pTemp.X < imgWidth - 1)
      pTemp.X++
    else
      pTemp.X ← 0
      if (pTemp.Y < imgHeight - 1)
        pTemp.Y++
      else
        pTemp.Y ← 0;
      endif
    endif
  endwhile
  hasil[offset].X ← pTemp.X
  hasil[offset].Y ← pTemp.Y
  matrix[pTemp.X, pTemp.Y] ← true
endfor
→ hasil
```

Berikut ini adalah bagan mengenai proses enkripsi dan dekripsi pada algoritma pemisahan warna dan pengacakan pixel pada makalah ini:

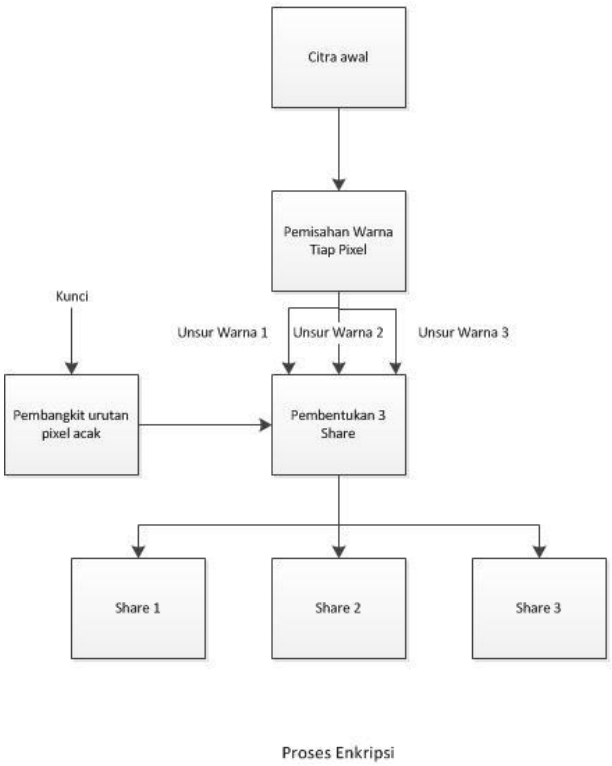


Diagram 1. Proses Enkripsi

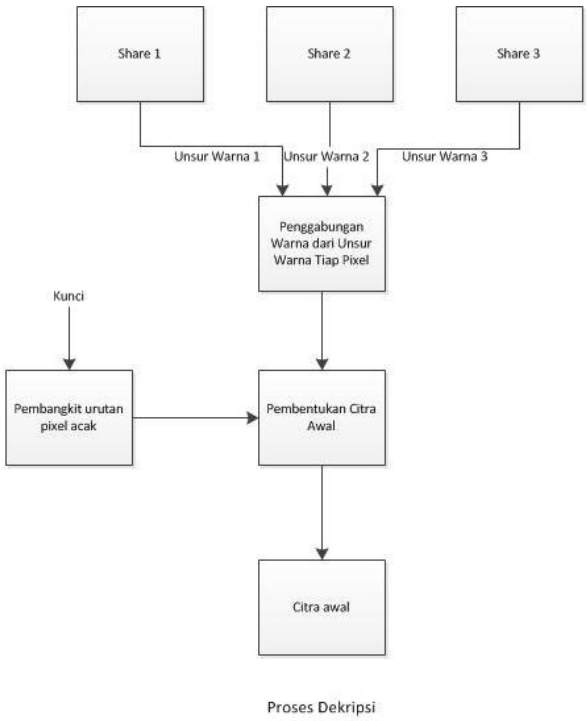
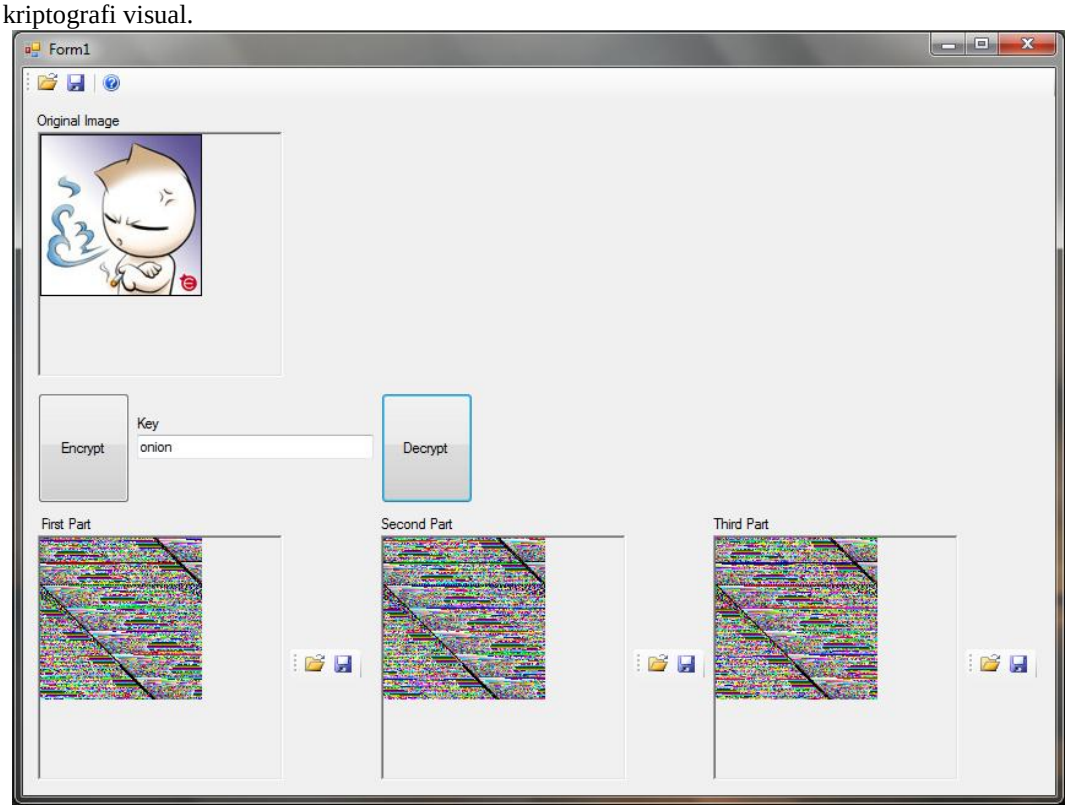


Diagram 2. Proses Dekripsi

Berikut ini adalah tampilan program yang digunakan untuk melakukan enkripsi dan dekripsi algoritma pemisahan warna dengan pengacakan pixel sebagai modifikasi kriptografi visual.



Gambar 7. Tampilan program yang dibuat untuk melakukan enkripsi dan dekripsi dengan pemisahan warna dan pengacakan pixel.

Berikut ini contoh dari citra digital berwarna yang dienkripsi menggunakan algortima kriptografi visual

Gambar 8. Citra awal

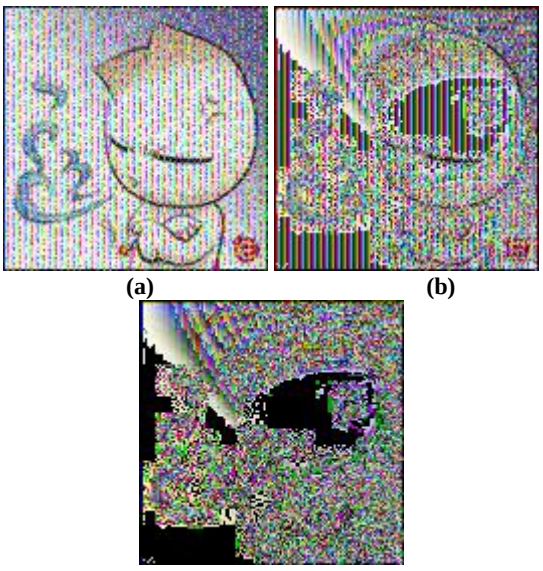
(a) (b)

(c)

Gambar 9. Citra share dengan kata kunci 'onion' (a) share1 (b) share2 (c) share3

Gambar 10. Citra hasil dekripsi menggunakan ketiga share dengan kata kunci yang tepat.

Gambar 11. Dekripsi menggunakan kata kunci yang tidak sesuai ('bawang')



(a) **(b)**

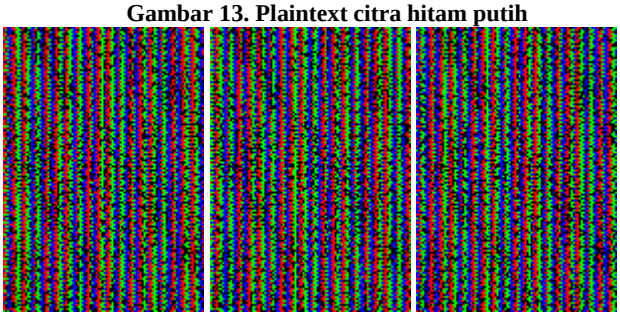
(c)

Gambar 12. (a). Dekripsi menggunakan 2 share dengan kata kunci yang benar. (b) Dekripsi dengan urutan yang salah (2,1,3) (c) Dekripsi dengan urutan yang salah (3, 1, 2)

IV. PERBANDINGAN DENGAN METODE KRIPTOGRAFI VISUAL LAINNYA

Kriptografi visual dengan pemisahan warna dan pengacakan pixel memilki beberapa keunggulan dengan metode kriptografi visual lainnya. Seperti teah diketahui sebelumnya, cara ini menggunakan pemisahan warna dari citra digital, artinya cara ini mendukung citra digital yang berwarna. Cara ini juga dapat digunakan untuk citra hitam putih, contohnya:

Gambar 13. Plaintext citra hitam putih



Gambar 14. Share 1, 2, dan 3 dengan kunci 'gajah'

Gambar 15. Hasil dekripsi

Gambar 16. Hasil dekripsi menggunakan kunci yang tidak sesuai.(‘ganesha’)

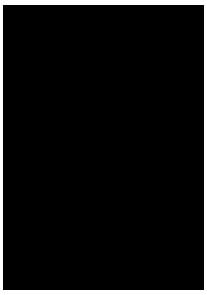
Gambar 17. Hasil dekripsi dengan menggunakan 2 buah share yang sesuai dengan kunci yang benar.

Kelebihan lain dari metode kriptografi visual ini adalah tingkat keamanannya. Selain harus memiliki share yang lengkap untuk medapatkan citra asli, pengguna yang akan mendekripsikan share juga harus memiliki kunci yang tepat. Kurangnya share yang dimiliki masih dapat menampilkan citra yang mirip dengan citra aslinya, namun jika pengguna mendekripsikan share dengan kuci yang salah, hasil dekripsi akan tidak memiliki arti.

Meskipun memiliki kelebihan, cara ini juga memiliki kekurangan. Kekurangan pertama adalah share yang dihasilkan berupa gambar acak yang tentunya akan menimbulkan kecurigaan akan adanya pesan rahasia. Kekurangan kedua adalah perlunya komputasi dalam melakukan dekripsi. Tidak seperti kriptografi visual lainnya yang hanya perlu menumpukkan gambar untuk mendapatkan citra awal, penumpukkan gambar dengan cara ini tidak akan menghasilkan apa-apa, hanya gambar acak akibat adanya pengacakan pixel dan penggantian unsur warna pada enkripsi.

Pengacakan pixel yang digunakan juga masih memerlukan komputasi yang cukup berat sehingga memerlukan waktu yang lama untuk mengenkripsi dan mendekripsikan citra yang berukuran cukup besar.

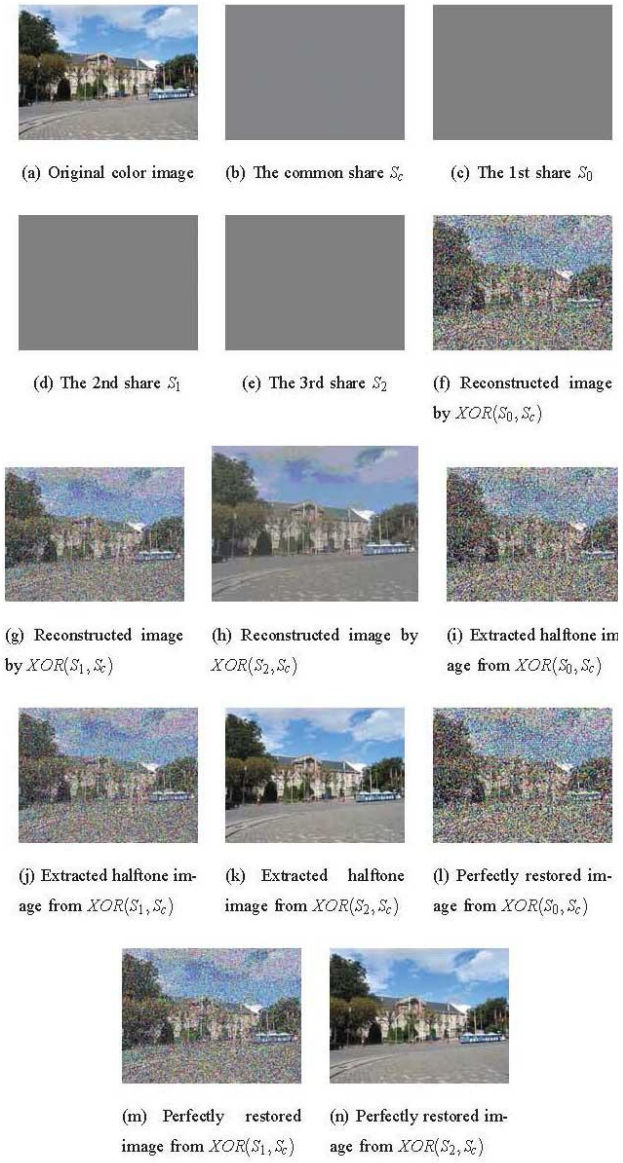
Kekurangan ketiga adalah urutan share yang digunakan harus tepat untuk mendapatkan citra asli. Kesalahan dalam urutan akan menghasilkan gambar yang tidak berarti.



Gambar 18. Hasil dekripsi dengan menggunakan ketiga share dengan urutan yang tidak sesuai. (3, 1 ,2)

Kekurangan terakhir adalah, kata kunci yang tidak sesuai mungkin dapat melakukan dekripsi dengan benar. Hal tersebut karena pada penghitungan seed memungkinkan dua kata yang berbeda menghasilkan angka yang sama. Pada makalah ini, seed dihitung dengan cara menjumlahkan nilai integer ASCII dari tiap karakter pada kunci. Dengan demikian, jika kunci yang digunakan adalah ‘abc’, dekripsi dapat dilakukan dengan menggunakan ‘bbb’ atau ‘cba’ atau yang lainnya.

Jika dibandingkan dengan kriptografi visual yang diajukan Duo Jin, Wei-Qi Yan, Mohan S. Kankanhalli dari School of Computing, National University of Singapore, Progressive Color Visual Cryptography, cara ini masih kurang efektif.



Gambar 19. Multiresolution visual cryptography Jin, Yan, Kankanhalli

Metode Jin, Yan, dan Kankanhalli dapat mengembalikan citra yang sama seperti semula dari share yang tampak hanya seperti gambar berwarna abu-abu dengan noise dengan perhitungan yang sederhana, yaitu menggunakan XOR.

V. Saran Untuk Algoritma Baru

Perbaikan yang bisa diambil dari perbandingan dengan metode kriptografi visual lainnya adalah dengan menggunakan metode yang sesedikit mungkin menggunakan komputasi agar proses dekripsi dapat dengan cepat dilakukan. Selain itu, sebaiknya digunakan juga metode steganogafi agar tidak menimbulkan kecurigaan akan adanya pesan tersembunyi pada share. Camouflage yang dihasilkan harus memiliki noise seminimal mungkin.

Penggunaan kunci cukup baik mengingat adanya peningkatan keamanan kalau share dari citra diketahui orang yang tidak berhak. Hindari juga kemungkinan dua kata yang berbeda dapat digunakan sebagai kunci untuk melakukan enkripsi dan dekripsi untuk mencegah kemungkinan dapat dilakukannya dekripsi menggunakan kata kunci yang tidak sesuai.

VI. Kesimpulan

Metode pemisahan warna dengan pengakan pixel memiliki tingkat keamanan yang cukup baik, namun memiliki kekurangan pada kebutuhan komputasi yang cukup berat pada proses enkripsi dan dekripsi.

VII. Ucapan Terima Kasih

Atas terselesaikannya makalah ini, saya mengucapkan terima kasih kepada dosen pembimbing dalam perkuliahan Kriptografi, Pak Rinaldi Munir, karena telah memberikan pengetahuan dasar yang memungkinkannya dibuatnya makalah ini.

References

[1] Munir, Rinaldi, Kriptografi Visual, Bahan Tambahan IF3058 Kriptografi, 2011.
[2] Jin, Duo, Yan, Wei-Qi, and Mohan S. Kankanhalli, Progressive Color Visual Cryptography, 2004.

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 1 Mei 2010

ttd

Bobby H. Suryanaga – 13508022

Lampiran

Source code yang dibuat untuk program pengenkripsi dan dekripsi (hanya bagian pemroses citra)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Drawing;

namespace VisualCryptography
{
    class ImageProcessor
    {
        public ImageProcessor() { }

        private Bitmap plain;
        private Bitmap cipher1;
        private Bitmap cipher2;
        private Bitmap cipher3;

        public Bitmap getPlain() { return plain; }
        public Bitmap getCipher1() { return cipher1; }
        public Bitmap getCipher2() { return cipher2; }
        public Bitmap getCipher3() { return cipher3; }

        public void setPlain(Bitmap B) { plain = B; }
        public void setCipher1(Bitmap B) { cipher1 = B; }
        public void setCipher2(Bitmap B) { cipher2 = B; }
        public void setCipher3(Bitmap B) { cipher3 = B; }

        public void Encrypt(string Key)
        {
            Bitmap B1 = new Bitmap(plain);
            Bitmap B2 = new Bitmap(plain);
            Bitmap B3 = new Bitmap(plain);
            Point[] P = pseudoRandomize(Key, plain.Width, plain.Height, plain.Width
* plain.Height);
            int index = 0;
            for (int j = 0; j < plain.Height; j++)
            {
                for (int i = 0; i < plain.Width; i++)
                {
                    Color c = plain.GetPixel(i, j);
                    if (i % 3 == 0)
                    {
                        B1.SetPixel(P[index].X, P[index].Y, Color.FromArgb(c.R, (i /
2 * c.G) % 255, (j / 2 * c.R) % 255));
                        B2.SetPixel(P[index].X, P[index].Y, Color.FromArgb((i / 2 *
c.R) % 255, c.G, (j / 2 * c.B) % 255));
                        B3.SetPixel(P[index].X, P[index].Y, Color.FromArgb((i / 2 *
c.B) % 255, (j / 2 * c.G) % 255, c.B));
                    }
                    else if (i % 3 == 1)
                    {
                        B3.SetPixel(P[index].X, P[index].Y, Color.FromArgb(c.R, (i /
2 * c.B) % 255, (j / 2 * c.G) % 255));
                        B1.SetPixel(P[index].X, P[index].Y, Color.FromArgb((i / 2 *
c.G) % 255, c.G, (j / 2 * c.R) % 255));
                        B2.SetPixel(P[index].X, P[index].Y, Color.FromArgb((i / 2 *
c.R) % 255, (j / 2 * c.B) % 255, c.B));
                    }
                    else
                    {
                        B2.SetPixel(P[index].X, P[index].Y, Color.FromArgb(c.R, (i /
2 * c.B) % 255, (j / 2 * c.G) % 255));
```

```
                        B3.SetPixel(P[index].X, P[index].Y, Color.FromArgb((i / 2 *
c.B) % 255, c.G, (j / 2 * c.R) % 255));
                        B1.SetPixel(P[index].X, P[index].Y, Color.FromArgb((i / 2 *
c.R) % 255, (j / 2 * c.G) % 255, c.B));
                    }
                    index++;
                }
            }
            cipher1 = B1;
            cipher2 = B2;
            cipher3 = B3;
        }

        public void Decrypt(string Key)
        {
            plain = new Bitmap(cipher1.Width, cipher1.Height);
            Point[] P = pseudoRandomize(Key, cipher1.Width, cipher1.Height,
cipher1.Width * cipher1.Height);
            int index = 0;
            for (int j = 0; j < cipher1.Height; j++)
            {
                for (int i = 0; i < cipher1.Width; i++)
                {
                    Color c1 = cipher1.GetPixel(P[index].X, P[index].Y);
                    Color c2 = cipher2.GetPixel(P[index].X, P[index].Y);
                    Color c3 = cipher3.GetPixel(P[index].X, P[index].Y);
                    if (i % 3 == 0)
                    {
                        plain.SetPixel(i, j, Color.FromArgb(c1.R, c2.G, c3.B));
                    }
                    else if (i % 3 == 1)
                    {
                        plain.SetPixel(i, j, Color.FromArgb(c3.R, c1.G, c2.B));
                    }
                    else
                    {
                        plain.SetPixel(i, j, Color.FromArgb(c2.R, c3.G, c1.B));
                    }
                    index++;
                }
            }
        }

        bool isArrPenuh(Point[] arrP, int angkaKosong)
        {
            for (int i = 0; i < arrP.Length; i++)
            {
                if ((arrP[i].X == angkaKosong) && (arrP[i].Y == angkaKosong))
                {
                    return false;
                }
            }
            return true;
        }

        Point[] pseudoRandomize(String key, int imgWidth, int imgHeight, int
jmlPixel)
        {
            Point[] hasil = new Point[jmlPixel];
            Point pTemp = new Point(0, 0), pTempTemp = new Point(-1, -1);
            int seed = 0;
            bool[,] matrix = new bool[imgWidth, imgHeight];
            for (int j = 0; j < imgHeight; j++)
            {
                for (int i = 0; i < imgWidth; i++)
```

```
                {
                    matrix[i, j] = false;
                }
            }
            for (int i = 0; i < key.Length; i++)
            {
                seed += (int)key[i];
            }

            for (int i = 0; i < hasil.Length; i++)
            {
                hasil[i] = new Point(-1, -1);
            }

            for (int offset = 0; offset < hasil.Length; offset++)
            {
                pTemp.X = ((seed * offset) % imgWidth);
                pTemp.Y = ((seed * (offset + 2)) % imgHeight);

                while (matrix[pTemp.X, pTemp.Y] && (!isArrPenuh(hasil, -1)))
                {
                    pTempTemp.X = pTemp.X;
                    pTempTemp.Y = pTemp.Y;
                    if (pTemp.X < imgWidth - 1)
                    {
                        pTemp.X++;
                    }
                    else
                    {
                        pTemp.X = 0;
                        if (pTemp.Y < imgHeight - 1)
                        {
                            pTemp.Y++;
                        }
                        else
                        {
                            pTemp.Y = 0;
                        }
                    }
                }

                    hasil[offset].X = pTemp.X;
                    hasil[offset].Y = pTemp.Y;
                    matrix[pTemp.X, pTemp.Y] = true;
            }
            return hasil;
        }
    }
}
```