Rancangan Proses Verifikasi Penggantian Kata Sandi Akun dengan Implementasi Kriptografi Visual

Suhailie 13517045[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13517045@std.stei.itb.ac.id

***Abstract*—Perkembangan teknologi yang semakin luas memunculkan berbagai kemudahan dalam kehidupan sehari-hari. Berbagai aplikasi atau *software* telah dibuat untuk memudahkan sebuah proses seperti transaksi, komunikasi, dan lainnya. Aplikasi - aplikasi tersebut biasanya akan meminta pengguna untuk membuat sebuah akun personal sehingga pengguna dapat mengakses aplikasi tersebut dengan informasi alamat *e-mail* dan kata sandi (*password*), sehingga data - data yang diperlukan dapat disimpan dan diatur oleh administrasi aplikasi. Apabila kata sandi tersebut dilupakan oleh pengguna, pengguna dapat meminta proses penggantian kata sandi. Namun, terdapat kelemahan pada proses tersebut yang menggunakan *email-verification*, *sms-verification*. Dan proses tersebut masih banyak digunakan untuk aplikasi - aplikasi lain. Dalam artikel ini, akan dijelaskan implementasi kriptografi visual untuk proses verifikasi penggantian kata sandi sebagai sebuah metode yang lebih aman.**

***Keywords*—kriptografi visual, verifikasi, penggantian kata sandi, implementasi**

I. Pendahuluan

Dalam era sekarang, teknologi telah berkembang dengan pesat. Dengan perkembangan teknologi tersebut, berbagai bidang kehidupan sehari - hari telah dimudahkan. Mulai dari komunikasi, manusia telah dapat berkomunikasi secara jarak jauh tanpa harus bertemu secara langsung. Dalam bidang transaksi, manusia telah dapat membayar secara langsung tanpa harus menggunakan tunai dan bertatap muka.

Berbagai aplikasi atau *software* telah dibuat untuk mempermudah berbagai bidang kehidupan sehari-hari seperti aplikasi *e-mail*, aplikasi *m-banking*, dan aplikasi lainnya. Aplikasi tersebut mengharuskan pengguna untuk membuat sebuah akun personal. Akun tersebut akan digunakan oleh pengguna sebagai sebuah akses ke fitur-fitur aplikasi yang disediakan.

Pembuatan akun memerlukan pengguna untuk memasukkan beberapa data-data lain. Data yang penting adalah alamat *e-mail* dan kata sandi (*password*). Kedua data ini akan digunakan juga oleh pengguna untuk dapat masuk ke dalam akun sebuah aplikasi. Sehingga data tersebut perlu diketahui dan diingat oleh pengguna.

Menurut sebuah survei [4], sebanyak 78% dari 500 lebih pengguna cenderung untuk melupakan kata sandi akun mereka. Hal ini disebabkan penggunaan aplikasi lain yang banyak, sehingga mereka memiliki lebih dari 1 jenis kata sandi untuk masing-masing aplikasi yang mereka gunakan. Hal ini dapat berakibat pengguna harus mengingat sekian banyak kata sandi. Namun, seringkali pengguna melupakan kata sandi mereka karena satu hal dan lainnya.

Oleh karena itu, mereka akan meminta penggantian kata sandi baru untuk dapat masuk kembali ke akun mereka. Proses penggantian kata sandi tersebut memerlukan verifikasi dari sistem aplikasi kepada pengguna. Verifikasi ini digunakan untuk mengkonfirmasi bahwa pengguna tersebut merupakan pemilik akun yang bersangkutan.

Proses verifikasi ini menggunakan metode seperti *e-mail verification* yang mengirimkan sebuah kode pada *e-mail* akun pengguna. Kode tersebut akan digunakan sebagai kunci verifikasi untuk dimasukkan pada proses penggantian kata sandi. Bila kode yang dimasukkan benar, maka akan dilanjutkan kepada tahap penggantian kata sandi dengan memasukkan kata sandi baru.

Namun, metode verifikasi di atas masih terdapat kelemahan-kelemahan seperti halnya: pengguna yang tidak masuk ke dalam akun *e-mail* tidak bisa mendapatkan kode verifikasi yang dikirimkan sistem. Atau apabila ada pengguna lain yang telah pernah masuk ke dalam akun tersebut, kode tersebut dapat digunakan pengguna lain untuk mengganti kata sandi tanpa sepengetahuan pemilik akun. Dengan adanya hal tersebut, seringkali terjadi kehilangan akun personal pengguna.

Dalam artikel ini, akan dijelaskan implementasi kriptografi visual untuk proses verifikasi penggantian kata sandi akun. Implementasi ini diharapkan dapat memperkuat keamanan dalam proses verifikasi tersebut.

II. Dasar Teori

A. Kriptografi

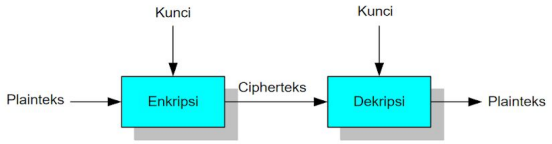
Kriptografi merupakan ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan, integritas data, dan otentikasi (Menez, 1996) [1]. Kriptografi juga dapat diartikan sebagai ilmu dan seni untuk menjaga keamanan pesan (Schneier, 1996) [1]. Ketiga aspek dapat dijelaskan secara berikut:

- Kerahasiaan (*confidentiality)*
  Aspek ini menjelaskan bahwa informasi dapat terjaga kerahasiaannya dari pihak ketiga (atau orang yang

tidak diinginkan).

- Integritas Data (*data integrity*)
  Aspek ini menjelaskan bahwa informasi dapat terjaga keasliannya, tanpa adanya perubahan dari isi informasi sebenarnya.

- Otentikasi (*authentication*)
  Aspek ini menjelaskan bahwa informasi yang dikirim merupakan pemilik informasi sebenarnya atau yang ditujukan.



Gambar II.1. Ilustrasi proses kriptografi [1]

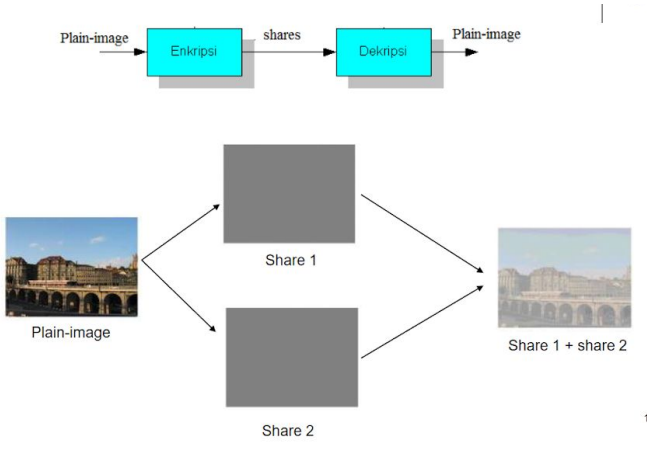
Dalam kriptografi, pengirim akan mengirimkan pesan atau informasi kepada penerima tanpa diketahui oleh pihak ketiga. Pengirim akan melakukan enkripsi, yaitu proses menyandikan isi pesan (disebut sebagai *plaintext*) ke dalam isi sandi (disebut sebagai *ciphertext*). Enkripsi tersebut dilakukan dengan sebuah kunci. Pengirim akan mengirimkan *ciphertext* dan sebuah kunci kepada penerima.

Untuk mengetahui isi pesan yang sebenarnya, penerima harus mengembalikan *ciphertext* tersebut ke dalam isi pesan yang sebenarnya menggunakan kunci yang diterima. Proses ini dinamakan dengan dekripsi.

Kriptografi sendiri memiliki banyak metode dan teknik yang telah ditemukan dan dikembangkan. Salah satu teknik kriptografi tersebut adalah kriptografi visual.

B. Kriptografi Visual

Kriptografi visual merupakan sebuah teknik kriptografi yang mengenkripsi informasi visual dengan sebuah cara sehingga dekripsi cukup dilakukan dengan mempersepsi informasi menggunakan indra penglihatan (mata) [2]. Teknik ini pertama kali diperkenalkan oleh Moni Naor dan Adi Shamir dalam makalah mereka yang berjudul "*Visual Cryptography*" pada tahun 1994.



Gambar II.2. Ilustrasi proses kriptografi visual [2]

Pada kriptografi visual, proses enkripsi dan dekripsi akan dilakukan melalui gambar. Gambar yang ingin dikirimkan (*plain image*) dibagi menjadi beberapa *share*. *Share* merupakan gambar hasil enkripsi, yang terlihat seperti gambar acak yang tidak bermakna. *Share* tersebut akan berperan sebagai *ciphertext* dan kunci dekripsi.

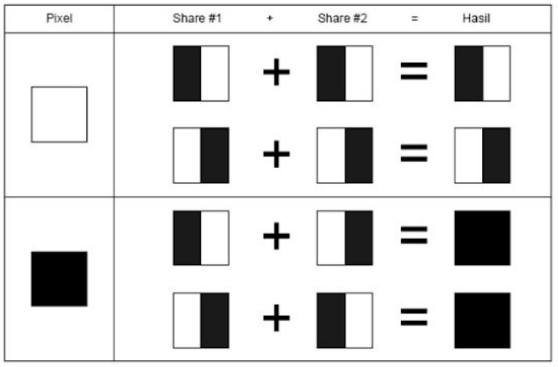
Untuk mendapatkan gambar seperti semula, *share* - *share* yang dimiliki tersebut ditumpuk menjadi menjadi satu. Hasil tumpukan tersebut akan menunjukkan isi gambar asli, walaupun memiliki kualitas yang tidak sama dengan gambar aslinya.

Kriptografi visual dapat diterapkan pada gambar yang memiliki jenis warna seperti biner (hitam-putih), abu-abu (*grayscale*), dan berwarna (*true color*).

C. Kriptografi Visual pada Gambar Biner

Setiap gambar terdiri dari sejumlah *pixel*, yang dapat diketahui dari ukuran gambarnya (1.200 x 1.500 berarti 1.800.000 *pixel*). *Pixel* merupakan unit satuan terkecil pada gambar yang memiliki satu warna tertentu. Setiap *pixel* memiliki ukuran *n*-bit, yang mana pada gambar biner setiap *pixel* berukuran 1-bit. *Pixel* pada gambar biner akan bernilai 1 jika berwarna hitam, dan bernilai 0 jika berwarna putih.

Dalam kriptografi visual pada gambar biner, setiap *pixel* akan dibagi menjadi *sub-pixel*. *Sub-pixel* dari setiap *share* yang ditumpuk, akan menghasilkan sebuah *pixel* yang dapat dipersepsi sebagai hitam atau putih.



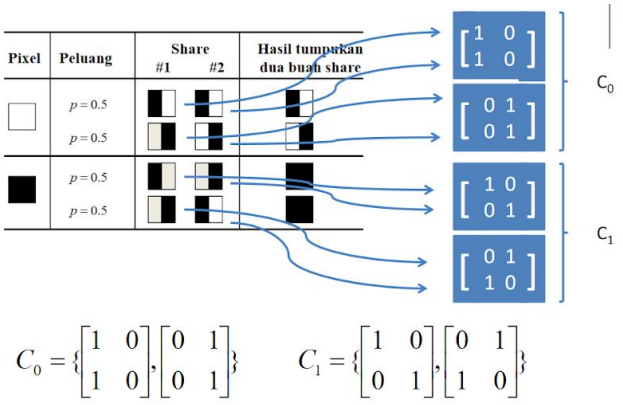
Gambar II.3. *Pixel* dan pembagiannya menjadi 2 *sub-pixel* [2]

Pada gambar di atas, dapat terlihat bahwa *pixel* hitam akan selalu tampak hitam sempurna pada hasil penumpukan, sedangkan *pixel* putih akan mengandung *noise*, namun masih dapat dipersepsi oleh manusia.

Langkah kerja kriptografi visual pada gambar biner adalah sebagai berikut (menggunakan skema $n = 2$, $m = 2$):

1. Setiap *pixel* akan muncul pada *n* buah *share*, dan tiap *share* terdiri atas *m* buah *sub-pixel* berwarna hitam dan putih. Kemudian dibuat sebuah matriks S berukuran *n* x *m*, di mana:
   - S[i,j] = 1 jika *sub-pixel* ke-j pada *share* ke-i berwarna hitam
   - S[i,j] = 0 jika *sub-pixel* ke-j pada *share* ke-i berwarna putih
2. Didefinisikan dua buah matriks yaitu C_0 dan C_1. C_0 merupakan semua matriks S yang merepresentasikan *pixel* putih. C_1 merupakan semua matriks S yang

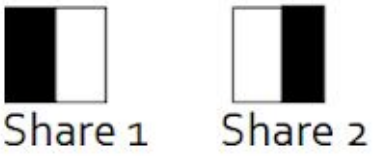
merepresentasikan *pixel* hitam.



$$C_0 = \left\{ \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \right\} \qquad C_1 = \left\{ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \right\}$$

Gambar II.4. Contoh pendefinisian matriks C_0 dan C_1 dalam skema $n = 2$, $m = 2$.

3. Ambil sebuah *pixel* pada *plain image* (dinamakan *pixel* P). Jika P berwarna hitam, ambil secara acak sebuah matriks S pada C_1, dan jika P berwarna putih, ambil secara acak sebuah matriks S pada C_0.
4. Misalkan P berwarna hitam dan matriks yang diambil dari C_1 adalah $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, maka hasil *sub-pixel* pada *share* 1 dan *share* 2 adalah berikut.



Gambar II.5. Hasil *sub-pixel* pada langkah ke-4

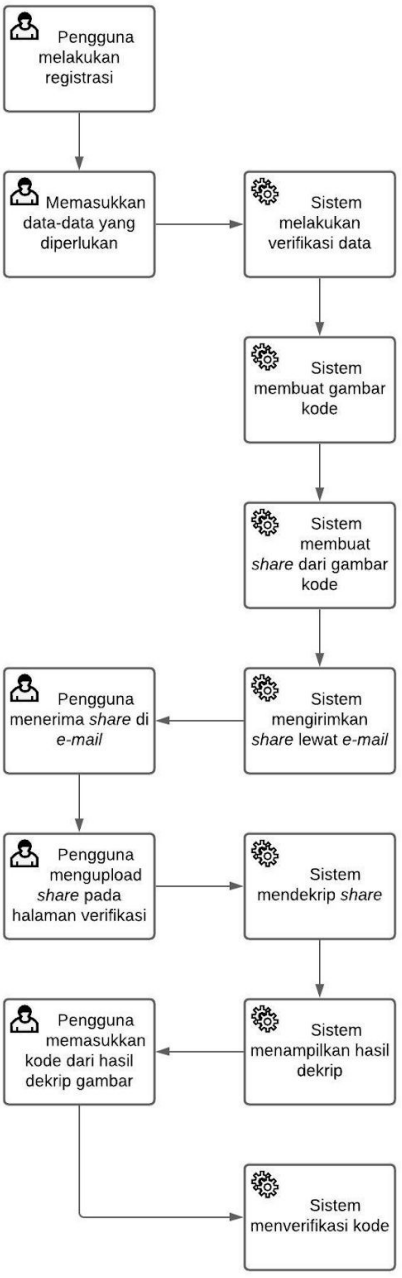
5. Ulangi langkah 3 dan 4 untuk *pixel* lainnya dari *plain image*.

III. Implementasi

A. Rancangan Struktur Sistem

Untuk implementasi kriptografi visual pada proses verifikasi penggantian kata sandi, akan dirancang sebuah struktur untuk proses keberjalanan sistem. Struktur tersebut akan dimulai dari proses registrasi, kemudian dilanjutkan dengan proses verifikasi penggantian kata sandi.

a. Proses Registrasi



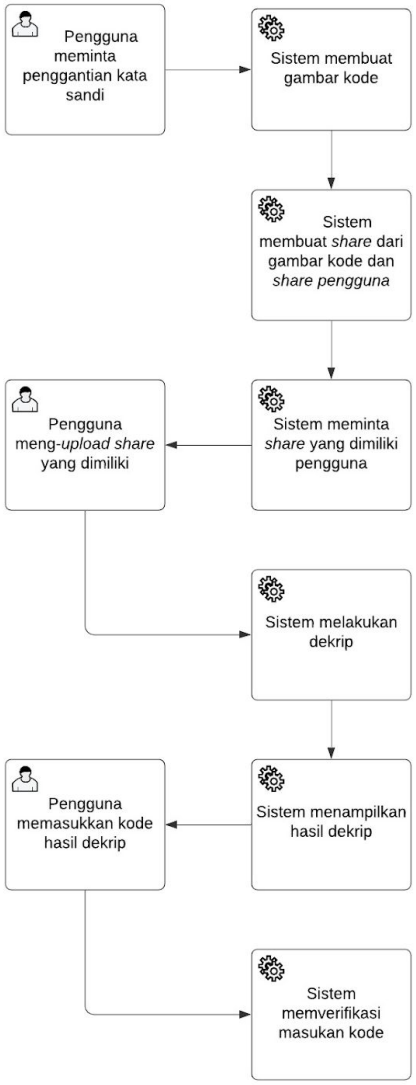
Gambar III.1. Desain proses registrasi

Pada tahap registrasi, pengguna akan melakukan registrasi dan memasukkan data-data yang diperlukan seperti nama, alamat *e-mail*, kata sandi, nomor telepon, dan lainnya. Sistem akan menerima data tersebut dan membuat sebuah gambar berisi kode. Kode tersebut akan digunakan sebagai kode

verifikasi *e-mail*. Kemudian sistem membuat *share* dari gambar kode menggunakan skema $n = 2$, $m = 2$. Sebuah *share* akan dikirim ke *e-mail* pengguna untuk proses verifikasi. *Share* ini merupakan *share* yang akan harus disimpan pengguna karena berfungsi sebagai kunci untuk pengguna.

Pengguna harus meng-*upload share* yang diterima ke halaman verifikasi sistem. Sistem akan melakukan dekrip (penumpukan gambar) dan menampilkan hasilnya. Bila *share* yang digunakan benar, maka akan tertampil kode verifikasi, dan pengguna memasukkan kode tersebut pada halaman verifikasi. Bila kode yang dimasukkan sesuai, maka proses verifikasi selesai dan sistem akan membuatkan akun.

b. Proses Verifikasi Penggantian Kata Sandi



Gambar III.2. Desain proses verifikasi penggantian kata sandi

Pada tahap verifikasi ini, pengguna akan meminta penggantian kata sandi akun. Sistem menerima permintaan tersebut dan membuat sebuah gambar yang berisi kode. Kode tersebut akan digunakan untuk kode verifikasi penggantian. Kemudian sistem akan membuat sebuah *share* dari gambar kode dan *share* yang dimiliki oleh pengguna.

Sistem akan meminta *share* yang dimiliki oleh pengguna, dan pengguna harus meng-*upload share* yang diterima saat proses registrasi. Sistem akan melakukan dekrip (penumpukan gambar) dan menampilkan hasil tersebut. Bila *share* yang diterima benar, maka akan muncul tampilan gambar kode. Pengguna memasukkan kode tersebut, dan sistem akan memverifikasi kode. Bila sesuai, maka pengguna akan dipersilahkan untuk mengganti kata sandi akun.

B. Implementasi Kriptografi Visual

Implementasi kriptografi visual menggunakan bahasa pemrograman Python. Kode implementasi dapat diakses dari link: https://github.com/suhailiealx/uas_kripto.

Pada implementasi, *pixel* berwarna hitam pada program bernilai 0 dan sebaliknya *pixel* berwarna putih bernilai 1.

- Pembuatan *share* dari file asli

```
from PIL import Image, ImageDraw
import os
import sys
from random import SystemRandom
random = SystemRandom()
xrange = range

if len(sys.argv) != 2:
        print("This takes one argument; the image to be
split.")
        exit()
infile = str(sys.argv[1])

if not os.path.isfile(infile):
        print("That file does not exist.")
        exit()

img = Image.open(infile)

f, e = os.path.splitext(infile)
out_filename_A = f+"_share1.png"
out_filename_B = f+"_share2.png"

img = img.convert('1')  # convert image to 1 bit

print("Image size: {}".format(img.size))

width = img.size[0]*2
height = img.size[1]*2
print("{} x {}".format(width, height))
out_image_A = Image.new('1', (width, height))
out_image_B = Image.new('1', (width, height))
draw_A = ImageDraw.Draw(out_image_A)
draw_B = ImageDraw.Draw(out_image_B)

C0 = ((1, 0, 1, 0), (0, 1, 0, 1))
C1 = ((1, 0, 0, 1), (0, 1, 1, 0))

# Cycle through pixels
for x in xrange(0, int(width/2)):
        for y in xrange(0, int(height/2)):
        pixel = img.getpixel((x, y))
        if pixel == 0: #Pixel was black
        pat1 = random.choice(C1)
        draw_A.point((x*2, y*2), pat1[0])
        draw_A.point((x*2+1, y*2), pat1[1])
        draw_A.point((x*2, y*2+1), pat1[0])
        draw_A.point((x*2+1, y*2+1), pat1[1])

        pat2 = C1[1 - C1.index(pat1)]
```

```
            draw_B.point((x*2, y*2), pat2[2])
            draw_B.point((x*2+1, y*2), pat2[3])
            draw_B.point((x*2, y*2+1), pat2[2])
            draw_B.point((x*2+1, y*2+1), pat2[3])

            else : #Pixel was white
            pat1 = random.choice(C0)
            draw_A.point((x*2, y*2), pat1[0])
            draw_A.point((x*2+1, y*2), pat1[1])
            draw_A.point((x*2, y*2+1), pat1[0])
            draw_A.point((x*2+1, y*2+1), pat1[1])

            pat2 = C0[1 - C0.index(pat1)]
            draw_B.point((x*2, y*2), pat2[2])
            draw_B.point((x*2+1, y*2), pat2[3])
            draw_B.point((x*2, y*2+1), pat2[2])
            draw_B.point((x*2+1, y*2+1), pat2[3])

out_image_A.save(out_filename_A, 'PNG')
out_image_B.save(out_filename_B, 'PNG')
print("Done.")
```

Algoritma III.1. Pembuatan *share* dari file asli

Program akan menerima sebuah file gambar yang akan dilakukan enkripsi. Pertama-tama, akan dibuat dua data file gambar kosong dengan ukuran 2 kali dari file gambar input. Kemudian buat matriks C_0 dan C_1 sesuai dengan skema yang digunakan (2,2). Melalui iterasi setiap *pixel* dari file gambar input, akan diambil matriks (*tuple*) S secara acak dari matriks C_0 (bila *pixel* berwarna putih) atau C_1 (bila *pixel* berwarna hitam).

Matriks S tersebut digunakan untuk menentukan nilai pixel dari *share* 1 dan *share* 2. Setelah iterasi selesai, maka data file gambar kedua *share* akan disimpan dalam bentuk gambar berformat PNG.

- Penggabungan *share*

```
from PIL import Image, ImageDraw
import os
import sys
from random import SystemRandom
random = SystemRandom()
xrange = range

if len(sys.argv) != 3:
            print("This takes two argument; the image to be
split.")
            exit()
infile1 = str(sys.argv[1])
infile2 = str(sys.argv[2])

if not os.path.isfile(infile1):
            print("That file 1 does not exist.")
            exit()
if not os.path.isfile(infile2):
            print("That file 2 does not exist.")
            exit()


img1 = Image.open(infile1) #share1
img2 = Image.open(infile2) #share2

f, e = os.path.splitext("outfile")
out_filename_decrypt = f+"_decrypt.png"

img1 = img1.convert('1')  # convert image to 1 bit
img2 = img2.convert('1')  # convert image to 1 bit

width = img1.size[0]
height = img1.size[1]

out_image_decrypt = Image.new('1', (width, height))
draw_decrypt = ImageDraw.Draw(out_image_decrypt)

#Cycle through pixels
for x in xrange(0, int(width)):
            for y in xrange(0, int(height)):
            pixel1 = img1.getpixel((x, y))
            pixel2 = img2.getpixel((x, y))

            draw_decrypt.point((x,y), (pixel1 or pixel2)) #using
OR operator

out_image_decrypt.save(out_filename_decrypt, 'PNG')
print("Done.")
```

Algoritma III.2. Penggabungan *share*

Program akan menerima dua buah file gambar *share* untuk dilakukan dekripsi. Pertama-tama, akan dibuat sebuah file gambar kosong dengan ukuran yang sama dengan ukuran input gambar. File ini digunakan untuk menyimpan hasil dekripsi. Dengan melalui iterasi setiap *pixel* dari kedua gambar *share*, nilai *pixel* tersebut akan dilakukan operasi OR sebagai nilai *pixel* untuk gambar hasil dekrip.

Setelah selesai iterasi, data file hasil dekrip akan disimpan dalam bentuk gambar berformat PNG.

- Pembuatan *share* dari file asli dan *share* lain

```
from PIL import Image, ImageDraw
import os
import sys
from random import SystemRandom
random = SystemRandom()
xrange = range

if len(sys.argv) != 3:
            print("This takes two argument; the image to be
split.")
            exit()
infile1 = str(sys.argv[1])
infile2 = str(sys.argv[2])

if not os.path.isfile(infile1):
            print("That file 1 does not exist.")
            exit()
if not os.path.isfile(infile2):
            print("That file 2 does not exist.")
            exit()


img1 = Image.open(infile1) #original file
img2 = Image.open(infile2) #share user

f, e = os.path.splitext("outfile")
out_filename_system = f+"_shareSystem.png"

img1 = img1.convert('1')  # convert image to 1 bit
img2 = img2.convert('1')  # convert image to 1 bit

width = img1.size[0]*2
height = img1.size[1]*2

out_image_system = Image.new('1', (width, height))
```

```
draw_system = ImageDraw.Draw(out_image_system)

C0 = ((1, 0, 1, 0), (0, 1, 0, 1))
C1 = ((1, 0, 0, 1), (0, 1, 1, 0))

# Cycle through pixels
for x in xrange(0, int(width/2)):
         for y in xrange(0, int(height/2)):
         pixel1 = img1.getpixel((x, y))
         if pixel1 == 0: #Pixel was black
         draw_system.point((x*2, y*2),
not(img2.getpixel((x*2,y*2))))
         draw_system.point((x*2+1, y*2),
not(img2.getpixel((x*2+1,y*2))))
         draw_system.point((x*2, y*2+1),
not(img2.getpixel((x*2,y*2+1))))
         draw_system.point((x*2+1, y*2+1),
not(img2.getpixel((x*2+1,y*2+1))))

         else : #Pixel was white
         draw_system.point((x*2, y*2),
img2.getpixel((x*2,y*2)))
         draw_system.point((x*2+1, y*2),
img2.getpixel((x*2+1,y*2)))
         draw_system.point((x*2, y*2+1),
img2.getpixel((x*2,y*2+1)))
         draw_system.point((x*2+1, y*2+1),
img2.getpixel((x*2+1,y*2+1)))


out_image_system.save(out_filename_system, 'PNG')
print("Done.")
```

Algoritma III.3. Pembuatan *share* dari file asli dan *share* lain

Program ini memiliki alur yang sama dengan program pembuatan *share* dari hanya file asli. Program menerima dua input gambar yaitu file asli beserta file *share* lain. Akan dibuat sebuah data file kosong untuk menyimpan hasil gambar *share.*

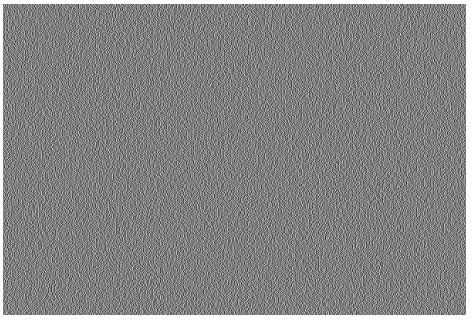
Melalui iterasi setiap *pixel* pada file asli, nilai *pixel* hasil gambar *share* akan ditentukan dari nilai operasi XOR pada nilai *pixel* file asli dan nilai *pixel share* lain. Setelah selesai, maka data file hasil gambar *share* akan disimpan dalam berformat PNG.

IV. Pengujian dan Analisis

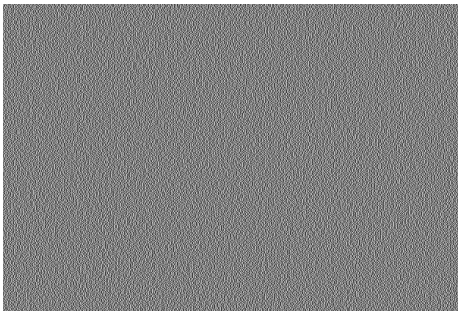
Dengan hasil pengujian pada implementasi yang dibuat. didapatkan hasil:

- Proses pembuatan *share* dari gambar asli

Gambar IV.1. Gambar Kode 1



Gambar IV.2 Hasil *share* 1



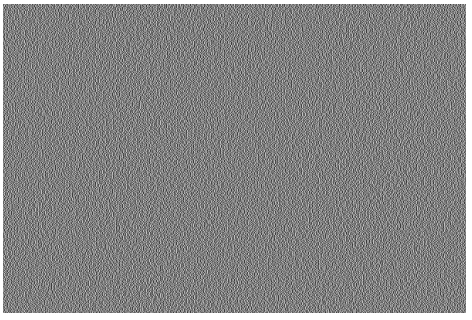
Gambar IV.3 Hasil *share* 2

- Proses penggabungan *share*

Gambar IV.4 Hasil penggabungan *share*

Implementasi algoritma berhasil menggabungkan kedua *share* dari gambar IV.2 dan gambar IV.3. Hasil gambar tersebut menunjukkan bahwa isi hasil sama dengan isi dari gambar file asli.

- Proses pembuatan *share* dari gambar asli dan *share* lain



Gambar IV.5 Hasil *share* dari file asli gambar IV.1 dan *share* dari gambar IV.2

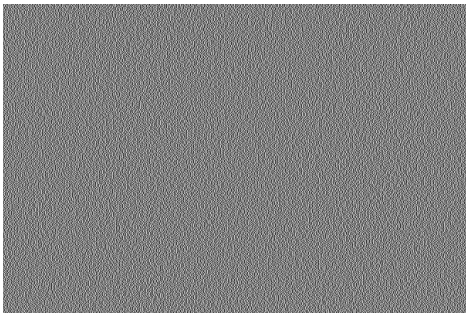
Gambar IV.6 Hasil penggabungan *share* dari gambar IV.5 dengan *share* dari gambar IV.2

Dari hasil pengujian, didapatkan bahwa implementasi algoritma tersebut berhasil membuat sebuah *share* berdasarkan sebuah file asli dengan *share* lain. Hasil penggabungan kedua *share* juga menunjukkan gambar dengan isi yang sama dengan file asli.

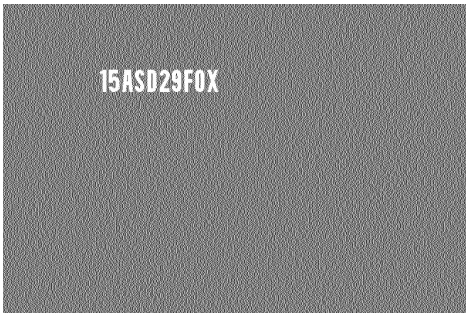
Selain itu, juga dilakukan pengujian terhadap pembuatan *share* berdasarkan file asli lain dan *share* lain yang sama sebelumnya.

15ASD29FOX

Gambar IV.7 Gambar Kode 2



Gambar IV.8 Hasil *share* dari file asli gambar IV.7 dan *share* dari gambar IV.2



Gambar IV.9 Hasil penggabungan *share* dari gambar IV.8 dengan *share* dari gambar IV.2

Dari hasil pengujian, didapatkan bahwa hasil penggabungan kedua *share* tetap dapat memberikan isi gambar yang sama dengan isi file asli.

Demikian, maka gambar *share* IV.2 dapat dijadikan sebagai file *share* yang disimpan pengguna pada saat proses registrasi akun. Pada saat proses verifikasi penggantian kata sandi akun, sistem akan membuat sebuah kode gambar dan membuat *share* berdasarkan kode gambar dan *share* pengguna yang tersimpan dalam *database*.

Contoh hasil pembuatan *share* tersebut adalah gambar IV.5 dan gambar IV.8. Kemudian sistem akan menerima input gambar *share* dari pengguna untuk digabungkan kembali dengan hasil *share* tadi. Bila *share* yang diterima benar, maka akan dihasilkan isi gambar file asli seperti pada gambar IV.6 dan gambar IV.9.

V. Kesimpulan dan Saran

Penggunaan kriptografi visual pada proses verifikasi penggantian kata sandi akun dapat dinyatakan lebih aman dikarenakan pemilik akun memiliki sebuah file *share* yang jarang diketahui atau dipahami oleh banyak orang. Proses implementasi juga membuktikan kebenaran isi gambar asli dapat dihasilkan secara tepat. Dengan begitu, kebenaran verifikasi yang dilakukan dapat lebih terjaga.

Walaupun demikian, masih terdapat beberapa pengembangan yang dapat dilakukan seperti pengimplementasian melalui gambar berwarna (*true color*).

Ucapan Terima Kasih

Puji syukur ke hadirat Tuhan Yang Maha Esa, oleh karena berkat-Nya penulis dapat menyelesaikan makalah ini dengan tepat waktu. Penulis menyampaikan terima kasih kepada dosen pengampu mata kuliah IF4020 Kriptografi, bapak Dr. Ir. Rinaldi Munir, M.T., atas wawasan dan ilmu pengetahuan mengenai pembelajaran kriptografi yang diberikan oleh beliau sehingga makalah ini dapat dibuat. Selain itu, penulis juga berterima kasih kepada semua pihak yang telah turut membantu dalam proses pembuatan makalah ini. Dengan makalah ini, penulis berharap dapat membantu pengembangan wawasan dan pengetahuan pembaca.

Referensi

[1] Munir, R. 2020. "Pengantar Kriptografi (2020)". Slide terdapat pada link: http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2020-2021/Pengantar-Kriptografi-(2020).pdf.

[2] Munir, R. 2020. "Kriptografi Visual, Teori dan Aplikasinya (Bag. 1)". Slide terdapat pada link: http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2020-2021/Kriptografi-Visual-Bagian1.pdf.

[3] Munir, R. 2020. "Kriptografi Visual, Teori dan Aplikasinya (Bag. 2)". Slide terdapat pada link: http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2020-2021/Kriptografi-Visual-Bagian2.pdf.

[4] "Study: 3 in 4 Users Required a Reset of a Forgotten Password in the Last 90 Days". https://securityintelligence.com/news/study-3-in-4-users-required-a-reset-of-a-forgotten-password-in-the-last-90-days/. Diakses pada tanggal 21 Desember 2020.

[5] https://gist.github.com/deibit/ccc2b55ae9eab94392e4118c05aded52.
Diakses pada tanggal 21 Desember 2020

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, 21 Desember 2020

Suhailie - 13517045