SVD Implementation Analysis on Image Cryptography

Brian Albar Hadian, 13523048[.2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*13523048@std.stei.itb.ac.id*, [2]*bri.hadian@gmail.com*

***Abstract*— Informasi merupakan hal yang penting pada masa modern ini sehingga diperlukan suatu cara untuk melakukan perlindungan terhadap informasi tersebut. Salah satu informasi yang dapat dilindungi adalah gambar dengan melakukan metode kriptografi yang memanfaatkan metode SVD (Singular Value Decomposition) dengan modifikasi tertentu. Hasil dari metode ini adalah gambar dengan dimensi yang sama, tetapi mengalami acak sedemikian sehingga gambar tersebut tidak memiliki informasi yang berguna apabila tidak melakukan dekripsi terlebih dahulu. Metode ini kemudian memiliki kelebihan berupa proses yang cukup cepat dan tingkat kemiripan dengan gambar semula yang relatif rendah, tetapi disertai dengan kekurangan bahwa gambar yang dapat dienkripsi merupakan gambar dengan dimensi persegi.**

***Keywords*—SVD, Gambar, Informasi, Enkripsi**

I. Latar Belakang

Semenjak revolusi industri, telah terjadi berbagai perkembangan dalam teknologi dan kesejahteraan manusia. Hal ini disebabkan oleh meningkatnya kualitas informasi yang tersebar antar negara dan Kerjasama yang dilakukan seiring dengan berjalannya waktu. Hal ini kemudian mengakibatkan peningkatan taraf hidup melalui penerapan teknologi, kebijakan hingga studi yang dilakukan oleh berbagai negara untuk kepentingan manusia. Dengan begitu, tak dapat dipungkiri bahwa informasi merupakan hal yang penting demi perkembangan kesejahteraan manusia dan pemerataan standar hidup bersama.

Perkembangan teknologi selama ini umumya berdiri dari suatu kebutuhan manusia dan akhirnya diterapkan dalam bentuk implementasi dari ilmu-ilmu yang ada. Selanjutnya, dengan meningkatnya standar hidup manusia, tenaga manusia tidak lagi difokuskan untuk melakukan hal yang monoton, melainkan difokuskan pada kegiatan yang membutuhkan daya pikir dan kreativitas yang tinggi. Dengan adanya kebutuhan tersebut, dikembangkan teknologi yang dapat mengolah informasi sebagai dasar dalam menyelesaikan suatu pekerjaan tertentu dalam bentuk komputer. Peralihan kerja manusia yang sebelumnya berfokus pada kebutuhan bertahan hidup menjadi kebutuhan manusia untuk mempermudah hidup merupakan konsekuensi dari majunya peradaban.

Sebagaimana yang telah disebutkan, kebutuhan manusia saat ini lebih berfokus pada hal yang membutuhkan daya pikir tinggi dan dengan berkembangnya teknologi, manusia akhirnya dapat melakukan hal tersebut, tetapi, hal ini juga mengimplikasikan bahwa kebutuhan terkait informasi dan teknologi semakin tinggi. Informasi kemudian menjadi bernilai karena sifatnya yang dapat mengembangkan suatu Masyarakat apabila disertakan dengan penerapan teknologi yang tepat. Namun, tak jarang juga bahwa informasi disalahgunakan sebagai alat untuk memanipulasi dan mengancam berbagai pihak. Hal ini dapat dilihat dari maraknya kasus kejahatan siber, *phising*, serangan malware dan ransomware, serta banyak lagi. Dengan itu, dibutuhkan suatu cara untuk melindungi dan menyebarkan informasi sedemikian sehingga tidak dapat disalahgunakan. Bidang ilmu yang mempelajari hal-hal terkait perlindungan dan penyebaran informasi dengan aman disebut sebagai ilmu kriptografi.

Kriptografi merupakan sebuah ilmu yang membahas teknik matematis yang berkaitan dengan topik keamanan informasi(Munir, 2019). Kriptografi digunakan untuk menyebarkan informasi antar pihak sehingga orang lain tidak dapat mengerti dan/atau mendapatkan informasi yang dihendaki. Ilmu ini sudah ada semenjak masa mesir kuno (± 2000 SM) yang digunakan sebagai alat komunikasi selama perang dan kondisi darurat. Selain itu, kriptografi juga digunakan semenjak zaman romawi kuno, yakni pada saat kekaisaran Julius Caesar (100 – 44 SM) yang mengembangkan salah satu metode kriptografi yang paling dikenal saat ini, yakni Caesar Cipher. Selain itu juga, kriptografi sempat digunakan pada masa Revolusi Perancis oleh Jean-Baptiste de Saint-Just untuk memerintahkan rencana pembunuhan terhadap salah satu keluarga bangsawan Perancis, tetapi pesan tersebut berhasil dipecahkan oleh salah satu anggota monarki dan berakhir tragis bagi Jean-Baptiste. Dengan berbagai penggunaan dan sejarah yang telah disebutkan, dapat dinyatakan bahwa ilmu kriptografi merupakan ilmu yang bermanfaat dan praktis digunakan demi keamanan penyebaran informasi.

Kriptografi memiliki dasar pada pemanfaatan bidang matematika, teori bilangan, matriks, dan aljabar untuk menghasilkan suatu algoritma yang dapat menyandikan suatu pesan menjadi bentuk pesan yang lain. Beberapa

contoh dari metode kriptografi diantaranya *Symmetric Key Cryptography, Asymmetric Key Cryptography, Hashing, Secret Sharing, Digital Signatures, Elliptic Curve Cryptography, Quantum Cryptography,* Steganografi, dll. Berbagai metode yang telah disebutkan memiliki dasar pada bidang matematika dan biasa digunakan oleh perusahaan untuk menyandikan informasi-informasi sensitif.

1.1 Singular Value Decomposition

Singular Value Decomposition (SVD) merupakan suatu metode untuk melakukan dekomposisi terhadap suatu matriks menjadi tiga matriks terpisah. Metode ini biasanya digunakan untuk melakukan penyisipan data terhadap informasi yang terdapat pada matriks tersebut. Metode SVD dapat dinyatakan sebagai berikut.

Untuk setiap matriks $A \in R^{m \times n}$, terdapat suatu kombinasi U, S, V sedemikian sehingga

$$A = USV^T$$

Dengan U adalah suatu matriks ortonormal m x n, S adalah suatu matriks diagonal n x n dengan elemen-elemen pada diagonal utamanya bernilai tidak negative, dan V^T adalah matriks ortonormal n x n. Nilai diagonal pada S merupakan nilai yang berurut mengecil dan merupakan akar kuadrat dari nilai eigen pada matriks A^TA.

Berdasarkan penerapan di dunia nyata, informasi, umumnya pada komputer, dapat dinyatakan dalam bentuk matriks dengan ketentuan tertentu. Dalam hal ini, salah satu contoh dari penerapan ini adalah penggunaan nilai bilangan bulat dalam bentuk matriks pada gambar dengan format RASTER. Dengan demikian, informasi pada gambar tersebut dapat diolah kembali sehingga dapat dienkripsi sesuai kebutuhan tertentu.

II. METODE

<tata cara melakukan enkripsi, sesuain sama yang dari sumber + sisipin pesan ke gambarnya>

<Hal yang bisa dijadiin parameter keberhasilan : besar data hasil dan waktu proses, tingkat kemiripan dengan gambar awal>

Metode enkripsi yang digunakan didasarkan pada penerapan SVD pada matriks yang berisi informasi yang hendak dienkripsi. Dalam hal ini, informasi yang dienkripsi merupakan gambar dengan nilai piksel (R, G, B) yang dienkripsi sedemikian sehingga akan disatukan kembali pada akhir tahap enkripsi

Tahap yang disebutkan dibawah akan diterapkan pada setiap warna dasar yang dimiliki oleh piksel sehingga pada akhir metode enkripsi, diperoleh tiga matriks informasi yang berbeda

Tahap 1: Penentuan nilai kunci

Untuk memulai tahap enkripsi, dilakukan pemilihan acak terhadap kunci yang akan digunakan dengan ketentuan sebagai berikut.

$[7 < key1 < 10]$

$[0.9 < key2 < 3$ dengan $key2 \neq 1.5]$

$[0 < key3 < 3]$

Tahap 2 : Pengacakan nilai pada setiap warna dasar piksel

Nilai warna dasar pada setiap piksel akan diacak dengan menggunakan key1 berdasarkan urutan berikut

$$A1 = key1*\max(A) - A,$$
$$A2 = key1*\max(A1) - A1$$

Selanjutnya, akan dilakukan pengacakan kembali pada A dan A1 untuk memperoleh nilai ekstrem dari setiap matriks. Proses ini akan menghasilkan dua matriks dengan sifat memiliki nilai ekstrem pada A dan A1

$$B1 = A1 - A2,$$
$$B2 = key2*B1 + A2$$

Tahap 3 : Penerapan metode SVD untuk matriks B1 dan B2

Selanjutnya, dilakukan penerapan SVD untuk B1 dan B2 sedemikian sehingga diperoleh enam matriks seperti dibawah

$$[U_{B1}, S_{B1}, V_{B1}] = SVD\ (B1)$$
$$[U_{B2}, S_{B2}, V_{B2}] = SVD\ (B2)$$

Tahap 4 : Penyusunan ulang matriks

Berdasarkan hasil pada tahap 3, dilakukan pengacakan pada susunan matriks yang dimiliki oleh B1 dan B2 pada matriks S. Namun, pengacakan pada matriks selain S dijuga diperbolehkan dengan syarat hanya dilakukan sekali. Pada tahap ini, akan dilakukan penukaran terhadap matirks S pada B1 dan B2. Proses ini dapat dinyatakan sebagai berikut

$$C1 = U_{B1} * S_{B2} * V_{B1}^T$$
$$C2 = U_{B2} * S_{B1} * V_{B2}^T$$

Tahap 5 : Iterasi ulang

Untuk memperoleh tingkat kriptografi yang baik, dapat dilakukan iterasi dari tahap 2 hingga tahap 4 untuk memperoleh hasil matriks yang baru. Hal ini perlu diperhatikan selanjutnya karena jumlah perulangan dapat mempengaruhi metode dekripsi yang digunakan.

Berikut merupakan contoh penerapan iterasi ulang dari C1 dan C2, menjadi D1 dan D2 sebagai berikut

$$D1 = C1 - C2,$$
$$D2 = key3*D1 + C2$$

Kemudian, dilakukan tahap 3, yakni dekomposisi menjadi matriks SVD dan melakukan penukaran pada matriks S sebagaimana dinyatakan di bawah ini.

$$[U_{D1}, S_{D1}, V_{D1}] = SVD\ (D1)$$
$$[U_{D2}, S_{D2}, V_{D2}] = SVD\ (D2)$$

Selanjutnya, dilakukan penukaran pada matriks S pada dekomposisi matriks D1 dan D2,

$$E1 = U_{D1} * S_{D2} * V_{D1}^T$$

$$E2 = U_{D2} * S_{D1} * V_{D2}^T$$

Tahap 6 : Enkripsi dengan kunci akhir

Selanjutnya, dilakukan pemilihan kunci kembali, misal key4 dengan ketentuan batas seperti bawah ini,

[0 < key4 < 3]

Kemudian, lakukan operasi seperti dibawah ini untuk mendapatkan matriks akhir, F,

$$F = E1 + key4 * E2$$

Tahap 7 : Normalisasi matriks

Kemudian, dengan diperoleh matriks F dengan dimensi yang sama seperti gambar yang diperoleh sebelumnya, dilakukan normalisasi terhadap matriks F untuk menghindari nilai yang berada diluar batas ataupun tidak valid.

Operasi tersebut dinyatakan sebagai berikut.

$$FF = (F - MI) * 256 / (MA - MI)$$

Dengan MA adalah nilai maksimum yang ada pada matriks F dan MI adalah nilai minimum yang ada pada matriks F.

Tahap 8 : Pengacakan properti nilai ekstrem

Berdasarkan nilai maksimum dan minimum yang telah diperoleh, nilai tersebut akan diuraikan menjadi empat bilangan berdasarkan properti dari nilai ekstrem tersebut, yakni,

(1) Nilai tanda = 0 jika positif dan 0.1 jika negative
(2) Nilai bilangan bulat = int(int(abs(MI)) / 255)
(3) Nilai sisa = int(abs(MI)) mod 255
(4) Nilai pecahan = abs(MI) - int(abs(MI))

Hal ini dilakukan juga pada nilai ekstrem maksimum dan kemudian disertakan pada matriks informasi tersebut dengan empat bilangan dari nilai ekstrem minimum disertakan pada empat slot pertama dari matriks FF, kemudian empat slot terakhir dari matriks FF untuk empat bilangan dari nilai ekstrem maksimum.

Tahap 9 : Enkripsi pesan

Selanjutnya, lakukan enkripsi pesan yang hendak disampaikan dan lakukan enkripsi pesan tersebut melalui algoritma SHA-256 untuk memperoleh 256 karakter. Selanjutnya, disusun suatu matriks M dengan ukuran n x n dengan n adalah banyak kolom dari matriks FF.

Kemudian, sertakan setiap karakter dari hasil enkripsi pesan secara berurut dari slot pertama hingga jumlah karakter pesan terenkripsi sudah habis. Selanjutnya, apabila masih terdapat slot kosong pada matriks tersebut, maka secara berurut akan diisi dengan nilai perkalian sebagai berikut

$$a_i = (a_{i-1} * a_{i-2}) \bmod 256$$

apabila terdapat nilai $a_i = 0$, maka dilakukan pengacakan nilai acak untuk a_i dengan batas [0, 255].

Tahap 10 : Perkalian matriks pesan dan gambar

Terakhir, dilakukan perkalian matriks pesan dan gambar untuk memperoleh hasil dan kemudian dilakukan penyusunan ulang terlebih dahulu terhadap matriks gambar dengan menggabungkan setiap warna untuk membentuk matriks baru sebagai matriks akhir yang dapat dikirimkan.

Operasi tersebut dapat dinyatakan sebagai berikut.

$$I = FF * M$$

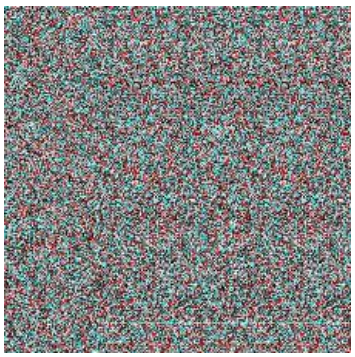
Terakhir, dengan disusunnya metode yang telah disebutkan, disusun pula suatu metode untuk menjadi perbandingan proses enkripsi pada situasi awal dengan situasi akhir. Beberapa hal yang akan menjadi parameter penilaian adalah:

1. Kecepatan proses enkripsi
   Semakin cepat proses enkripsi, semakin baik algoritma yang telah disusun.
2. Kemiripan dengan gambar sebelumnya
   Semakin berbeda dengan gambar sebelumnya, maka semakin baik algoritma enkripsi. Hal ini akan diukur menggunakan *cosine similarity*.

III. HASIL DAN PEMBAHASAN

A. Data dan Hasil Eksperimen

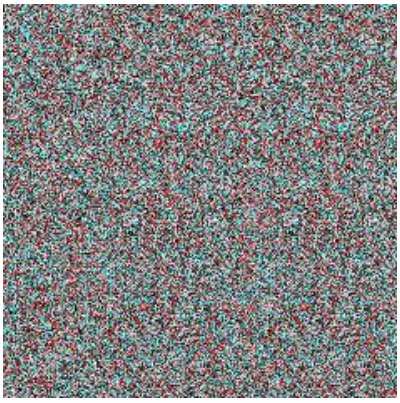
Gambar 1 (sebelum enkripsi)



Gambar 1 (Setelah enkripsi)

Dimensi = 120 x 120
Waktu yang dibutuhkan : 0.3752622604370117
Tingkat kemiripan : 0.7723434375537815

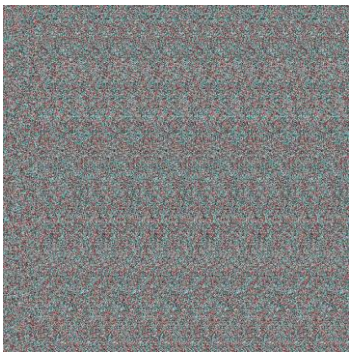
Gambar 2 (Sebelum enkripsi)



Gambar 2 (Setelah enkripsi)

Dimensi : 236 x 236
Waktu : 0.41884684562683105
Kemiripan : 0.7907598296075689

Gambar 3 (Sebelum enkripsi)



Gambar 3 (Setelah enkripsi)

Dimensi = 640 x 640
Waktu = 3.3204774856567383
Kemiripan = -0.16973081643855978

Gambar 4 (Sebelum enkripsi)

Gambar 4 (Setelah enkripsi)

Dimensi = 148 x 148
Waktu = 0.18908023834228516
Kemiripan = 0.7203933909422588

B. Kode Eksperimen

```
from PIL import Image
from typing import List
import random
import numpy as np
import hashlib
import time

# drawbacks of this method: can only be applied to
images of dimensions minimum m x 16, where m is real
numbers
'''
main algorithm
1. follow the algorithm from the pdf until the part where
there is svd
2. for the rightmost matrix, multiply it by a matrix formed
from the encryptions of the message
3. mesage matrix is comprised of n x n, and the sha value
is sorted from the first until the last by column. when
there's more room, fill it with x[i - 1] * x[i - 2] mod x[i -
3] until the matrix is fully formed
4. with the last two matrix have been multiplied, the final
svd-format matrix is the final encrypted image
'''


def cosine_similarity(vec1, vec2):
    # Compute dot product
```

```
    dot_product = np.dot(vec1, vec2)

    # Compute magnitudes
    norm_vec1 = np.linalg.norm(vec1)
    norm_vec2 = np.linalg.norm(vec2)

    # Compute cosine similarity
    similarity = dot_product / (norm_vec1 * norm_vec2)

    return similarity

def hexToDecimals(hashHex : str) -> List[int]:
    # kamus lokal
    decimal = []

    hexDict = {
        '0': 0, '1': 1, '2': 2, '3': 3,
        '4': 4, '5': 5, '6': 6, '7': 7,
        '8': 8, '9': 9, 'a': 10, 'b': 11,
        'c': 12, 'd': 13, 'e': 14, 'f': 15
    }

    # algorithm
    for i in range(len(hashHex)) :
        decimal.append(hexDict[hashHex[i]])

    return decimal

def encryptMessage(imgMatrix : np.array, msg : str):

    # algoritma

    # encrypt initial message
    hashObj = hashlib.sha256(msg.encode())
    hashHex = hashObj.hexdigest()

    # create matrix to be multiplied by the image matrix
    convertedHex = hexToDecimals(hashHex)

    # copy converted hex into msgMatrix
    msgMatrix =
np.array(copyEncryptMatrix(convertedHex, imgMatrix))

    # multiply the msg matrix with the img matrix
    resMatrix = imgMatrix @ msgMatrix
    resMatrix = resMatrix % 256

    return resMatrix

def copyEncryptMatrix(convertedHex : List[int],
imgMatrix : List[List[int]]):

    msgArray = []
    msgMatrix = []

    for i in range(len(imgMatrix)):
        for j in range(len(imgMatrix[i])):
            if ((i * len(imgMatrix[i]) + j)) >=
len(convertedHex):
                appendVal= (msgArray[len(msgArray) - 1] *
msgArray[len(msgArray) - 2]) % 256
                while (appendVal == 0):
                    appendVal = np.random.randint(0, 256)
                msgArray.append(appendVal)
            else :
                msgArray.append(convertedHex[(i *
len(imgMatrix[i]) + j)])

    # partition after certain length
    tempArray = []
    for i in range(len(msgArray)):
        tempArray.append(msgArray[i])
        if ((len(tempArray)) % len(imgMatrix[0]) == 0):
            msgMatrix.append(tempArray)
            tempArray = []

    return np.array(msgMatrix)

def encryptImage(colMatrix : List[List[int]], key1 : int,
key2 : int, key3 : int, msg : str) :
    # kamus lokal
    maxColMatrix = max(max(row) for row in colMatrix)
    A2 = [[(key1 * maxColMatrix) for _ in
range(len(colMatrix))] for _ in range(len(colMatrix[0]))]

    # algoritma

    # A1
    A1 = [[(key1 * maxColMatrix) for _ in
range(len(colMatrix))] for _ in range(len(colMatrix[0]))]
    for i in range(len(A1)):
        for j in range(len(A1[i])):
            A1[i][j] = int(A1[i][j]- colMatrix[i][j])
    A1 = np.array(A1)

    # A2
    maxA1 = max(max(row) for row in A1)
    A2 = [[(key1 * maxA1) for _ in range(len(A1))] for _
in range(len(A1[0]))]
    for i in range(len(A2)):
        for j in range(len(A2[i])):
            A2[i][j] = int(A2[i][j] - A1[i][j])
    A2 = np.array(A2)

    # B1 & B2
    B1 = A1 - A2
    B2 = B1
    for i in range(len(B1)):
        for j in range(len(B1[i])):
            B2[i][j] = int((B2[i][j] * key2) + A2[i][j])

    # SVD decomposition
    UB1, SB1, VB1 = np.linalg.svd(B1)
    S_matrix1 = np.zeros_like(B1, dtype=float) #
converting into diagonal matrix
    np.fill_diagonal(S_matrix1, SB1)
    UB2, SB2, VB2 = np.linalg.svd(B2)
    S_matrix2 = np.zeros_like(B2, dtype=float)
    np.fill_diagonal(S_matrix2, SB2)

    # build new interchanged matrix
    C1 = UB1 @ S_matrix2 @ VB1.T
```

```
    C1.astype(int)
    C2 = UB2 @ S_matrix1 @ VB2.T
    C2.astype(int)

    # combine matrix (addition with key)
    F = (C1 + key3 * C2)
    F.astype(int)
    # print(F)

    # min and max value of F
    MI = min(min(row) for row in F)
    MA = max(max(row) for row in F)

    # normalizing matrix
    FF = ((F - MI) * 256 / (MA - MI)).astype(int)

    # keeping the MI and MA val in the FF matrix
    FF[0][0] = 0 if MI >= 0 else 0.1 # sign part
    FF[0][1] = int(int(abs(MI)) / 255) # scaled integer part
    FF[0][2] = int(abs(MI)) % 255 # remainder part
    FF[0][3] = abs(MI) - int(abs(MI)) # fractional part

    FF[len(FF) - 1][len(FF[0]) - 1] = 0 if MA >= 0 else 0.1
# sign part
    FF[len(FF) - 1][len(FF[0]) - 2] = int(int(abs(MA)) /
255) # scaled integer part
    FF[len(FF) - 1][len(FF[0]) - 3] = int(abs(MA)) % 255 #
remainder part
    FF[len(FF) - 1][len(FF[0]) - 4] = abs(MA) -
int(abs(MA)) # fractional part

    # encrypt message into the FF matrix (final processing)

    return np.array(encryptMessage(FF, msg))

# main algorithm

# Driver
# Open an image
IMG_FILE = Image.open("images/shrek.jpg")

# Convert the image to RGB mode (if not already)
image = IMG_FILE.convert("RGB")

# Get the width and height of the image
width, height = image.size

# Initialization
red = np.array([[0 for _ in range(height)] for _ in
range(width)])
green = np.array([[0 for _ in range(height)] for _ in
range(width)])
blue = np.array([[0 for _ in range(height)] for _ in
range(width)])

# filling all the matrix
for y in range(height):
    for x in range(width):
        r, g, b = image.getpixel((x, y))
        red[y][x] = r
        green[y][x] = g
        blue[y][x] = b

# generating key for encrypt
key1 = random.uniform(7, 10)
key2 = random.uniform(1, 3)
while key2 == 1.5 :
    key2 = random.uniform(0.9, 3)
key3 = random.uniform(0, 3)

msg = "insert message here..."
redNew = encryptImage(red, key1, key2, key3, msg)
rgbMatrixNew = np.stack((encryptImage(red, key1, key2,
key3, msg), encryptImage(green, key1, key2, key3, msg),
encryptImage(green, key1, key2, key3, msg)), axis=-1)
rgbMatrixNew = np.array(rgbMatrixNew,
dtype=np.uint8)
encryptedImage = Image.fromarray(rgbMatrixNew)
encryptedImage.save("shrek.jpg")

# testing : dimension
print(f"width : {width}, height : {height}")

# Testing : time
start_time = time.time()
encryptImage(red, key1, key2, key3, msg)
end_time = time.time()
print(f"elapsed time : {end_time - start_time}")

# testing : similarity
print(f"similarity : {cosine_similarity(redNew.ravel(),
red.ravel())}")
```

C. Pembahasan

Berdasarkan hasil yang telah diperoleh, dapat dinyatakan bahwa gambar yang dihasilkan berbeda jauh dengan gambar asli dan tidak dapat dilakukan pengambilan informasi secara langsung. Selain itu, dapat dilihat pula bahwa hasil enkripsi memiliki tingkat keberhasilan yang cukup besar dengan syarat bahwa gambar memiliki dimensi persegi.

Selanjutnya, dapat ditinjau bahwa gambar yang diproses merupakan gambar yang berwarna sehingga algoritma enkripsi berhasil melakukan enkripsi terhadap gambar berwarna maupun gambar tanpa warna. Selain itu, wkatu yang dibutuhkan untuk melakukan proses enkripsi berbanding lurus dengan dimensi dari gambar dan semakin tinggi dimensi gambar tersebut, maka proses semakin lama. Hal ini dapat dilihat melalui perbandingan hasil uji pada gambar 1 dan 2, serta gambar 2 dan gambar 3.

Kemudian, meninjau kembali pada sisi kemiripan, gambar setelah enkripsi memiliki tingkat kemiripan yang relatif cukup jauh dari gambar asli sehingga algoritma enkripsi berhasil melakukan enkripsi terhadap informasi yang diinginkan. Hal ini disebabkan karena operasi yang digunakan bersifat sederhana sehingga tidak memungkinkan bagi gambar untuk memiliki kompleksitas yang rumit dalam melakukan enkripsi pada setiap informasi yang ada.

Selain itu, algoritma yang disusun memiliki pengaruh juga pada proses yang dilakukan karena algoritma ini

memiliki kompleksitas $O(n^2)$ dengan n merupakan Panjang dimensi dari gambar. Hal ini utamanya berkontribusi pada proses enkripsi sehingga waktu yang dibutuhkan cukup lama apabila gambar yang dienkripsi memiliki dimensi yang besar.

Selanjutnya, algoritma enkripsi yang digunakan juga memiliki beberapa kelebihan, yakni waktu enkripsi yang relatif rendah untuk jumlah piksel tertentu dan tingkat kemiripan yang relatif cukup kecil. Hal ini dapat mendorong pemanfaatan algoritma enkripsi dalam keseharian untuk melindungi informasi yang ada. Namun, di luar itu, terdapat pula kekurangan dari algoritma yang dimiliki, yakni gambar yang menjadi masukan dari algoritma masih harus merupakan gambar dengan dimensi persegi. Hal ini dapat menjadi pertimbangan dan saran untuk kedepannya dalam merancang algoritma yang lebih baik dan lebih aksesibel untuk berbagai jenis gambar. Selain itu, terdapat kekurangan juga bahwa kompleksitas algoritma yang dimiliki masih kurang baik karena dapat menigkat secara kuadratik, yang cenderung kurang baik dalam hal performa, sehingga dapat memperlambat proses dan/atau memperberat kerja dari komputer pengguna

V. KESIMPULAN

Informasi merupakan hal yang signikan dan perlu dilindungi sehingga diperlukan algoritma enkripsi yang dapat melindungi informasi yang ada. Berdasarkan hasil eksperimen terhadap susunan algoritma yang telah dibuat, algoritma enkripsi pada makalah ini memiliki tingkat keberhasilan yang cukup baik, waktu performansi yang relatif cukup baik, dan tingkat kemiripan terhadap gambar sebelum proses enkripsi yang relatif cukup rendah. Meskipun begitu, algoritma ini dapat digunakan untuk keperluan sehari-hari karena dapat melindungi secara sederhana. Dengan demikian, penggunaan algoritma enkripsi ini dapat digunakan untuk keperluan sehari-hari dengan pertimbangan lebih lanjut.

VII. ACKNOWLEDGMENT

Dengan ini, saya menyatakan terimakasih pada Pak Rinaldi Munir, selaku dosen pengampu Mata Kuliah Aljabar Linear dan Geometri, karena telah membimbing Saya hingga titik ini.

REFERENCES

[1] El Abbadi, N. K., Mohamad, A., & Abdul-Hameed, M. (2014). Image encryption based on singular value decomposition. *Journal of Computer Science*, *10*(7), 1222–1230.W.-K. Chen, *Linear Networks and Systems* (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.

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, 27 Desember 2024

Brian A. Hadian
13523048