Implementasi Penyisipan Tanda Tangan Digital dengan Steganografi yang Disertai *Image Compression*

Wisnu Aditya Samiadji (13519093)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13519093@std.stei.itb.ac.id / wisnuaditya14@gmail.com

***Abstrak*— Steganografi adalah seni dan ilmu menulis pesan tersembunyi atau menyembunyikan pesan dengan suatu cara sehingga selain si pengirim dan si penerima, tidak ada seorang pun yang mengetahui atau menyadari bahwa ada suatu pesan rahasia. Tentu jika ruang penyimpanan yang digunakan untuk gambar tersebut kecil, kita dapat menyimpan dan menerima gambar dengan pesan rahasia tersebut. Akan tetapi, pesan rahasia tersebut berpotensi rusak apabila gambar stego dimampatkan. Pada karya tulis ini, penulis ingin bereksperimen dengan implementasi steganografi untuk memasukkan tanda tangan digital (*digital signature*), kemudian gambar stego akan dimampatkan dengan metode *image compression* tertentu.**

***Kata Kunci*—Steganografi, Tanda Tangan Digital, Image Compression**

I. Pendahuluan

Steganografi adalah seni dan ilmu menulis pesan tersembunyi atau menyembunyikan pesan dengan suatu cara sehingga selain si pengirim dan si penerima, tidak ada seorang pun yang mengetahui atau menyadari bahwa ada suatu pesan rahasia. Teknik ini telah digunakan dalam sistem komunikasi untuk mengirimkan pesan rahasia tanpa terdeteksi. Dalam media digital, gambar, video, dan file audio telah digunakan sebagai kedok untuk mengirim pesan tersembunyi.

Tanda tangan digital adalah skema matematika untuk memverifikasi keaslian pesan atau dokumen digital. Tanda tangan digital yang valid, yang memenuhi prasyarat, memberikan keyakinan yang sangat tinggi kepada penerima bahwa pesan tersebut dibuat oleh pengirim yang dikenal, dan bahwa pesan tersebut tidak diubah dalam perjalanan.

Teknik *image compression* semakin marak digunakan seiring meningkatnya kebutuhan media digital. Perkembangan teknologi fotografi menyebabkan ukuran citra digital yang dihasilkan semakin besar dan memakan memori yang juga besar. Selain itu, orang sering sekali menyimpan gambar atau berkas lainnya dalam sebuah cloud storage, seperti Google Drive. Seperti yang kita ketahui, kuota penyimpanan yang didapatkan oleh pengguna gratis (*free user*) itu terbatas. Dalam jangka waktu panjang, kuota tersebut akan segera habis jika kita menyimpan berkas ukuran besar terus menerus.

Gambar 1. Penyimpanan terbatas pada *cloud storage* Google Drive (sumber: milik pribadi)

Dalam makalah ini, penulis ingin menyisipkan tanda tangan digital yang dibangkitkan dengan algoritma RSA (Rivest-Shamir-Adleman) pada beberapa gambar dengan teknik steganografi berbasis LSB (*Least Significant Bit*). Kemudian, gambar stego akan dimampatkan dengan metode *image compression*.

II. Landasan Teori

A. *Steganografi*

Steganografi berasal dari kata Yunani *steganos*, yang berarti tertutup atau rahasia, dan *graphien* (tulisan atau gambar). Steganografi adalah praktik menyembunyikan informasi di dalam pesan lain, sehingga keberadaan informasi tersebut tidak terlihat. Contoh dari steganografi konvensional adalah menyembunyikan tulisan seperti menggunakan tinta yang tidak terlihat. Dalam komputasi, sebuah pesan, gambar, video, atau file lainnya disembunyikan di dalam file lain. Oleh karena itu, steganografi terdiri dari dua properti, yaitu pesan rahasia (*secret message*) dan pesan sampul (*cover message*).

Ada beberapa kriteria yang perlu dipertimbangkan pada steganografi, yaitu:

- *Imperceptibility*, pesan rahasia tidak mudah dirasakan.
- *Fidelity*, pesan sampul sebagian besar tetap sama seperti sebelum pesan rahasia disisipkan.
- *Recovery*, pesan rahasia harus dapat dipulihkan.
- *Capacity*, terkait dengan jumlah informasi yang dapat disimpan oleh pesan rahasia.

Peningkatan pada satu faktor dapat menyebabkan pertukaran pada faktor-faktor lainnya. Sebagai contoh, sebuah teknik steganografi yang dapat menyembunyikan informasi dengan sangat baik mungkin tidak dapat menyimpan banyak informasi, namun yang kurang aman mungkin dapat menyimpan lebih banyak informasi.

Tidak seperti kriptografi, yang dirancang untuk tidak dapat dimengerti oleh pihak ketiga yang tidak berwenang, steganografi dirancang untuk disembunyikan dari pihak ketiga. Data rahasia tidak hanya harus ditemukan - sebuah upaya yang bermasalah - tetapi juga harus dienkripsi, yang bisa jadi sulit.

B. *Tanda Tangan Digital*

Sejak zaman dahulu, tanda tangan sudah digunakan untuk otentikasi dokumen cetak. Tanda tangan memiliki karakterisitik sebagai berikut:

- Tanda tangan adalah bukti yang otentik
- Tanda tangan tidak dapat dilupakan
- Tanda tangan tidak dapat dipindah untuk digunakan ulang
- Dokumen yang telah ditandatangani tidak dapat diubah
- Tanda tangan tidak dapat disangkal


Hari: Senin, Tanggal 12 September 2016
Waktu: 14.00 Wib
Tempat: Aula

Demikian kami buat surat ini dengan sebenarnya. Harap semua peserta untuk hadir sesuai jadwal yang telah disebutkan di atas. Terima kasih.

Tertanda tangan

Mr Mukidi

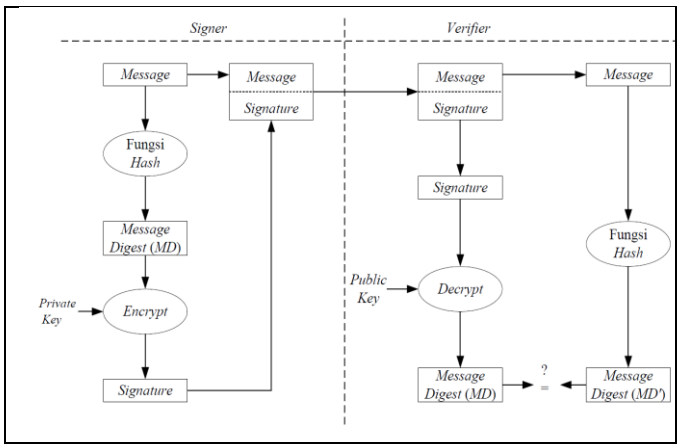
Master of Ceremony
Doktor Mukidi, Mpd

Gambar 2. Tanda tangan pada sebuah surat resmi (sumber: 28-Tanda-tangan-digital-2023)

Fungsi tanda tangan pada dokumen kertas juga diterapkan untuk otentikasi pada data digital (pesan , dokumen elektronik). Tanda tangan untuk data digital dinamakan tanda tangan digital (*digital signature*). Tanda tangan digital di dalam konteks kriptografi tidak sama dengan tanda tangan yang didigitisasi (*digitized signature*) dengan cara dipindai atau difoto.

Ada 2 proses dalam tanda tangan digital, yaitu menandatangani pesan (signing) dan verifikasi pesan (*verification*). Tanda tangan digital adalah nilai kriptografis yang bergantung pada isi pesan dan kunci, tidak seperti pada dokumen cetak yang selalu sama.

Ada 2 cara yang dilakukan untuk menandatangani pesan secara digital, yaitu dengan mengenkripsi pesan (khusus untuk pesan rahasia) dan menggunakan fungsi hash dan kriptografi kunci publik (untuk pesan yang tidak perlu rahasia). Makalah ini akan menggunakan kombinasi fungsi hash dan kriptografi kunci publik untuk implementasinya, dengan skema sebagai berikut:



Gambar 3. Skema penggunaan tanda tangan digital dengan kombinasi fungsi hash dan kriptografi kunci publik (sumber: 28-Tanda-tangan-digital-2023)

C. *Image Compression*

Image Compression atau kompresi citra merupakan operasi citra yang dilakukan dengan tujuan menghilangkan redundansi yang ditemukan pada sebuah citra. Sebuah citra biasanya perlu dikompresi karena representasi citra digital membutuhkan memori yang besar. Kompresi citra bertujuan untuk mengurangi kebutuhan memori untuk ruang penyimpanan tanpa mengurangi kualitas citra secara visual.

Ada 2 jenis metode kompresi untuk citra digital, yaitu:

- ***Lossy Image Compression***: Merupakan tipe metode kompresi citra yang menghasilkan citra hasil pemampatan yang hampir sama dengan citra semula. Ada informasi yang hilang akibat pemampatan, tetapi dapat ditolerir oleh persepsi visual. Tipe metode kompresi citra ini bertujuan untuk memperoleh nisbah pemampatan yang tinggi. Contoh metode kompresi yang termasuk pada *Lossless Image Compression* diantaranya yaitu *Discrete Cosine Transform (DCT) Compression.*

- ***Lossless Image Compression****:* Merupakan tipe metode kompresi citra yang selalu menghasilkan citra hasil penirmampatan yang tepat sama dengan citra

semula, pixel per pixel. Citra hasil kompresi tidak memiliki informasi yang hilang akibat pemampatan. Selain itu, citra hasil kompresi juga memiliki nisbah pemampatan rendah, namun kualitas citra mampat tetap tinggi. Tipe metode ini dibutuhkan untuk memampatkan citra yang tidak boleh terdegradasi akibat pemampatan, misalnya citra hasil rontgen / X-ray. Contoh metode yang termasuk dalam *Lossless Image Compression* diantaranya yaitu *Huffman Coding*

Untuk implementasi pada makalah ini, penulis merasa jika *lossy image compression* tidak bisa digunakan untuk penyisipan tanda tangan digital dengan steganografi. Oleh karena itu, penulis memilih Huffman Coding yang merupakan metode *lossless image compression.*

D. Least Significant Bit Steganography

Di dalam setiap byte bit bitnya tersusun dari kiri ke kanan dalam urutan yang kurang berarti (*least significant bits* atau LSB) hingga bit bit yang berarti (*most significant bits* atau MSB). Steganografi dengan metode LSB menggunakan fakta bahwa mengubah nilai bit LSB tidak mengubah persepsi citra secara keseluruhan. Contohnya, jika 11010001 misalnya menyatakan warna merah , maka 11010000 juga menyatakan warna merah yang berubah sangat sedikit, sehingga tidak dapat dibedakan dengan mata manusia. Bit-bit pesan yang disembunyikan di dalam citra harus dapat diekstraksi kembali. Caranya adalah dengan membaca byte-byte di dalam citra , mengambil bit LSB-nya, dan merangkainya kembali menjadi bit-bit pesan.

E. Algoritma RSA

Algoritma RSA (Rivest-Shamir-Adleman) merupakan algoritma kunci publik yang paling terkenal dan paling banyak aplikasinya. Algoritma ini Dibuat oleh tiga peneliti dari MIT (*Massachussets Institute of Technology*), yaitu Ronald Rivest , Adi Shamir , dan Leonard Adleman, pada tahun 1976. Keamanan algoritma RSA terletak pada sulitnya memfaktorkan bilangan bulat yang besar menjadi faktor-faktor prima. Algoritma pemfaktoran yang tercepat saat ini memiliki kompleksitas

$$O(\exp(\sqrt[3]{\frac{64}{9}} \times b \times \log(b^2))$$

untuk bilangan bulat n sepanjang b bit. Hingga saat ini belum ditemukan algoritma pemfaktoran bilangan bulat besar dalam waktu polinomial. Fakta inilah yang membuat algoritma RSA dianggap masih aman untuk saat ini . Semakin panjang bilangan bulatnya, maka semakin lama waktu yang dibutuhkan untuk memfaktorkannya .

F. Huffman Coding

Huffman Coding merupakan metode yang didasarkan pada algoritma *Greedy*. Prinsip kerjanya adalah pixel dengan nilai keabuan yang sering muncul dikodekan dengan panjang bit yang lebih sedikit, sebaliknya nilai keabuan yang jarang muncul dikodekan dengan bit yang lebih panjang. Langkah-langkah Huffman Coding adalah sebagai berikut:

- Setiap nilai keabuan dinyatakan sebagai simpul. Setiap simpul di assign dengan frekuensi kemunculan nilai keabuan tersebut
- Urutkan secara menaik simpul simpul berdasarkan frekuensi kemunculannya.
- Gabung dua buah simpul yang mempunyai frekuensi kemunculan paling kecil pada sebuah akar. Akar mempunyai frekuensi yang merupakan jumlah dari frekuensi dua buah pohon penyusunnya.
- Ulangi langkah 2 sampai tersisa hanya satu buah pohon biner
- Beri label setiap sisi pada pohon biner. Sisi kiri dilabeli dengan 0 dan sisi kanan dilabeli dengan 1.
- Telusuri pohon biner dari akar ke daun. Barisan label-label sisi dari akar ke daun menyatakan kode Huffman untuk derajat keabuan yang bersesuaian

III. IMPLEMENTASI

Implementasi yang digunakan untuk makalah ini menggunakan Google Colab dengan bahasa pemrograman Python yang terbentuk sebagai Python Notebook yang berformat ipynb. Program dibuat dengan bantuan beberapa library bawaan Python seperti hashlib. Selain itu, penulis juga menggunakan library *OpenCV for Python* untuk membaca data gambar.

A. Program Utama

Program dimulai dengan memilih 1 dari 4 aksi, yaitu membangkitkan kunci privat dan publik, menandatangani dan memampatkan gambar, melakukan verifikasi tanda tangan yang tersisipkan pada gambar, dan mematikan program.

```
def image_Steganography ():
    session = True;
    while (session == True):
      a = input("Image Steganography \n 1.
Generate Key \n 2. Sign and encode the data  \n 3.
Decode and verify the data \n 4. Exit\n Your input
is: ")
      userinput = int(a)
      if (userinput == 1):
        generateKeys()
        private_key, public_key = loadKeys()
      elif (userinput == 2):
        image_name = input("Enter image name(with
extension): ")
        fname = input("Enter the name of new
encoded image(with extension): ")
        message = input("Enter signature message:
")
```

```
        private_key, public_key = loadKeys()
        encrypted_message = message.encode()
        signature = sign(encrypted_message,
private_key)
        print("\nEncoding....")
        encode_text(fname, image_name,
signature.hex())
        compress_Image(fname, 1)
      elif (userinput == 3):
        image_name = input("Enter the name of the
steganographed image that you want to decode (with
extension) :")
        print("\nDecoding....")
        private_key, public_key = loadKeys()
        message = input("Enter signature message:
")
        encrypted_message = message.encode()
        decoded_message =
bytes.fromhex(decode_text(image_name))
        print("Decoded message is :",
decoded_message)
        valid = verify(encrypted_message,
decoded_message, public_key)
         if(valid == True):
          print("The signature is valid.")
        else:
          print("The signature is invalid.")
      else:
          session = False
```

B. Least Significant Bit Steganography

Penyisipan tanda tangan dimulai dengan memasukkan nama berkas gambar awal, nama berkas gambar stego yang akan dibangkitkan, dengan pesan yang digunakan untuk tanda tangan. Kemudian, gambar akan dibaca dengan OpenCV dan proses dilanjutkan dengan menjalankan fungsi di bawah:

```
def stego_encode(image, secret_message):
  # calculate the maximum bytes to encode
  maxEmbedSize = image.shape[0] *
image.shape[1] * 3 // 8
  print("Maximum bytes to encode:",
maxEmbedSize)
  #Check if the number of bytes to encode is
less than the maximum bytes in the image
  if len(secret_message) > maxEmbedSize:
      raise ValueError("Error encountered
insufficient bytes")
  secret_message = str(secret_message) +
"@@@@@" # you can use any string as the
delimeter
  data_index = 0
  binary_secret_msg =
messageToBinary(secret_message)
  data_len = len(binary_secret_msg) #Find the
length of data that needs to be hidden
  for values in image:
      for pixel in values:
          # convert RGB values to binary format
          r, g, b = messageToBinary(pixel)
          # modify the least significant bit
only if there is still data to store
          if data_index < data_len:
              pixel[0] = int(r[:-1] +
binary_secret_msg[data_index], 2)
              data_index += 1
          if data_index < data_len:
               pixel[1] = int(g[:-1] +
binary_secret_msg[data_index], 2)
               data_index += 1
           if data_index < data_len:
               pixel[2] = int(b[:-1] +
binary_secret_msg[data_index], 2)
              data_index += 1
          # if data is encoded, just break out
of the loop
          if data_index >= data_len:
              break
  return image
```

Berikut adalah fungsi untuk mengeluarkan tanda tangan dari berkas gambar stego:

```
def stego_decode(image):
  binary_data = ""
  for values in image:
      for pixel in values:
          #convert the red,green and blue
values into binary format
          r, g, b = messageToBinary(pixel)
#convert the red,green and blue values into
binary format
          binary_data += r[-1]
          binary_data += g[-1]
          binary_data += b[-1]
  # split by 8-bits
```

```
  all_bytes = [ binary_data[i: i+8] for i in
range(0, len(binary_data), 8) ]
  # bits to characters
  decoded_data = ""
  for byte in all_bytes:
      decoded_data += chr(int(byte, 2))
      if decoded_data[-5:] == "@@@@@": #check
if we have reached the delimeter which is
"@@@@@"
          break
  return decoded_data[:-5] #remove the
delimeter.
```

C. RSA

Proses pembangkitan kunci RSA dibantu dengan library *RSA for Python*. Ukuran kunci yang digunakan untuk testing adalah 512 bit, namun menurut NIST dianjurkan menggunakan panjang kunci 2048 bit. Kunci RSA kemudian disimpan dengan format PEM string.

```
def generateKeys():
    (publicKey, privateKey) = RSA.newkeys(512)
    with open('publicKey.pem', 'wb') as p:
        p.write(publicKey.save_pkcs1('PEM'))
    with open('privateKey.pem', 'wb') as p:
        p.write(privateKey.save_pkcs1('PEM'))
def loadKeys():
    with open('publicKey.pem', 'rb') as p:
        publicKey =
RSA.PublicKey.load_pkcs1(p.read())
    with open('privateKey.pem', 'rb') as p:
        privateKey =
RSA.PrivateKey.load_pkcs1(p.read())
    return privateKey, publicKey
```

Proses *signing* untuk disisipkan dalam gambar dimulai dengan memasukkan nama berkas gambar awal dan pesan yang digunakan untuk membangkitkan tanda tangan. Tanda tangan dibangkitkan dengan kunci privat pengguna.

Proses verifikasi tanda tangan yang tersisip dalam gambar dimulai dengan memasukkan nama berkas gambar stego dan pesan. Jika pesan yang dimasukkan pada proses ini berbeda dengan yang digunakan untuk membangkitkan tanda tangan (berarti tanda tangannya berbeda dengan tanda tangan orang yang diinginkan sebagai pengirim), maka tanda tangan akan dinyatakan tidak valid, dan sebaliknya. Proses verifikasi ini menggunakan kunci publik pengguna/pengirim.

Dalam eksekusinya, tanda tangan yang dibangkitkan memiliki tipe data bytes. Karena pesan yang tersisipkan saat di-decode akan menghasilkan string dan tidak bisa dibandingkan dengan bytes meski nilainya sama, maka tanda tangan yang disisipkan dalam gambar merupakan bentuk hex-nya. Dengan begitu, proses verifikasi bisa dilakukan (dapat dilihat pada *section* "Program Utama", ada baris yang *signature* menjadi bentuk hex dari *signature* tersebut.

D. Huffman Coding

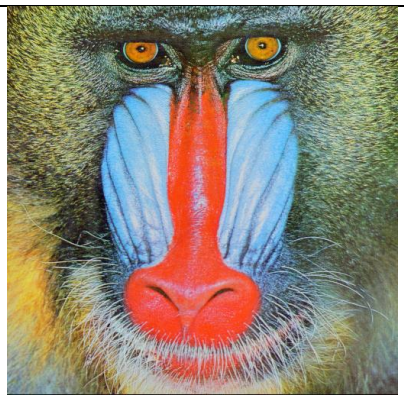
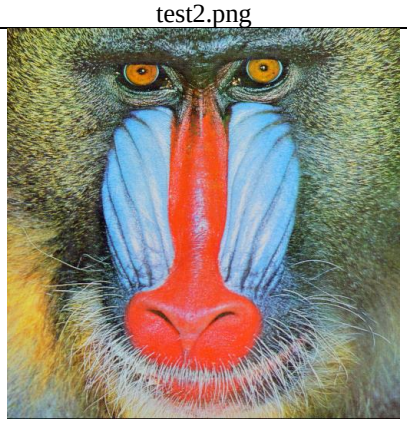
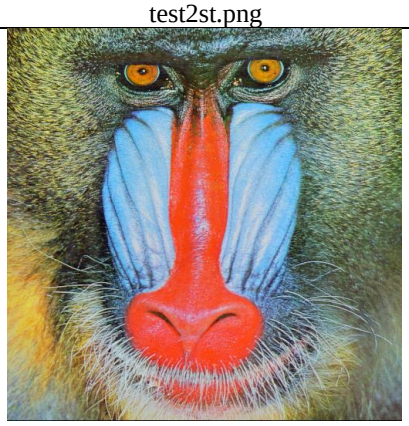
Berikut adalah potongan kode program untuk membuat Huffman Tree:

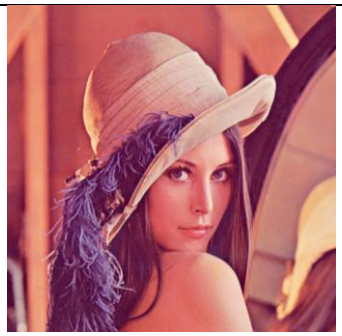
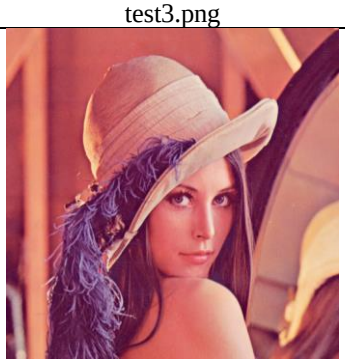
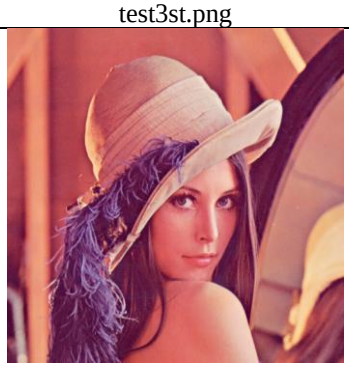
```
   letters = []
  only_letters = []
  for letter in my_string:
      if letter not in letters:
          frequency =
my_string.count(letter)              #frequency of
each letter repetition
          letters.append(frequency)
          letters.append(letter)
          only_letters.append(letter)
  nodes = []
  while len(letters) > 0:
      nodes.append(letters[0:2])
      letters = letters[2:]
  # sorting according to frequency
  nodes.sort()
  huffman_tree = []
  huffman_tree.append(nodes) #Make each unique
character as a leaf node
  # huffman tree generation
  newnodes = combine_nodes(nodes, huffman_tree)
  huffman_tree.sort(reverse = True)
  checklist = []
  for level in huffman_tree:
      for node in level:
          if node not in checklist:
              checklist.append(node)
          else:
              level.remove(node)
```

IV. Hasil dan Analisis

A. *Hasil*

Berikut merupakan hasil eksekusi program dengan 3 buah gambar berbeda, masing-masing dengan gambar stego yang tidak dimampatkan dan yang dimampatkan:

<table>
<tr><th>No</th><th>Gambar</th></tr>
<tr><td rowspan="5">1</td><td> 
test1.png</td></tr>
<tr><td> 
test1st.png</td></tr>
<tr><td> 
test1stcompressed.png</td></tr>
<tr><td>
<table>
<tr><td>Tanda Tangan Valid</td><td>Ya</td></tr>
<tr><td>Ukuran Stego Tanpa Kompresi</td><td>407.91KB</td></tr>
<tr><td>Ukuran Stego dengan Kompresi</td><td>339.66KB</td></tr>
<tr><td>Nisbah</td><td>83.27%</td></tr>
</table>
</td></tr>
<tr><td rowspan="5">2</td><td> 
test2.png</td></tr>
<tr><td> 
test2st.png</td></tr>
<tr><td> 
test2stcompressed.png</td></tr>
<tr><td>
<table>
<tr><td>Tanda Tangan Valid</td><td>Ya</td></tr>
<tr><td>Ukuran Stego Tanpa Kompresi</td><td>628.08KB</td></tr>
<tr><td>Ukuran Stego dengan Kompresi</td><td>617.94KB</td></tr>
<tr><td>Nisbah</td><td>98.38%</td></tr>
</table>
</td></tr>
</table>

3	 test3.png	
	 test3st.png	
	 test3stcompressed.png	
	Tanda Tangan Valid	Ya
	Ukuran Stego Tanpa Kompresi	508.92KB
	Ukuran Stego dengan Kompresi	468.93KB
	Nisbah	92.14%

Tabel 1

Berikut adalah contoh keluaran program berisi perbandingan apabila pesan yang dijadikan untuk tanda tangan tidak sama dengan dan sama dengan yang tersisipkan pada gambar stego. Gambar yang digunakan adalah test2 dengan pesan "This is Wisnu" digunakan untuk membangkitkan tanda tangan digital:

```
Enter signature message: This is Wisnu

Decoded message is :
b'M@\xe1UH\x81G\xdf\xf44\xc8\x04\xa13\
xac\xe6d\x01\x1b0<{<]2\xf2.8\r]\x99\xf
d\x17\xc7\x05\xd1\x87\xcb\x04\xb2\x8e\
xe2V\xce\xa0s\x97iz\xb2b\x1d\x8a\xd9q\
xb7\x84\xfe\x91\xec\xa5\xf20\xe0'
The signature is valid.
```

```
Enter signature message: This is AAA

Decoded message is :
b'M@\xe1UH\x81G\xdf\xf44\xc8\x04\xa13\
xac\xe6d\x01\x1b0<{<]2\xf2.8\r]\x99\xf
d\x17\xc7\x05\xd1\x87\xcb\x04\xb2\x8e\
xe2V\xce\xa0s\x97iz\xb2b\x1d\x8a\xd9q\
xb7\x84\xfe\x91\xec\xa5\xf20\xe0'
The signature is invalid.
```

Tabel 2

B. Analisis

Pada tabel 1, pengecekan validasi tanda tangan berhasil meskipun gambar stego dimampatkan. Selain itu, gambar stego yang dihasilkan memiliki tingkat kemiripan yang tinggi dengan gambar awal.

Pada tabel 2, karena signature messagenya berbeda, tanda tangannya tidak valid. Misal AAA dianggap sebagai pengirim aslinya dan penerima mengetahui kalau *signature message* dia adalah "This is AAA". Tetapi, karena tanda tangan yang tersisipkan berbeda, maka jelas kalau AAA bukanlah pengirim gambar stego ini.

V. Kesimpulan dan Saran

Berdasarkan tabel 1 dan 2, implementasi tanda tangan digital untuk steganografi gambar yang kemudian dimampatkan cukup berhasil. Selain itu, hipotesis penulis tentang penggunaan metode *lossless image compression* terbukti menghasilkan data stego yang tetap utuh dan bisa divalidasi, tentu dengan ukuran berkas yang lebih kecil dari seharusnya.

Saran untuk ke depannya agar mengoptimasi algoritma untuk penggunaan gambar dengan resolusi tinggi. Penulis merasa cukup pesimis untuk menggunakan gambar resolusi tinggi karena pada makalah yang penulis susun sebelumnya, penggunaan gambar tipe tersebut membutuhkan waktu eksekusi yang sangat lama.

Project Link

wisnu1314/IF4020_Kriptografi_Tugas-Makalah_Wisnu-Aditya (github.com)

Acknowledgment

Penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa, karena berkat kehadiran-Nya, penulis dapat menyelesaikan makalah ini. Tak lupa penulis berterima kasih kepada kedua orang tua penulis, serta Bapak Dr. Ir. Rinaldi, MT sebagai dosen mata kuliah Kriptografi. Terakhir, penulis ingin berterima kasih kepada sahabat penulis yang bersedia mebantu menyumbangkan ide selama pembuatan makalah ini.

REFERENSI

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2022-2023/08-Steganografi-Bagian1-2023.pdf (Diakses pada Mei 2023)

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2022-2023/19-Algoritma-RSA-2023.pdf (Diakses pada Mei 2023)

[3] https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2022-2023/28-Tanda-tangan-digital-2023.pdf (Diakses pada Mei 2023)

[4] https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2022-2023/25-Image-Compression-Bagian1-2022.pdf (Diakses pada Mei 2023)

[5] https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2022-2023/Makalah/Makalah-IF4073-Citra-Sem1-2022%20(17).pdf (Diakses pada Mei 2023)

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, 22 Mei 2023

Ttd

Wisnu Aditya Samiadji 13519093