Optimasi Kapasitas Steganografi Digital dengan Teknik Kompresi *Huffman Coding*

Bagas Aryo Seto - 13521081
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13521081@std.stei.itb.ac.id

***Abstract*—Steganografi adalah teknik atau ilmu yang mempelajari cara untuk menyembunyikan pesan rahasia sedemikian sehingga keberadaan pesan tersebut tidak terdeteksi. Terdapat beberapa kriteria agar steganografi dapat dikatakan berkualitas, salah satunya adalah kapasitas penyimpanan pesan yang besar. Kapasitas suatu steganografi dapat diperbesar dengan melakukan kompresi terhadap pesan rahasia sebelum dilakukan proses steganografi. *Huffman Coding* merupakan salah satu metode kompresi digital yang paling banyak digunakan. Penelitian ini bertujuan untuk meneliti implementasi kompresi *Huffman Coding* pada pesan rahasia sebagai upaya optimasi kapasitas penyimpanan steganografi digital.**

***Keywords*—Steganografi, *Huffman Coding*, Kompresi**

I. PENDAHULUAN

Perkembangan era digital yang semakin pesat saat ini membawa manusia dalam dunia yang penuh akan data dan informasi. Data dan informasi merupakan hal yang sangat penting bagi setiap orang. Terdapat data dan informasi yang bersifat publik dan terbuka untuk semua orang, dan terdapat juga data yang bersifat pribadi atau rahasia. Data dan informasi yang bersifat pribadi harus dijaga keamanannya dari pihak-pihak tidak bertanggung jawab yang berusaha untuk meretas atau mencuri data rahasia tersebut untuk kepentingan pribadi. Oleh karena itu, diperlukan cara untuk menjaga keamanan atau menyembunyikan keberadaan data.

Salah satu cara untuk menyembunyikan data adalah menggunakan steganografi. Steganografi merupakan seni, teknik, atau ilmu menyembunyikan pesan rahasia (*information hiding*) dengan suatu cara sedemikian sehingga keberadaan pesan tersebut tidak terdeteksi. Sejarah steganografi dimulai oleh bangsa Yunani. Dalam buku *Histories of Herodatus* yang ditulis oleh Herodatus (485 – 525 BC), diceritakan kisah perang antara kerajaan Rusia dan rakyat Yunani. Herodatus mengisahkan cara Histaiaeus, penguasa Yunani pada saat itu, mengirim pesan rahasia ke Aristagoras of Miletus untuk melawan Persia. Histaieus menulis pesan dengan cara menato pesan pada kepala budak yang sudah dibotaki, lalu membiarkan rambut para budak tumbuh kembali agar pesan tersembunyi sebelum mengirimkan para budak ke Aristagoras. Selanjutnya Aristagoras akan membotaki kembali para budak agar pesan rahasia dapat dibaca [1].

Terdapat beberapa kriteria steganografi dapat dikatakan berkualitas, yaitu *imperceptible*, *fidelity*, *recovery*, *capacity*, dan *robustness*. Salah satu kriteria tersebut yakni *capacity* berarti steganografi harus memiliki kapasitas penyimpanan yang cukup untuk menyimpan pesan rahasia. Steganografi memiliki kapasitas yang terbatas karena sifatnya yang menyembunyikan sehingga diperlukan kapasitas *cover* yang lebih besar daripada kapasitas pesan rahasianya. Salah satu cara untuk menambah kapasitas pesan rahasia pada steganografi adalah dengan melakukan kompresi terhadap pesan terlebih dahulu.

II. TEORI DASAR

A. Citra Digital

Citra adalah gambar pada suatu bidang dua dimensi. Agar sebuah citra dapat diolah dengan komputer digital, maka suatu citra harus direpresentasikan secara numerik dengan nilai-nilai diskrit. Representasi citra dengan fungsi kontinu menjadi nilai-nilai diskrit disebut digitalisasi citra. Citra yang dihasilkan dari proses digitalisasi citra tersebut adalah citra digital. Pada umumnya citra digital berbentuk persegi panjang dengan tinggi dan lebarnya merupakan dimensi ukuran atau resolusi dari citra.[2]

Setiap elemen pada citra digital disebut dengan *image element*, *picture element*, *pel*, atau *pixel*. Citra dengan resolusi m x n memiliki mn buah *pixel*. Setiap *pixel* direpresentasikan dengan bilangan biner n-bit. Citra digital diklasifikasikan menjadi beberapa kategori berdasarkan jumlah bit represntasi *pixel*nya:[2]

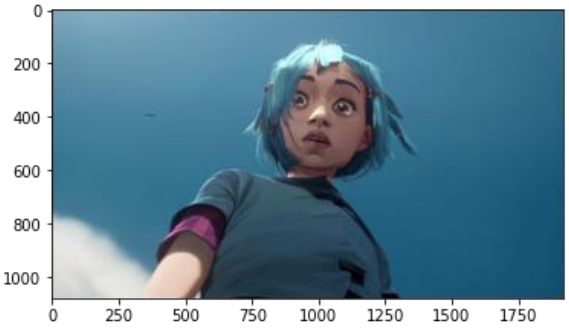
- Citra 1-bit
  Citra 1-bit biasa disebut citra biner. Pada kategori ini, citra digital direpresentasikan oleh satu bit saja sehingga citra biner hanya memiliki dua kemungkinan nilai *pixel*. *Pixel* bernilai 0 mempresentasikan warna hitam sedangkan *pixel* bernilai 1 atau 255 mempresentasikan warna putih.
- Citra 8-bit
  Citra 8-bit disebut juga dengan citra *grayscale*. Setiap *pixel* pada citra *grayscale* direpresentasikan oleh sebuah kanal bilangan 0 sampai 255 yang menyatakan nilai keabuan. *Pixel* dengan nilai 0 memiliki nilai keabuan paling gelap sedangkan *pixel* dengan nilai 255 memiliki nilai keabuan paling terang.
  Citra *grayscale* dapat dikonversi menjadi citra biner dengan cara mengganti nilai *pixel* yang bernilai kurang

dari 127 menjadi 0 dan mengganti nilai *pixel* yang bernilai lebih dari 127 menjadi 255.

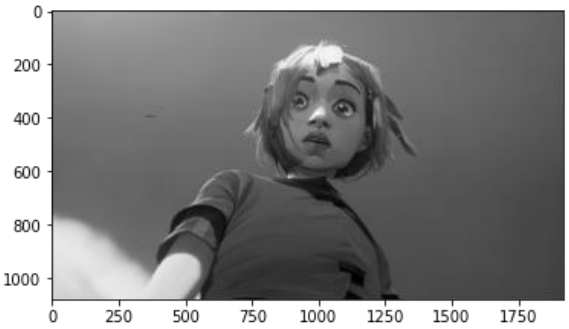
- Citra 24-bit
  Dibandingkan dengan citra 1-bit dan 8-bit yang hanya berwarna hitam, putih, atau abu-abu, citra 24-bit memiliki warna yang beragam. Oleh karena itu, citra 24-bit disebut juga citra berwarna. Warna pada citra 24-bit dimungkinkan karena terdapat 3 kanal bernilai 8-bit pada setiap *pixel*nya. Ketiga kanal tersebut masing-masing mempresentasikan konsentrasi warna *Red*, *Green*, dan *Blue*. Besar konsentrasi warna untuk setiap kanal *pixel* adalah 0 sampai 255.
  Citra berwarna dapat dikonversikan menjadi citra *grayscale* dengan rumus pada persamaan 1:

$$x = 0.299R + 0.587G + 0.114B \quad (1)$$

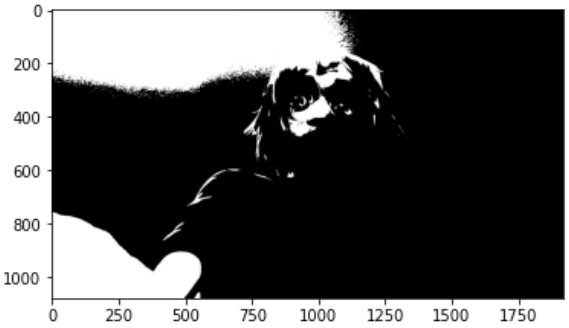
Dengan x merupakan nilai *pixel* citra 8-bit hasil konversi dan R, G, dan B masing-masing merupakan konsentrasi warna *Red*, *Green*, dan *Blue* citra 24-bit.
Citra pada gambar 2.2. merupakan konversi citra *grayscale* 8-bit dari citra berwarna 24-bit pada gambar 2.1. sedangkan gambar 2.3. merupakan konversi citra biner 1-bit dari citra *grayscale* 8-bit pada gambar 2.2.



Gambar 2.1. Citra 24-bit (Sumber: Netflix Arcane)



Gambar 2.2. Citra 8-bit

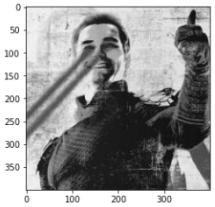


Gambar 2.3. Citra 1-bit

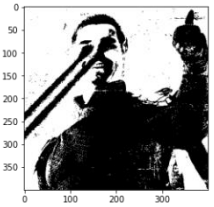
B. Bitplane Citra Digital

Bitplane atau bidang bit dari citra digital merupakan himpunan bit yang terletak pada urutan tertentu dalam *byte* masing-masing. Setiap *byte* tersusun atas bit-bit yang paling berarti (*Most Significant Bit* atau MSB), yaitu bit paling kiri, dan bit-bit yang kurang berarti (*Less Significant Bit* atau LSB).

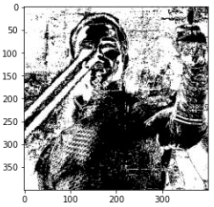
Jika setiap bit dari MSB ke LSB diplot menjadi sebuah citra *bitplane* maka akan diperoleh delapan citra.



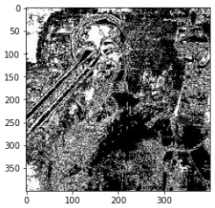
Gambar 2.4. gambar original



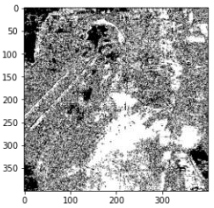
Gambar 2.5. bitplane 7



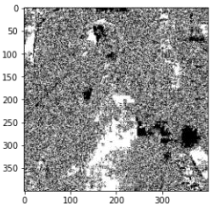
Gambar 2.5. bitplane 6



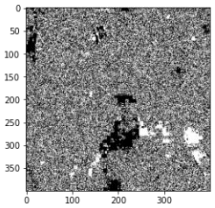
Gambar 2.6. bitplane 5



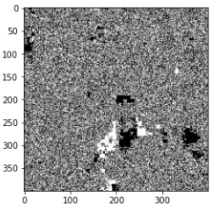
Gambar 2.7. bitplane 4



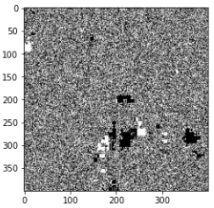
Gambar 2.8. bitplane 3



Gambar 2.9. bitplane 2



Gambar 2.10. bitplane 1



Gambar 2.11. bitplane 0

(Sumber : Amazon The Boys)

C. Steganografi

Steganografi berasal dari Bahasa Yunani *steganos*, yang berarti tersembunyi, dan *graphien*, yang berarti tulisan. Steganografi merupakan ilmu atau teknik menyembunyikan pesan rahasia (*information hiding*) dengan suatu cara sedemikian sehingga keberadaan pesan tersebut tidak terdeteksi. Berbeda dengan kriptografi yang menyembunyikan isi pesan, steganografi cenderung menyembunyikan keberadaan pesan dari pihak lain.

Steganografi digital adalah penyembunyian pesan digital di dalam dokumen digital lainnya. Pesan ataupun dokumen digital yang dimaksud dapat berupa teks, gambar, audio, maupun video.

D. Terminologi Steganografi Digital

Terdapat beberapa terminologi dasar dalam Steganografi Digital. Berikut ini beberapa terminologi dasar yang perlu diketahui: [1]

- *Embedded message* adalah pesan yang ingin disembunyikan. Pesan ini dapat berupa teks, gambar, audio, maupun video.

- *Cover-object* adalah media digital yang akan digunakan untuk menyembunyikan *embedded message*. Media digital yang digunakan dapat berupa teks, gambar, audio, maupun video.
- *Stego-object* adalah media digital yang telah disembunyikan *embedded message* di dalamnya.
- *Stego-key* adalah kunci yang digunakan untuk menyisipkan dan mengekstraksi *embedded message* dari *stego-object*.

E. Kriteria Steganografi

Steganografi yang berkualitas harus memenuhi beberapa kriteria. Berikut ini beberapa kriteria Steganografi yang berkualitas: [1]

- *Imperceptible* yaitu pesan tersembunyi pada Steganografi yang berkualitas tidak dapat dipersepsi atau dirasakan secara visual atau secara audial.
- *Fidelity* yaitu derajat kesamaan antara *cover-object* dan *stego-object*. *Stego-object* hasil Steganografi berkualitas tidak boleh jauh berubah dari *cover-object*nya.
- *Recovery* yaitu *embedded message* dalam *stego-object* harus dapat diekstraksi kembali.
- *Capacity* yaitu Steganografi berkualitas harus memiliki kapasitas penyimpanan *embedded message* sebesar mungkin.
- *Robustness* yaitu *stego-object* hasil Steganografi dapat bertahan terhadap serangan. Akan tetapi, *robustness* pada Steganografi bukanlah kriteria yang dipentingkan karena sifat Steganografi itu sendiri yang cenderung menyembunyikan pesan ketimbang melindungi pesan.

F. Metode Steganografi Least Significant Bit

Metode Steganografi *Least Significant Bit* (LSB) merupakan metode steganografi sederhana yang memanfaatkan *bitplane* 0 atau *bitplane* LSB dari suatu citra. Dalam metode ini, bit LSB akan diubah dengan bit pesan. Bidang bit 0 dipilih karena sifatnya yang terlihat seperti citra acak (lihat Gambar 2.11). Perubahan nilai bit pada bidang ini tidak akan mengubah persepsi citra secara keseluruhan [1].

Terdapat beberapa variasi metode steganografi LSB. Varian pertama adalah LSB *sequential* dimana pesan disisipkan secara berurutan atau *sequential*. Kedua, terdapat metode LSB acak dimana pesan disisipkan pada *pixel* yang acak di dalam media digital, hal ini bertujuan agar pesan dapat tersimpan dengan lebih tersembunyi. Variasi selanjutnya adalah m-bit LSB, dimana digunakan lebih dari satu bit LSB untuk menyimpan lebih banyak pesan.

G. Steganalisis

Steganalisis adalah teknik atau ilmu yang mempelajari pendeteksian adanya pesan tersembunyi pada suatu media. Steganalisis bertujuan untuk menentukan apakah sebuah media mengandung pesan tersembunyi. Steganalisis juga dapat digunakan untuk menentukan panjang pesan tersembunyi [1].

Terdapat beberapa metode steganalisis. Metode steganalisis *Visual Attack* atau serangan berbasis visual memanfaatkan indra penglihatan untuk menginspeksi kerusakan gambar akibat penyisipan pesan. Metode ini hanya dapat digunakan pada media berbasis citra. Salah satu contoh metode steganalisis *Visual Attack* adalah *Enhanced LSB*. Teknik *Enhanced LSB* dilakukan dengan cara mengubah semua bit dalam *byte* menjadi bit LSB.

Metode steganalisis lainnya adalah *Statistical Attack* atau serangan berbasis statistik yang menggunakan analisis matematis untuk menemukan perbedaan antara *cover-object* dengan *stego-object*. Metode ini didasari fakta bahwa penyisipan pesan akan mengubah media cover sehingga *embedded message* dapat dideteksi secara statistik. Contoh metode steganalisis *Statistic Attack* yaitu *histogram analysis, Chi-square analysis*, dan *sample pair analysis*.

H. Pohon

Pohon atau *tree* adalah graf tak-berarah terhubung yang tidak memiliki sirkuit. Pohon berakar atau *rooted tree* adalah pohon yang satu buah simpulnya diperlakukan sebagai akar (*root*) dan sisi-sisinya diberi arah sehingga terbentuk graf berarah. Pohon berakar yang urutan anak-anaknya penting disebut pohon terurut (*Ordered Tree*). Pohon terurut yang setiap simpulnya memiliki paling banyak dua buah anak dinamakan pohon biner (*Binary Tree*) [3].

Gambar 2.12. 3 buah pohon biner
(Sumber: : Matematika Diskrit Jilid 3 2010 (Dr. Rinaldi Munir)

I. Huffman Coding

Huffman Coding adalah salah satu algortima kompresi data sederhana bersifat *lossless* yang mengubah data menjadi kode prefiks optimal. Kompresi data yang bersifat *lossless* adalah kompresi data yang bisa dipulihkan menjadi data semula. Kode prefiks merupakan himpunan kode sedemikian sehingga tidak ada anggota kumpulan yang merupakan awalan dari anggota yang lain [3]. *Huffman Coding* menkonversi data menjadi kode prefiks agar tidak menimbulkan ambiguitas ketika dilakukan proses pemulihan (*decoding*) dari kode hasil kompresi kembali ke data awal.

Algoritma *Huffman Coding* memperhatikan frekuensi kemunculan simbol-simbol dalam data untuk melakukan kompresi. Simbol-simbol dengan frekuensi kemunculan tinggi akan dikonversi menjadi rangkaian kode bit yang lebih pendek daripada simbol-simbol dengan frekuensi kemunculan rendah. Proses konversi *Huffman Coding* menggunakan konsep aplikasi pohon biner.[3]

Langkah-langkah pembentukan Kode Huffman adalah sebagai berikut:

1) Pilih dua simbol dengan frekuensi kemunculan paling kecil, contoh X dan Y.
2) Kombinasikan kedua simbol sebagai *parent* dari simbol X dan Y menjadi simbol XY dengan frekuensi kemunculannya adalah jumlah kemunculan kedua anaknya.

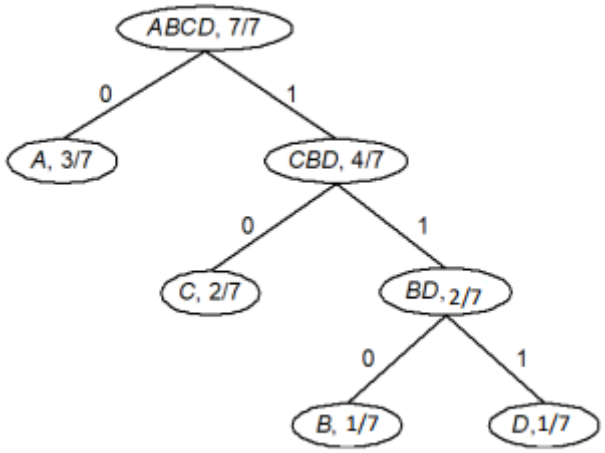
3) Ulangi langkah 1 dan 2 hingga semua simbol habis dan terbentuk *root* dari pohon.
4) Beri label secara konsisten sisi kiri dengan 0 dan sisi kanan dengan 1.
5) Lintasan dari akar ke daun berisi sisi-sisi pohon yang deretan labelnya menyatakan Kode Huffman untuk simbol daun tersebut.

Sebagai contoh untuk mendapatkan Kode Huffman dari string *"ABACCDA"* pertama-tama tentukan frekuensi kemunculan untuk setiap simbol.

Simbol	Frekuensi	Simbol	Frekuensi
A	3	*C*	2
B	1	*D*	1

Tabel 2.1. Frekuensi kemunculan simbol pada string "ABACCDA"

Dengan mengikuti langkah pembentukan Kode Huffman diatas, dibentuk pohon biner Kode Huffman sebagai berikut:



Gambar 2.13. pohon biner hasil Huffman Coding string "ABACCDA"
(Sumber: : Matematika Diskrit Jilid 3 2010 (Dr. Rinaldi Munir)

Dari proses tersebut didapatkan Kode Huffman untuk setiap karakter yaitu *'A'* adalah 0, *'B'* adalah 110, *'C'* adalah 10, dan *'D'* adalah 111. Sehingga *encoding* string *"ABACCDA"* menggunakan *Huffman Coding* menghasilkan "0110010101110". Rangkaian hasil Kode Huffman yang memiliki panjang 13 bit ini jauh lebih pendek dibandingkan rangkaian bit ASCII dari string tersebut yaitu 7 x 8 = 56 bit.

III. PEMBAHASAN

A. Langkah Steganografi

Pada penelitian ini, digunakan metode steganografi berbasis *Least Significant Bit (LSB)* varian *sequential*. Sebelum dilakukan penyisipan *embedded message* ke dalam *cover-object*, alih-alih merepresentasikan *embedded message* sebagai rangkain bit ASCII, *embedded message* akan direpresentasikan sebagai rangkaian bit Kode Huffman sebagai upaya optimasi kapasitas steganografi.

Langkah yang dilakukan untuk pemyisipkan *embedded message* ke dalam *cover*-object dengan kompresi *Huffman Coding* pada penelitian ini adalah sebagai berikut:

1) Menghitung kekerapan kemunculan setiap simbol di dalam *embedded message* lalu membuat Kode Huffman untuk setiap simbol.
2) Mengubah *embedded message* menjadi rangkaian bit Kode Huffman, lalu menambahkan 32-bit panjang rangkaian bit di depan rangkaian sebagai penanda.
3) Mengubah gambar *cover-object* menjadi matriks *pixel*.
4) Mengubah matriks *pixel* gambar menjadi array gambar satu dimensi agar mempermudah pengolahan.
5) Mengubah *least significant bit* pada n elemen awal array gambar sesuai dengan rangkaian bit pesan, dengan n adalah 32 ditambah dengan panjang rangkaian bit pesan.
6) Mengubah array gambar satu dimensi yang sudah disisipkan *embedded message* menjadi matriks *pixel*.
7) Mengubah matriks *pixel* yang sudah disisipkan *embedded message* menjadi gambar.
8) Gambar yang sudah disisipkan *embedded message* merupakan *stego-object*.

Langkah yang dilakukan untuk mengekstraksi *embedded message* dari *stego-object* pada penelitian ini adalah sebagai berikut:

1) Mengubah gambar *stego-object* menjadi matriks *pixel.*
2) Mengubah matriks *pixel* gambar menjadi array gambar satu dimensi agar mempermudah pengolahan.
3) Mengambil LSB dari 32 elemen pertama array gambar sebagai n, yaitu panjang *embedded message*.
4) Mengambil LSB sepanjang n dari elemen ke 32 array gambar sebagai rangkaian bit *embedded message*.
5) *Decode* rangkaian bit *embedded message* menggunakan Kode Huffman yang sudah dimiliki.

B. Konstruksi Kode Huffman

Penelitian ini menggunakan lirik lagu Bohemian Rhapsody oleh Queen sebagai *embedded message* (lirik lagu diambil dari youtube/BohemianRhapsody pada 10 Desember 2022). Tabel frekuensi kemunculan karakter pada *embedded message* ditampilkan pada tabel 3.1. dengan kolom *c* berisi karakter dan kolom *freq* berisi frekuensi kemunculan karakter.

c	***freq***	***c***	***freq***	***c***	***freq***
\n	63	I	14	i	83
\s	333	J	1	j	7
!	3	L	1	k	7
'	14	M	4	l	96
(	2	N	7	m	72
)	2	O	4	n	82
,	27	P	2	o	148
-	16	S	5	p	12
.	3	T	2	r	52
?	2	W	3	s	65
A	4	a	122	t	118
B	8	b	21	u	40
C	2	c	19	v	15
D	1	d	38	w	24
E	1	e	155	y	60

F	1	f	18	z	1
G	7	g	43		
H	1	h	56		

Tabel 3.1. Frekuensi setiap karakter pada embedded message

Selanjutnya dibentuk Kode Huffman untuk setiap karakter yang muncul pada *embedded message* menggunakan konsep pohon. Kode dalam bahasa *python* berikut ini merupakan algoritma inti pemrosesan string menjadi *dictionary* berisi karakter dan Kode Huffmannya.

```
# abstract data type node
class node:
  def __init__(self, freq, symbol, left=None,
right=None):
      self.symbol = symbol
      self.freq = freq
      self.left = left
      self.right = right
      self.huff = ''
  def __lt__(self, nxt):
      return self.freq < nxt.freq


# recursive function to get huffman codes
def nodesToDict(node, value=''):
  global huffmanDict
  newValue = value + str(node.huff)
  if not node.left and not node.right:
      huffmanDict[node.symbol] = newValue
  if node.left: nodesToDict(node.left,
newValue)
  if node.right: nodesToDict(node.right,
newValue)


# take string text and return its huffman codes
def textToHuffmanDict(text):
    global huffmanDict
    huffmanDict  = {}
    nodes = []
    frequency =
collections.Counter(text).items()

    for x in range(len(frequency)):
        heapq.heappush(nodes,
node(frequency[x][1], frequency[x][0]))

    while len(nodes) > 1:
        left = heapq.heappop(nodes)
        right = heapq.heappop(nodes)
        left.huff = 0
        right.huff = 1
        newNode = node(left.freq+right.freq,
left.symbol+right.symbol, left, right)
        heapq.heappush(nodes, newNode)

    nodesToDict(nodes[0])
    return huffmanDict
```

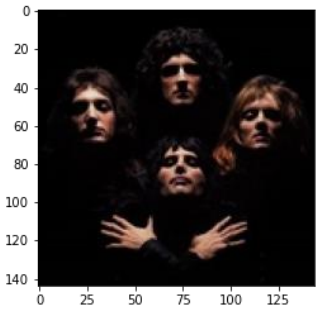
Kode Huffman hasil pemrosesan kemudian ditabulasikan pada tabel 3.2. dengan kolom *c* berisi karakter dan kolom *Huffman Code* berisi Kode Huffman dari karakter.

c	***Huffman Code***	***c***	***Huffman Code***
\n	10100	T	1011000100
\s	00	W	011000010
!	011000000	a	1001
'	0111000	b	1111101
(	0110000111	c	1111100
)	1011010010	d	111000
,	011001	e	1101
-	1011001	f	1011011
.	011000001	g	111111
?	1011000110	h	01101
A	101101000	i	11110
B	10110000	j	01110011
C	1011010011	k	01110100
D	10110101000	l	0100
E	10110101001	m	10111
F	10110001111	n	11101
G	01110101	o	1100
H	0110000110	p	0101001
I	0110001	r	01011
J	10110101010	s	10101
L	10110101011	t	1000
M	101101011	u	111001
N	01110010	v	0111011
O	01010000	w	010101
P	1011000101	y	01111
S	01010001	z	10110001110

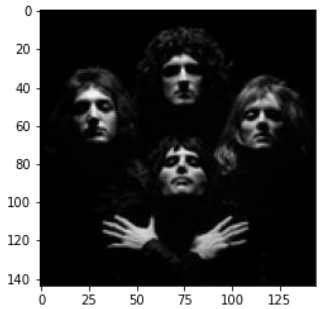
Tabel 3.2. Kode Huffman setiap karakter pada embedded message

C. Proses Steganografi

Sebagai *cover-object*, digunakan citra berdimensi 144x144 *pixel* RGB dan *grayscale*. *Cover-object* yang digunakan ditunjukkan gambar 3.1. dan gambar 3.2.



Gambar 3.1. cover-object RGB

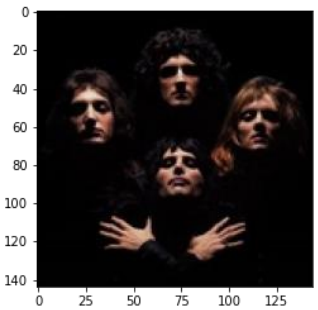


Gambar 3.2. cover-object Grayscale

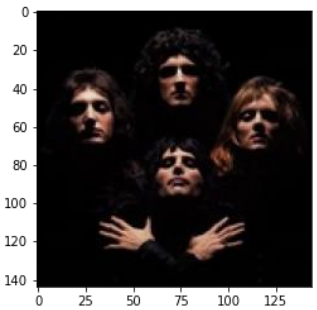
(Sumber : youtube/BohemianRhapsody)

Pertama, *cover-object* berwarna akan disisipkan rangkaian bit *embedded message* 8-bit ASCII yang tidak dikompresi dan rangkaian bit pesan yang sudah dikompresi dengan *Huffman*

Coding. Gambar 3.3. dan gambar 3.4. merupakan citra hasil penyisipan rangkaian 8-bit ASCII dan rangkaian bit kompresi *Huffman Coding* dari *embedded message* terhadap citra *cover-object* berwarna.

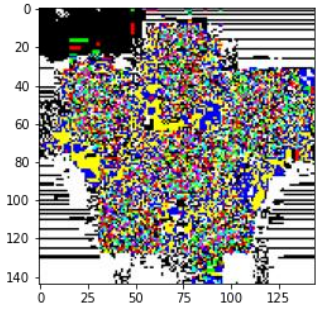


Gambar 3.3. stego-object RGB tanpa kompresi pesan

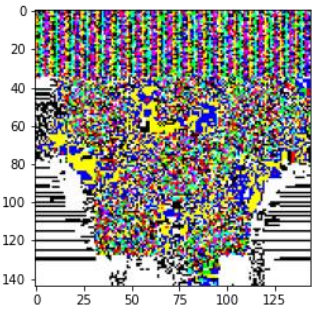


Gambar 3.4. stego-object RGB dengan kompresi pesan

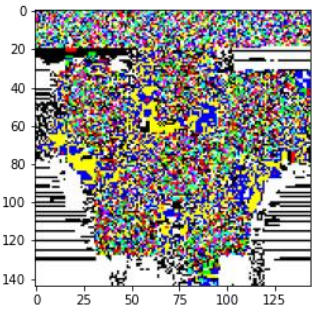
Selanjutnya dilakukan steganalisis metode *Enhanced LSB* terhadap kedua *stego-object* sebagai visualisasi perbandingan banyaknya pesan tersembunyi pada setiap citra. Gambar 3.5. merupakan hasil steganalisis terhadap *cover-object* berwarna yang digunakan, sedangkan gambar 3.6. dan gambar 3.7. merupakan citra hasil steganalisis *stego-object* tanpa kompresi pesan dan dengan kompresi pesan.



Gambar 3.5. citra steganalisis terhadap cover-object

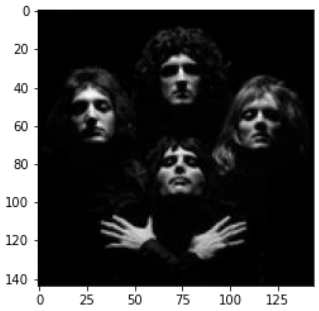


Gambar 3.6. citra steganalisis stego-object tanpa kompresi pesan

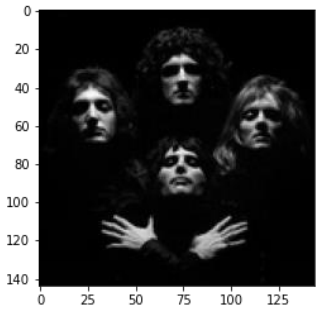


Gambar 3.7. citra steganalisis stego-object dengan kompresi pesan

Dilakukan hal yang sama terhadap *cover-object Grayscale*. Citra disisipkan *embedded message* berupa rangkaian 8-bit ASCII yang tidak dikompresi dan rangkaian bit pesan yang sudah dikompresi dengan *Huffman Coding*. Gambar 3.8. dan gambar 3.9. merupakan citra hasil penyisipan rangkaian 8-bit ASCII dan rangkaian bit kompresi *Huffman Coding* dari *embedded message* pada citra *cover-object* 8-bit.

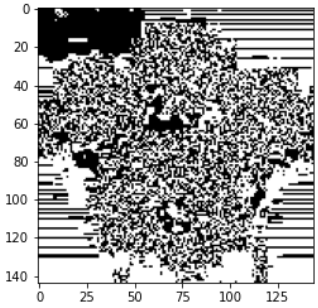


Gambar 3.8. stego-object Grayscale tanpa kompresi pesan

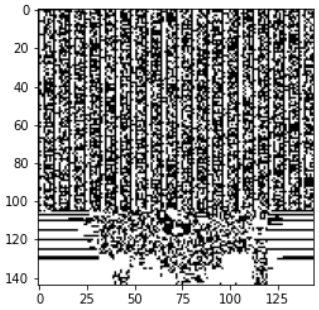


Gambar 3.9. stego-object Grayscale dengan kompresi pesan

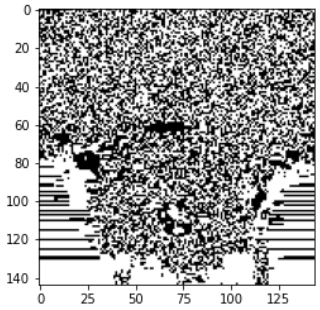
Kedua *stego-object Grayscale* dan *cover-object Grayscale* kemudian disteganalisis dengan metode *visual attack Enhanced LSB* sebagai visualisasi perbandingan banyaknya pesan tersembunyi pada setiap citra. Gambar 3.10. merupakan hasil steganalisis terhadap *cover-object Grayscale* yang digunakan, sedangkan gambar 3.11. dan gambar 3.12. merupakan citra hasil steganalisis *stego-object Grayscale* tanpa kompresi pesan dan dengan kompresi pesan.



Gambar 3.10. citra steganalisis terhadap cover-object Grayscale



Gambar 3.11. citra steganalisis stego-object Grayscale tanpa kompresi pesan



Gambar 3.12. citra steganalisis stego-object Grayscale dengan kompresi pesan

D. Analisis Kapasitas Steganografi

Diperoleh persamaan nilai kompresi antara rangkaian 8-bit ASCII *embedded message* dengan rangkaian Kode Huffman *embedded message* pada persamaan (4).

$$ASCIILength = 8 \times \sum freq_c \quad (2)$$

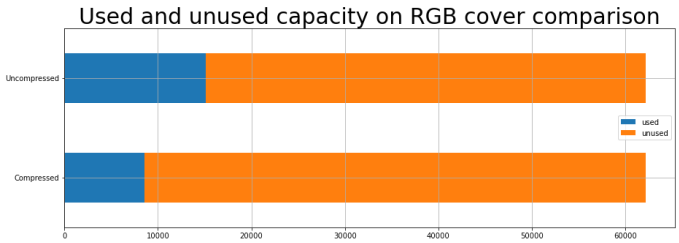
$$EncodedLength = \sum freq_c \times len_c \quad (3)$$

$$compression\ rate = \frac{ASCIILength - EncodedLength}{ASCIILength} \quad (4)$$

Dengan $freq_c$ merupakan frekuensi kemunculan karakter c dan len_c merupakan panjang Kode Huffman untuk karakter c.

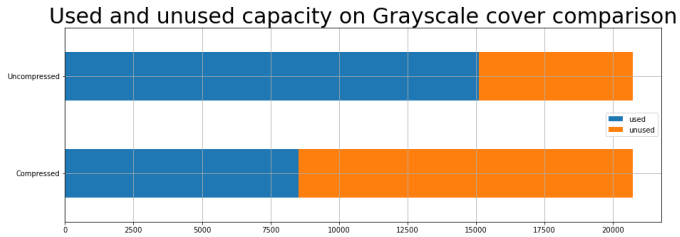
Berdasarkan tabel 3.1. dan tabel 3.2. serta persamaan (2), (3), dan (4) didapat panjang 8-bit rangkaian pesan tersembunyi $ASCIILength$ sebesar 15096, panjang rangkaian pesan yang sudah dikompresi $EncodedLength$ sebesar 8531, dan besar nilai kompresi $compression\ rate$ yaitu 0,4349 atau sama dengan 43,49%. Nilai kompresi tersebut berarti kompresi menggunakan *Huffman Coding* sangat efektif mengingat kapasitas pesan tersembunyi dalam steganografi sangat terbatas.

Pada contoh pertama, *cover-object* RGB berukuran 144x144 *pixel* memiliki kapasitas steganografi LSB sebesar banyak bit di dalam citra, yaitu $144 \times 144 \times 3 = 62208$. Penyisipan rangkaian 8-bit ASCII *embedded message* tanpa kompresi akan mengisi kapasitas *cover* sebesar 24,27%. Sedangkan penyisipan rangkaian bit Kode Huffman pesan akan mengisi kapasitas *cover* sebesar 13,71%. Gambar 3.13 menunjukkan visualisasi perbandingan kapasitas *cover-object* berwarna pada pesan tanpa kompresi dan pesan dengan kompresi.



Gambar 3.13. grafik perbandingan kapasitas cover-object RGB

Pada contoh selanjutnya, *cover-object Grayscale* berukuran 144x144 *pixel* memiliki kapasitas steganografi LSB sebesar banyak bit di dalam citra, yaitu $144 \times 144 = 20736$. Penyisipan rangkaian 8-bit ASCII *embedded message* tanpa kompresi akan mengisi kapasitas *cover* sebesar 72,80%. Sedangkan penyisipan rangkaian bit Kode Huffman pesan akan mengisi kapasitas *cover* sebesar 41,14%. Gambar 3.14 menunjukkan visualisasi perbandingan kapasitas *cover-object Grayscale* pada pesan tanpa kompresi dan pesan dengan kompresi.



Gambar 3.14. grafik perbandingan kapasitas cover-object Grayscale

IV. KESIMPULAN

Penggunaan teknik kompresi pesan *Huffman Coding* sebagai upaya optimasi kapasitas dalam steganografi cenderung efektif dengan tingkat nilai kompresi hingga 43.49%. Akan tetapi, proses pengekstraksian pesan tersembunyi pada steganografi menggunakan *Huffman Coding* membutuhkan Kode Huffman yang harus diketahui terlebih dahulu sebelum proses *decoding* pesan. Walaupun tampak seperti kelemahan, sifat ketergantungan metode ini terhadap Kode Huffman yang sudah ada sebelumnya dapat dimanfaatkan menjadi elemen enkripsi pada steganografi untuk meningkatkan keamanan pesan tersembunyi.

V. LAMPIRAN

Kode program yang digunakan untuk penelitian ini bisa dilihat di github.com/bagas003/Matdis-Steganography.

VI. UCAPAN TERIMA KASIH

Puji syukur terhadap Tuhan Yang Maha Esa karena atas rahmat-Nya, makalah berjudul "Optimasi Kapasitas Steganografi Digital dengan Teknik Kompresi *Huffman Coding*" ini dapat terselesaikan dengan baik. Terimakasih juga disampaikan kepada dosen pengampu mata kuliah IF2120 Matematika Diskrit Dr. Nur Ulfa Maulidevi atas bimbingannya selama perkuliahan.

DAFTAR PUSTAKA

[1] Munir, Rinaldi. 2019. Kriptografi Edisi Kedua. Bandung: Penerbit Informatika

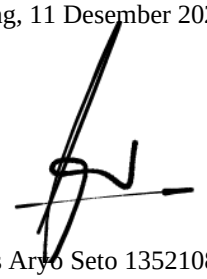
[2] HIdayatullah, Priyanto. 2017. Pengolahan Citra Digital Teori dan Aplikasi Nyata. Bandung: Penerbit Informatika.

[3] Munir, RInaldi. 2020. Matematika Diskrit Revisi Ketujuh. Bandung: Penerbit Informatika

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, 11 Desember 2022



Bagas Aryo Seto 13521081