



**UNIVERSITAS PERTAHANAN RI**

**Analisis Komparasi Algoritma Kompresi Bit pada  
Steganografi Media Citra untuk Keamanan  
Distribusi Informasi**

**AKHTAR ALFARISI SRIYANTO                   320200401002**

**Skripsi yang Ditulis untuk Memenuhi Sebagian Persyaratan  
dalam Mendapatkan Gelar Sarjana**

**FAKULTAS SAINS DAN TEKNOLOGI PERTAHANAN  
PROGRAM STUDI INFORMATIKA  
BOGOR 2024**

## LEMBAR PERSETUJUAN SKRIPSI

Nama : Akhtar Alfarisi Sriyanto  
NIM : 320200401002  
Program Studi : Informatika  
Fakultas : Sains dan Teknologi Pertahanan  
Judul Skripsi : Analisis Komparasi Algoritma Kompresi Bit pada Steganografi Media Citra untuk Keamanan Distribusi Informasi

Dosen Pembimbing 1,

Dr. Ir. Aulia Khamas Heikmakhtiar, S.Kom., M.Eng. Sembada Denrineksa Bimorogo, S.T., M.T.I.  
Penata III/c Penata Muda Tk.I III/b

NIP. 199105082022031002 NIP. 199306052022031002

Tanggal: 25 - 07 - 2024

Dosen Pembimbing 2,

Tanggal: 15 - 07 - 2024

Mengetahui,

Kepala Program Studi  
Informatika,

Adam Mardamsyah, M.Han.  
Kolonel Inf NRP. 11940019450871  
Tanggal: 25 - 07 - 2024

Dekan  
Fakultas Sains Dan Teknologi Pertahanan,

Prof. Dr. Ir. Muhamad Asvial, M.Eng.  
Pembina Utama Muda IV/c  
Tanggal: 25 - 07 - 2024

### LEMBAR PENGESAHAN SKRIPSI

Nama : Akhtar Alfarisi Sriyanto  
 NIM : 320200401002  
 Program Studi : Informatika  
 Fakultas : Sains dan Teknologi Pertahanan  
 Judul Skripsi : Analisis Komparasi Algoritma Kompresi Bit pada Steganografi Media Citra untuk Keamanan Distribusi Informasi

No	Nama	Tanda Tangan	Tanggal
1	Dosen Pembimbing 1:  Dr. Ir. Aulia Khamas Heikhmakhtiar, S.Kom, M.Eng. Penata III/c NIP. 199105082022031002		21/7/2014
2	Dosen Pembimbing 2:  Sembada Denrineksa Bimorogo, S.T., M.T.I. Penata Muda Tk.I III/b NIP. 199306052022031002		23/7/2014
3	Dosen Penguji I:  Aditya Adiprabowo, S.T., M.Sc. Penata Muda Tk.I III/b NIP. 199603212022031005		22/7/2014
4	Dosen Penguji II:  Anindito S. Kom, S.S. MTI., CHFI. PPPK Golongan X NIP. 197805192024211003		23/7/2014
5	Dosen Penguji III:  M. Azhar Prabukusumo, S.Kom., M.Kom PPPK Golongan X NIP. 198608152024211004		22/7/2014

## **PERNYATAAN ORISIONALITAS**

Dengan ini saya menyatakan bahwa dalam skripsi ini tidak terdapat karya atau bagian karya yang pernah diajukan untuk memperoleh gelar kesarjaaan jenjang apapun di suatu Perguruan Tinggi; dan sepanjang sepengetahuan saya juga tidak terdapat istilah, frasa, kalimat, paragraf, subbab atau bab dari karya yang pernah ditulis atau diterbitkan; kecuali yang secara tertulis diajukan dalam naskah ini dan disebutkan dalam Daftar Referensi. Apabila di kemudian hari terbukti bahwa terdapat plagiat dalam skripsi ini, saya bersedia menerima sanksi sesuai ketentuan peraturan/undang-undang yang berlaku.

Bogor, 25 -07 -2014



Akhirat Anansi Sriyanto

**PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH  
UNTUK KEPENTINGAN AKADEMIS**

Sebagai sivitas akademik Universitas Pertahanan Indonesia, saya yang bertanda tangan di bawah ini:

Nama : Akhtar Alfarisi Sriyanto  
NIM : 320200401002  
Program Studi : Informatika  
Fakultas : Sains dan Teknologi Pertahanan  
Jenis karya : Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Pertahanan Indonesia **Hak Bebas Royalti Noneksklusif (Non-exclusive Royalty-Free Right)** atas karya ilmiah saya yang berjudul:

**ANALISIS KOMPARASI ALGORITMA KOMPRESI BIT PADA STEGANOGRAFI MEDIA CITRA UNTUK KEAMANAN DISTRIBUSI INFORMASI**

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Pertahanan Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan Tugas Akhir saya selama tetap mencantumkan nama saya sebagai penulis dan sebagai pemilik Hak Cipta/Karya Intelektual dari skripsi ini.

Demikian pernyataan ini saya buat dengan kesadaran penuh tanpa paksaan dari pihak manapun.

Bogor, Juli 2024

Yang menyatakan



Akhtar Alfarisi Sriyanto

## KATA PENGANTAR

Puji syukur peneliti panjatkan kehadirat Tuhan Yang Maha Esa, karena berkat rahmat dan karunia-Nya penulisan skripsi dengan judul: “Analisis Komparasi Algoritma Kompresi Bit pada Steganografi Media Citra untuk Keamanan Distribusi Informasi” dapat diselesaikan.

Penyusunan skripsi ini ditujukan sebagai salah satu syarat dalam memperoleh gelar Sarjana pada Program Studi Informatika Fakultas Sains Dan Teknologi Pertahanan Universitas Pertahanan RI.

Penyusunan skripsi ini dapat diselesaikan berkat bantuan dan dukungan dari berbagai pihak baik secara langsung maupun tidak langsung. Untuk itu, pada kesempatan ini Kadet Mahasiswa mengucapkan terima kasih kepada:

1. Dr. Ir. Aulia Khamas Heikhmakhtiar, S.Kom., M. Eng, selaku dosen pembimbing 1 yang telah memberikan banyak ide, saran dan masukan kepada peneliti dalam pembuatan skripsi ini.
2. Sembada Denrineksa Bimorogo, S.T., M.T.I., selaku dosen pembimbing 2 yang telah memberikan banyak saran dan masukan kepada peneliti dalam pembuatan skripsi ini.
3. Aditya Adiprabowo, S.T., M.Sc., selaku dewan penguji
4. Anindito S. Kom, S.S. MTI., CHFI., selaku dewan penguji
5. M. Azhar Prabukusumo, S.Kom., M.Kom, selaku dewan penguji
6. Keluarga yang telah memberikan banyak dukungan, motivasi, dan doa selama peneliti melaksanakan skripsi.
7. Rekan-rekan yang banyak membantu, mendukung, menguatkan, serta saling memberikan motivasi dalam menjaga tingkat moril yang tinggi dalam mengerjakan skripsi.

Semoga Tuhan Yang Maha Esa membala kebaikan-kebaikan berbagai pihak atas bantuannya.

Saya menyadari bahwa skripsi ini masih kurang sempurna, oleh karena itu dengan kerendahan hati mengharapkan kritik dan saran yang konstruktif demi kesempurnaan skripsi ini.

Akhirnya, semoga skripsi ini dapat memberikan manfaat terhadap pengembangan ilmu pertahanan dan bermanfaat bagi *stakeholder* terkait dalam upaya meningkatkan keamanan distribusi informasi digital.

Bogor, 15-07-2024



Akhtar Alfarisi Sriyanto

## ABSTRAK

### Analisis Komparasi Algoritma Kompresi Bit pada Steganografi Media Citra untuk Keamanan Distribusi Informasi

Dalam konteks pertahanan nasional, perlindungan data rahasia menjadi krusial akibat seringnya kebocoran data. Kemajuan pesat teknologi informasi memudahkan komunikasi efisien namun juga meningkatkan risiko pencurian data pribadi. Penelitian ini bertujuan mengkaji steganografi sebagai pendekatan yang efektif untuk menyembunyikan pesan sensitif tanpa menimbulkan kecurigaan. Dalam proses penyisipan, kompresi bit dilakukan terhadap data menggunakan salah satu algoritma kompresi yaitu *Huffman Coding*, *Arithmetic Coding*, *Lempel-Ziv-Welch* (LZW), *J-bit Encoding*, dan *Deflate Compression*, lalu dilanjutkan penyisipan ke medium menggunakan algoritma *Least Significant Bit* (LSB). Sebaliknya, pada saat proses ekstraksi, algoritma LSB digunakan untuk mengekstrak data dari medium yang kemudian didekompresi menggunakan algoritma yang sama dengan yang digunakan dalam proses penyisipan. Setelah menerapkan steganografi, pengukuran kinerja dilakukan terhadap algoritma kompresi bit dengan parameter waktu pengerjaan algoritma dan panjang data setelah kompresi. Skema steganografi ini diimplementasikan menggunakan Node.js sebagai aplikasi backend serta React Vite sebagai aplikasi frontend. Metode perbandingan eksponensial digunakan untuk menentukan algoritma dengan rasio kompresi terbaik, yaitu *Deflate Compression* dengan rasio 2.502 untuk 1 paragraf, 2.366 untuk 5 paragraf, dan 2.320 untuk 10 paragraf. Berdasarkan hasil pengukuran waktu *encoding*, algoritma LZW paling cepat untuk 1 paragraf (0.480 ms), *J-bit Encoding* untuk 5 paragraf (1.147 ms), dan *Huffman Coding* untuk 10 paragraf (2.295 ms). Untuk waktu *decoding*, RLE paling cepat untuk 1 paragraf (0.513 ms), LZW untuk 5 paragraf (1.319 ms), dan Huffman Coding untuk 10 paragraf (1.535 ms). Tidak ada algoritma yang secara konsisten unggul, namun LZW menunjukkan performa yang sangat baik. Penelitian ini menyimpulkan bahwa Deflate Compression merupakan algoritma terbaik untuk kompresi data dalam pesan steganografi, dengan rasio kompresi unggul dan waktu pengerjaan yang sebanding. Hasil studi ini dapat digunakan untuk pengiriman pesan rahasia bagi instansi yang memerlukan keamanan dalam distribusi informasi.

**Kata kunci:** Distribusi Informasi, Steganografi, Kompresi Bit, *Run-Length Encoding*, *Huffman Coding*, *Arithmetic Coding*, *Lempel-Ziv-Welch*, *J-bit Encoding*, *Deflate Compression*

## ABSTRACT

### Comparative Analysis of Bit Compression Algorithms in Image Media Steganography for Secure Information Distribution

In the context of national defense, the protection of confidential data has become crucial due to frequent data leaks. The rapid advancement of information technology facilitates efficient communication but also increases the risk of personal data theft. This study aims to examine steganography as an effective approach to hiding sensitive messages without arousing suspicion. In the embedding process, bit compression is performed on the data using one of the compression algorithms, namely Huffman Coding, Arithmetic Coding, Lempel-Ziv-Welch (LZW), J-bit Encoding, and Deflate Compression, followed by embedding into the medium using the Least Significant Bit (LSB) algorithm. Conversely, during the extraction process, the LSB algorithm is used to extract data from the medium, which is then decompressed using the same algorithm used in the embedding process. After implementing steganography, performance measurements were conducted on the bit compression algorithms based on the execution time and the length of the data after compression. This steganography scheme was implemented using Node.js for the backend application and React Vite for the frontend application. The exponential comparison method was used to determine the algorithm with the best compression ratio, which is Deflate Compression with ratios of compression 2.502 for 1 paragraph, 2.366 for 5 paragraphs, and 2.320 for 10 paragraphs. Based on encoding time measurements, the LZW algorithm is the fastest for 1 paragraph (0.480 ms), J-bit Encoding for 5 paragraphs (1.147 ms), and Huffman Coding for 10 paragraphs (2.295 ms). For decoding time, RLE is the fastest for 1 paragraph (0.513 ms), LZW for 5 paragraphs (1.319 ms), and Huffman Coding for 10 paragraphs (1.535 ms). No algorithm consistently outperforms the others, but LZW shows very good performance. This study concludes that Deflate Compression is the best algorithm for data compression in steganographic messages, with a superior compression ratio and comparable execution time. The results of this study can be used for the transmission of secret messages for institutions requiring secure information distribution.

**Keywords:** Information Distribution, Steganograph, Bit Compression, Run-Length Encoding, Huffman Coding, Arithmetic Coding, Lempel-Ziv-Welch, J-bit Encoding, Deflate Compression

## DAFTAR ISI

HALAMAN JUDUL.....	i
LEMBAR PERSETUJUAN SKRIPSI.....	ii
LEMBAR PENGESAHAN SKRIPSI .....	iii
PERNYATAAN ORISIONALITAS .....	iv
PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS .....	v
KATA PENGANTAR.....	vi
ABSTRAK .....	viii
ABSTRACT .....	ix
DAFTAR ISI.....	x
DAFTAR GAMBAR.....	xiv
DAFTAR TABEL.....	xvii
BAB I PENDAHULUAN .....	1
1.1 Latar Belakang .....	1
1.2 Rumusan Masalah .....	3
1.3 Tujuan Penelitian.....	4
1.4 Manfaat Penelitian.....	4
1.5 Batasan Masalah.....	5
BAB II TINJAUAN PUSTAKA .....	7
2.1 Landasan Teori.....	7
2.1.1 Pertahanan Siber .....	7
2.1.2 Keamanan Informasi Militer .....	7
2.1.3 Steganografi .....	8
2.1.4 Algoritma Bit Kompresi .....	12

2.1.5 Algoritma <i>Run-Length Encoding</i> .....	13
2.1.6 Algoritma <i>Huffman Coding</i> .....	13
2.1.7 Algoritma <i>Arithmetic Coding</i> .....	14
2.1.8 Algoritma Lempel-Ziv-Welch (LZW).....	15
2.1.9 Algoritma <i>J-bit Encoding</i> .....	15
2.1.10 Algoritma <i>Deflate Compression</i> .....	15
2.1.11 Algoritma <i>Least Significant Bit</i> (LSB) .....	16
2.1.12 Metode Perbandingan Eksponensial.....	16
2.2 Penelitian Terdahulu .....	17
2.3 Kerangka Berpikir .....	24
<b>BAB III METODE PENELITIAN .....</b>	<b>27</b>
3.1. Desain Penelitian.....	27
3.2. Alur Kerja Algoritma Kompresi Bit.....	28
3.2.1. Alur Kerja Algoritma <i>Run-Length Encoding</i> .....	28
3.2.2. Alur Kerja Algoritma <i>Huffman Coding</i> .....	30
3.2.3. Alur Kerja Algoritma <i>Arithmetic Coding</i> .....	33
3.2.4. Alur Kerja Algoritma <i>Lempel-Ziv-Welch</i> .....	39
3.2.5. Alur Kerja Algoritma <i>J-bit Encoding</i> .....	44
3.2.6. Alur Kerja Algoritma <i>Deflate Compression</i> .....	48
3.3. Rancangan Alur Sistem .....	54
3.3.1 Rancangan Squnce Diagram .....	56
3.3.2 Rancangan Layar .....	58
3.4 Metode Perbandingan.....	61
3.4.1 Komparasi Efisiensi Waktu Kompresi Dengan Grafik Dan Tabel .....	61

3.4.2 Komparasi Persentasi Kompresi Terhadap <i>Plaintext</i> Menggunakan Metode Eksponensial .....	61
3.4.3 Komparasi Persentasi Kompresi terhadap Medium Menggunakan Grafik .....	63
3.5 Tempat dan Waktu Penelitian .....	64
3.5.1 Waktu Penelitian .....	64
3.5.2 Tempat Penelitian .....	66
3.6 Instrumen Penelitian .....	66
3.7 Pengumpulan Data .....	68
3.8 Pengolahan Data .....	71
3.8.1 Pengukuran Waktu Eksekusi.....	71
3.8.2 Kalkulasi Persentase Kompresi .....	71
3.8.3 Penghitungan Jumlah Bit Terhadap Medium .....	72
3.8.4 Analisis Data .....	72
3.8.5 Dokumentasi .....	72
BAB IV HASIL PENELITIAN DAN PEMBAHASAN .....	73
4.1 Masukan Data Pengujian .....	73
4.2 Implementasi Web .....	80
4.2.1 Penyisipan Gambar.....	80
4.2.2 Ekstraksi Gambar.....	107
4.3 Hasil Keluaran Data .....	123
4.3.1 Percobaan Terhadap Variasi 1 paragraf.....	123
4.3.2 Percobaan Terhadap Variasi 5 paragraf.....	127
4.3.3 Percobaan Terhadap Variasi 10 paragraf.....	132
4.4 Analisa Algoritma.....	136

4.5 Penerapan Metode Perbandingan Eksponensial .....	143
4.6 Hasil Analisis Waktu .....	156
<b>BAB V HASIL PENELITIAN DAN PEMBAHASAN .....</b>	<b>161</b>
5.1. Kesimpulan.....	161
5.2. Saran.....	162
<b>DAFTAR PUSTAKA .....</b>	<b>163</b>
<b>LAMPIRAN .....</b>	<b>xvi</b>

## DAFTAR GAMBAR

Gambar 2.1 Kerangka Berpikir.....	26
<i>Gambar 3.1 Desain Penelitian .....</i>	27
Gambar 3.2 Flowchart Run-Length Encoding .....	29
Gambar 3.3 Flowchart Huffman Coding .....	31
Gambar 3.4 Flowchart Arithmetic Coding.....	34
Gambar 3. 5 Flowchart Decoding Arithmetic Coding.....	37
Gambar 3.6 Flowchart Encoding Lempel-Ziv-Welch .....	40
Gambar 3.7 Flowchart Decoding Lempel-Ziv-Welch .....	42
Gambar 3.7 Flowchart Encoding J-bit Encoding .....	44
Gambar 3.9 Flowchart Decoding J-bit Encoding .....	46
Gambar 3.10 Flowchart Encoding Deflate Compression.....	49
Gambar 3.10 Flowchart Encoding Deflate Compression.....	52
Gambar 3.9 Diagram Alur Proses Penyisipan .....	55
Gambar 3.10 Diagram Alur Proses Dekripsi.....	56
Gambar 3.11 Sequence <i>Diagram</i> Proses <i>Embeddding</i> .....	57
Gambar 3.12 <i>Sequence Diagram</i> Proses Ekstraksi .....	58
Gambar 3.13 Rancangan Layar <i>Embedding</i> .....	59
Gambar 3.14 Rancangan Layar <i>Extraction</i> .....	60
<i>Gambar 4.1 Gambar Laman Embed .....</i>	80
Gambar 4.2 <i>Gambar Laman Embed Setelah Klik .....</i>	81
Gambar 4.3 Gambar Unggah RLE.....	82
Gambar 4.4 Gambar Pesan Rahasia RLE .....	83
Gambar 4.5 Gambar Pilih Metode RLE.....	84
Gambar 4.6 Gambar Unduh <i>Stegoimage RLE</i> .....	85
Gambar 4.7 Gambar <i>Stegoimage</i> Terunduh RLE.....	85
Gambar 4.8 Unggah Coverimage <i>Huffman</i> .....	86
Gambar 4.9 Masukan Pesan Rahasia <i>Huffman</i> .....	87
Gambar 4.10 Pilih Metode Kompresi <i>Huffman</i> .....	88
Gambar 4.11 Unduh dan Simpan Pohon <i>Huffman</i> .....	89

Gambar 4.12 Unduh Gambar <i>Huffman</i> .....	90
Gambar 4.13 Unggah Coverimage <i>Arithmetic</i> .....	90
Gambar 4.14 Masukan Pesan Rahasia <i>Arithmetic</i> .....	91
Gambar 4.15 Pilih Metode Kompresi <i>Arithmetic</i> .....	92
Gambar 4.16 Unduh Stegoimage, Pwr, Freq .....	93
Gambar 4.17 Unduh Stegoimage <i>Arithmetic</i> .....	94
Gambar 4.18 Unggah Coverimage LZW.....	95
Gambar 4.19 Masukkan Pesan Rahasia LZW .....	96
Gambar 4.20 Pilih Metode Kompresi LZW .....	97
Gambar 4.21 Unduh <i>Stegoimage</i> .....	98
Gambar 4.22 <i>Stegoimage</i> Terunduh LZW.....	98
Gambar 4.23 Unggah Coverimage JBE.....	99
Gambar 4.24 Masukkan Pesan Rahasia JBE .....	100
Gambar 4.25 Pilih Metode Kompresi JBE .....	101
Gambar 4.26 Unduh <i>Stegoimage</i> JBE .....	102
Gambar 4.27 Stegoimage Terunduh JBE.....	102
Gambar 4.28 Unggah Coverimage <i>Deflate</i> .....	103
Gambar 4.29 Masukkan Pesan Rahasia <i>Deflate</i> .....	104
Gambar 4.30 Pilih Metode Kompresi <i>Deflate</i> .....	105
Gambar 4.31 Unduh <i>Stegoimage</i> dan Pohon <i>Huffman Deflate</i> .....	106
Gambar 4.32 Gambar Terunduh <i>Deflate</i> .....	107
Gambar 4.33 Gambar Laman <i>Extract</i> .....	107
Gambar 4.34 <i>Gambar Laman Extract Setelah Klik</i> .....	108
Gambar 4.35 Gambar Unggah <i>Stegoimgae RLE</i> .....	109
Gambar 4.36 Gambar Pilih Metode RLE.....	110
Gambar 4.37 Baca Pesan RLE.....	110
Gambar 4.38 Unggah Stegoimage <i>Huffman</i> .....	111
Gambar 4.39 Pilih Metode Kompresi <i>Huffman</i> .....	112
Gambar 4.40 Tulis Pohon <i>Huffman</i> .....	112
Gambar 4.41 Membaca Hasil Pesan <i>Huffman</i> .....	113
Gambar 4.42 Unggah Stegoimage <i>Arithmetic</i> .....	114

Gambar 4.43 Pilih Metode Dekompresi <i>Arithmetric</i> .....	114
Gambar 4.44 Tuliskan <i>Arithmetric Freq</i> dan <i>Pwr</i> .....	115
Gambar 4.45 Membaca Hasil Pesan <i>Arithmetric</i> .....	116
Gambar 4.46 Unggah <i>Stegoimage LZW</i> .....	117
Gambar 4.47 Pilih Metode Dekompresi <i>LZW</i> .....	117
Gambar 4.48 Membaca Hasil Pesan <i>LZW</i> .....	118
Gambar 4.49 Unggah <i>Stegoimage JBE</i> .....	119
Gambar 4.50 Pilih Metode Dekompresi <i>JBE</i> .....	119
Gambar 4.51 Membaca Hasil Pesan <i>JBE</i> .....	120
Gambar 4.52 Unggah <i>Stegoimage Deflate</i> .....	121
Gambar 4.53 Pilih Metode Kompresi <i>Deflate</i> .....	121
Gambar 4.54 Tulis Pohon <i>Huffman Deflate</i> .....	122
Gambar 4. 55 Membaca Hasil Pesan .....	122

## DAFTAR TABEL

Tabel 3.1 Jadwal Penelitian .....	65
Tabel 3.2 Persyaratan Sistem .....	67
Tabel 4.1 Performa Algoritma Variasi Teks 1 Paragraf.....	144
Tabel 4.2 Nilai Eksponensial 1 Paragraf.....	147
Tabel 4.3 Performa Algoritma Teks 5 Paragraf .....	148
Tabel 4.4 Nilai Eksponensial 5 Paragraf.....	151
Tabel 4.5 Performa Algoritma Teks 10 Paragraf .....	152
Tabel 4.6 Nilai Eksponensial 10 Paragraf.....	155
Tabel 4.7 Performa Waktu <i>Encoding</i> 1 Paragraf .....	156
Tabel 4.8 Performa Waktu <i>Decoding</i> 1 Paragraf .....	157
Tabel 4.9 Performa Waktu <i>Encoding</i> 5 Paragraf .....	158
Tabel 4.10 Performa Waktu <i>Decoding</i> 5 Paragraf .....	158
Tabel 4.11 Performa Waktu <i>Encoding</i> 10 Paragraf .....	159
Tabel 4.12 Performa Waktu <i>Decoding</i> 10 Paragraf .....	160

# **BAB I**

## **PENDAHULUAN**

### **1.1 Latar Belakang**

Tuntutan terhadap hubungan interaksi antar manusia yang semakin dinamis pada beberapa dekade ke belakang membuat perkembangan teknologi informasi semakin meningkat pesat. Hal ini mempermudah penyampaian dan penerimaan informasi kepada siapa saja. Namun, tidak semua informasi bersifat publik dan dapat diakses oleh semua orang, beberapa di antaranya merupakan informasi pribadi. Kemudahan akses ini juga menciptakan lebih banyak celah yang dapat dimanfaatkan untuk melakukan tindakan kriminal, seperti pencurian data pribadi.

Dalam konteks pertahanan negara, banyak sekali informasi yang dapat diklasifikasikan sebagai informasi yang bersifat rahasia yang mana hal ini tentunya membuat konteks perlindungan data menjadi krusial. Sebagai contoh terdapat berbagai ancaman yang berkaitan dengan sektor militer, seperti yang dilansir dari okezone.com (Utama, 2023) bahwa Kepala Badan Siber dan Sandi Negara (BSSN) Letjen TNI (Purn) Hinsa Siburian mengungkap terdapat 376 dugaan kebocoran data sektor infrastruktur informasi vital, di mana terdapat 250 kasus pada tahun 2022 dan 127 kasus pada tahun 2023. Dalam konteks ini yang ingin kami tekankan bahwa konteks perlindungan terhadap informasi sangatlah krusial bahkan instansi seperti BSSN diperlukan untuk selalu waspada dan tidak boleh lengah terhadap kebocoran atas informasi bahkan berita kebocoran terhadap kebocoran data saja tidak boleh terdengar ke ranah publik untuk menghindari kekacauan.

Terdapat dua pendekatan yang dapat dilakukan untuk melakukan pengamanan informasi dalam rangka menjaga integritas dan kerahasiaan suatu data yaitu dengan dilakukannya kriptografi terhadap pesan atau dilakukan steganografi pada pesan yang akan dikirim. Steganografi dan kriptografi keduanya merupakan teknik yang digunakan untuk keamanan

informasi, tetapi keduanya memiliki pendekatan yang berbeda. Steganografi melibatkan penyembunyian data dalam media pembawa, seperti gambar atau suara, sehingga tidak terlihat secara kasat mata. Steganografi dapat menjadi pelindung tambahan dari upaya pengintaian dan manipulasi data dengan cara menyematkan data rahasia ke dalam berbagai media yang berbeda seperti gambar atau audio (Kurniawan, 2017). Selain menyamarkan intensi untuk berkomunikasi, steganografi juga memiliki kelebihan dalam transparansi persepsi, dimana pesan yang tersembunyi tidak terlihat secara kasat mata. Dengan ditambahkannya kompresi bit, makna sesungguhnya dari pesan yang disisipkan dapat diubah, memberikan keamanan tambahan seperti halnya kriptografi tetapi dengan tingkat transparansi yang lebih tinggi (Sihotang, 2017). Di sisi lain, kriptografi melibatkan konversi teks biasa menjadi teks tersandi menggunakan algoritma enkripsi. Steganografi berfokus pada penyembunyian keberadaan pesan, sedangkan kriptografi berfokus pada membuat pesan tidak dapat dipahami oleh pihak yang tidak berkepentingan (Hammad et al., 2022). Berdasarkan pernyataan-pernyataan tersebut metode steganografi lebih cocok untuk dapat digunakan dalam menyembunyikan informasi rahasia dengan menyembunyikan pesan sehingga pesan tersebut tidak dapat terlihat secara kasat mata.

Secara sederhana seperti yang ditulis oleh Aruna et al. (2023) proses dari steganografi pada teks digital dengan medium citra digital dapat dilakukan dengan metode *LSB Encoding*. Proses modifikasi terhadap berkas dilakukan dengan membandingkan representasi binari dari berkas dengan pesan. Apabila terdapat perbedaan, metode LSB akan melakukan operasi XOR atas berkas dan pesan tersembunyi. Sebaliknya, apabila tidak ada perbedaan tidak akan dilakukan operasi apapun terhadap berkas. Proses ini akan berulang hingga seluruh pesan rahasia berakhir.

Seperti yang Milosav et al. (2023) jelaskan bahwa salah satu hal yang perlu diperlukan dalam penggunaan steganografi adalah kapasitas penyematkan yang besar sehingga hal ini dapat menjadi suatu

permasalahan yang memerlukan solusi untuk meningkatkan kapasitas penyematan dalam penggunaannya. Kemudian seperti yang sudah Feng et al (2020) lakukan dalam mengurangi ukuran pesan teks bahwa peneliti dapat memaksimalkan pesan yang disisipkan dengan menggunakan kompresi bit untuk mengurangi ukuran teks. Beberapa algoritma kompresi bit yang dapat digunakan yaitu: *Huffman Encoding*, *Run Length Encoding*, *Lempel-Ziv Welch*, dan lain sebagainya. Secara sederhana karakteristik dari beberapa algoritma yang tersebut yaitu: *Huffman* menggunakan pohon *huffman* yang mana mengasosiasikan terhadap nilai yang sering muncul, *Run Length Encoding* bekerja dengan mencari frekuensi nilai yang sering muncul yang kemudian disimpan ke dalam *array* dalam bentuk 5 *bit*, kemudian contoh terakhir yaitu *Lempel-Ziv Welch* yaitu dia menggunakan *dictionary* untuk mencari pola yang terpanjang dan diiterasi hingga selesai.

Oleh karena itu penelitian ini akan membahas mengenai perbandingan implementasi algoritma kompresi bit dalam melakukan kompresi pada steganografi yang bertujuan mencari algoritma yang paling pas, dalam hal ini yang memiliki tingkat kompresi tertinggi serta waktu penggerjaan paling cepat. Dengan melakukan penelitian terhadap komparasi algoritma kompresi bit yang digunakan, hal ini dapat menjadi pertimbangan atas penelitian terkait penggunaan kompresi bit dalam implementasi steganografi.

## 1.2 Rumusan Masalah

Rumusan masalah pada penelitian ini berfokus pada identifikasi dan analisis efektivitas algoritma kompresi bit dalam steganografi pada berkas citra digital. Rumusan masalah dari penelitian ini yaitu:

- a. Bagaimana mengukur efisiensi dari kompresi bit pada implementasi teknik steganografi pada media citra dalam hal ini yaitu waktu, ukuran data medium, serta besaran *input*?
- b. Bagaimana memastikan keberhasilan fungsi terhadap penggunaan teknik steganografi yang dikembangkan?

- c. Bagaimana memastikan keberhasilan integrasi antara fungsi kompresi bit dan fungsi steganografi dalam sebuah aplikasi sederhana?

### **1.3 Tujuan Penelitian**

Berikut adalah tujuan dari dilaksanakan penelitian:

- a. Mengukur efisiensi dari kompresi bit pada implementasi teknik steganografi pada media citra dalam hal ini yaitu waktu, ukuran data medium, serta besaran *input*.
- b. Memastikan keberhasilan fungsi terhadap penggunaan teknik steganografi yang dikembangkan.
- c. Memastikan keberhasilan integrasi antara fungsi kompresi bit dan fungsi steganografi dalam sebuah aplikasi sederhana?

### **1.4 Manfaat Penelitian**

- a. Manfaat Teoritis

Manfaat teoritis dari dilaksanakannya penelitian ini adalah sebagai berikut:

- 1) Menambah khazanah pengetahuan baru dalam penerapan algoritma kompresi bit dalam lingkup steganografi.
- 2) Menjadi landasan untuk penelitian berikutnya dalam pemilihan algoritma yang efisien pada kompresi data di steganografi.

- b. Manfaat Praktis

Manfaat teoritis dari dilaksanakannya penelitian ini adalah sebagai berikut:

- 1) Mengetahui efisiensi algoritma kompresi bit, dalam menyisipkan pesan rahasia ke berkas citra digital.
- 2) Mengetahui skema komunikasi yang aman dengan menggunakan steganografi dan bit kompresi yang efisien sebagai landasannya.

- 3) Mengetahui perbedaan performa dari algoritma kompresi bit yang diuji.
- 4) Mengetahui cara pengimplementasian antarmuka secara sederhana dalam penggunaan algoritma kompresi bit terhadap pesan rahasia yang disisipkan pada citra digital.

## 1.5 Batasan Masalah

Berikut adalah beberapa batasan masalah yang akan menunjukkan fokus dari jalannya penelitian:

a. Jenis Media yang Digunakan

Penelitian ini membatasi pada penggunaan gambar digital sebagai media dalam penyematkan pesan rahasia.

b. Jenis Pesan yang Disematkan

Pesan yang akan disematkan adalah pesan teks dari sebuah buku berjudul “Economists At War” oleh Alan Bollard dan kami akan mengambil contoh paragraf sejumlah 10 paragraf yang dimulai dari paragraf 1 bab 1 hingga paragraf 10 bab 1 untuk menunjukkan bahwa data bukanlah data yang direkayasa dan berintegritas. Jumlah karakter yang ada pada 10 paragraf ini adalah 6905 karakter. Setelah dikonversikan ke dalam bit maka didapatkan sejumlah 55240 bit.

c. Spesifikasi Gambar

Untuk menjaga konsistensi, resolusi gambar yang digunakan dalam penelitian ini menggunakan gambar yang dibuat sendiri dengan dimensi gambar 512 x 512 rgba seperti jpg, jpeg, dan png berdasarkan perhitungan karakter *plaintext* yang digunakan yaitu diperlukan sekitar 55240 bit atau diperlukan medium sejumlah 55240 piksel yang mana dimensi ini merupakan dimensi yang sesuai karena dapat menampung sejumlah 262144 piksel dengan menyisakan 206904 piksel apabila

terdapat kompresi yang benar-benar tidak memberikan hasil yang baik dan malah memberikan *redundancy* lebih dari 2 kali besar *plaintext*. Gambar ini merupakan gambar piksel sederhana yang dapat dibuat dengan *tools* sederhana.

d. Jumlah Algoritma yang Dievaluasi

Evaluasi efisiensi akan difokuskan pada enam algoritma kompresi bit dalam steganografi untuk mempermudah proses analisis yaitu: *Run-Length Encoding*, *Huffman Coding*, *Arithmetic Coding*, *Lempel-Ziv-Welch (LZW)*, *J-bit Encoding*, dan *Deflate Compression*.

e. Parameter Evaluasi

Analisis akan berfokus pada tiga parameter yaitu waktu eksekusi algoritma, besar teks yang dapat dikompresi, serta kapasitas pesan yang dapat disematkan dalam representasi bit.

f. Penerapan

hasil penerapan dari penelitian ini akan dituangkan dalam bentuk program sederhana yang dapat menyisipkan text rahasia ke dalam media yang kemudian dipilih algoritma kompresi bit apa yang akan digunakan begitu pula dengan mengekstraknya kembali.

g. Bahasa Pemrograman

Bahasa pemrograman yang akan digunakan dalam penelitian ini adalah Javascript. Hal ini dikarenakan Javascript adalah bahasa yang memiliki kegunaan yang sangat luas sekali serta memiliki keterbacaan yang cukup sederhana sehingga ketika dalam penggerjaan tidak diperlukan perhatian khusus dalam pembacaan *syntax* dan lebih fokus dalam mengolah skema penggerjaan algoritma.

## **BAB II**

### **TINJAUAN PUSTAKA**

#### **2.1 Landasan Teori**

##### **2.1.1 Pertahanan Siber**

Menurut Denning pertahanan siber adalah penggunaan alat digital untuk mempertahankan sistem dan jaringan komputer dari serangan siber dan intrusi jahat (Denning & Strawser, 2012). Menurut kamus merriam-webster pertahanan siber adalah langkah atau tindakan yang diambil untuk melindungi sumber daya online (seperti jaringan komputer atau database) terhadap akses atau serangan tidak sah (*Cyber Defense Definition & Meaning - Merriam-Webster*, n.d.). menurut Darko pertahanan siber adalah adalah mekanisme pertahanan jaringan komputer yang mencakup respons terhadap tindakan dan perlindungan infrastruktur penting serta jaminan informasi untuk organisasi, entitas pemerintah, dan jaringan lain yang memungkinkan (Galiniec, 2023). Serangan siber dapat berbentuk pencurian data, perusakan sistem, atau gangguan terhadap ketersediaan layanan. Pertahanan siber sangat penting untuk melindungi aset-aset vital pada suatu negara, seperti infrastruktur kritis, sistem keuangan, dan informasi rahasia.

##### **2.1.2 Keamanan Informasi Militer**

Keamanan informasi militer mengacu pada perlindungan informasi dalam sistem informasi penting organisasi militer terhadap hilangnya kerahasiaan, integritas, atau ketersediaan. Ini melibatkan penerapan kontrol di berbagai dimensi, seperti teknis, administratif, dan fisik, untuk memastikan keunggulan organisasi militer dalam peperangan informasi dan intelijen kompetitif (Parkhomenko-Kutsevil, 2022). Keamanan informasi di militer sangat penting karena sifat sensitif dari informasi yang disimpan di perangkat digital dan komputer, termasuk sistem pertahanan dan kendali rudal. Tindakan khusus diambil untuk melindungi informasi ini dari serangan

pengguna jahat. Keamanan sumber daya informasi militer memerlukan persyaratan dan spesifikasi yang berbeda dibandingkan dengan aplikasi komersial atau bisnis. Hal ini mencakup memastikan perlindungan sumber daya informasi yang penting bagi militer, termasuk database digital yang tidak dapat diganggu gugat dan mencegah pengaruh informasi terhadap pengambilan keputusan oleh lembaga-lembaga negara.

Dengan melihat berbagai pandangan mengenai algoritma kompresi bit dan penggunaan steganografi, Kompresi bit untuk steganografi dapat dikorelasikan dengan keamanan informasi militer dengan menyediakan sarana untuk menyembunyikan informasi rahasia dalam media teks, seperti dokumen atau pesan, dengan cara yang aman dan efisien. Steganografi adalah metode menyembunyikan pesan rahasia di dalam media sampul yang nampak tidak mencurigakan, dan dengan menggunakan teknik kompresi bit, ukuran pesan rahasia dapat dikurangi, sehingga dapat dengan mudah dimasukkan ke dalam *covertext* tanpa menimbulkan kecurigaan. Hal ini meningkatkan keamanan informasi yang dikirimkan, karena keberadaan pesan tersembunyi tidak diketahui oleh penyerang. Selain itu, penggunaan kompresi mengurangi ruang penyimpanan atau kapasitas transmisi yang diperlukan untuk informasi, sehingga lebih efisien untuk transfer aman.

### **2.1.3 Steganografi**

#### **2.1.3.1 Sejarah Steganografi**

Steganografi berasal dari bahasa Yunani yaitu Steganos berarti tersembunyi atau menyembunyikan dan graphein berarti menulis. Secara sederhana steganografi berarti tulisan tersembunyi. Menurut terminologinya steganografi adalah seni dan ilmu menulis pesan tersembunyi dengan suatu metode atau cara sehingga selain si penerima dan pengirim pesan tidak ada orang lain yang tahu atau menyadari bahwa ada suatu pesan rahasia. Steganografi telah digunakan sejak zaman kuno untuk menyembunyikan pesan dari pihak ketiga (Li et al., 2023). Istilah “steganografi” pertama kali dicatat pada tahun 1499 oleh Johannes

Trithemius dalam bukunya tentang kriptografi dan steganografi (Lee, 2021). Teknik steganografi dapat diterapkan pada berbagai jenis data, antara lain gambar, berkas video, dan berkas audio (Dutta & Chakraborty, 2020). Berbeda dengan kriptografi, steganografi tidak hanya melindungi isi pesan tetapi juga menyembunyikan fakta bahwa pesan rahasia sedang dikirim (Kose et al., 2020). Steganografi adalah seni dan ilmu menyembunyikan pesan rahasia dalam berbagai jenis data, sehingga hanya pengirim dan penerima yang menyadari keberadaan pesan tersebut, berbeda dengan kriptografi yang hanya melindungi isi pesan.

Steganografi memiliki sejarah panjang yang dimulai dari Yunani kuno hingga berkembang ke metode modern yang melibatkan data dalam berkas multimedia. Contoh steganografi pertama yang diketahui dapat ditelusuri kembali ke Yunani kuno sepenelitir tahun 440 SM. Penguasa Yunani Histaeus menggunakan bentuk steganografi dengan menato pesan di kulit kepala budak dan menunggu rambut tumbuh kembali sebelum mengirim budak untuk menyampaikan pesan tersebut (Cheddad et al., 2008). Bentuk awal lain dari steganografi melibatkan ukiran pesan pada tablet lilin dan menutupinya dengan lapisan lilin baru. Steganografi terus berkembang seiring berjalannya waktu, dengan Sir Francis Bacon menggunakan variasi jenis huruf untuk menyandikan pesan pada tahun 1600-an (Kehar & Kaur, 2011). Saat ini, teknik steganografi telah berevolusi untuk memasukkan data dalam berkas multimedia, seperti bingkai video, menggunakan wilayah tertentu berdasarkan deteksi warna kulit manusia (Reyzin & Russell, 2004).

Dalam dunia modern, steganografi telah berkembang hingga mencakup steganografi digital, yang melibatkan penyembunyian informasi dalam berkas yang disimpan di komputer. Dengan pesatnya perkembangan jaringan saraf dalam, jaringan steganografi telah muncul sebagai pendekatan yang menjanjikan untuk komunikasi rahasia (Li et al., 2023). Jaringan ini berukuran relatif besar, sehingga menimbulkan kekhawatiran tentang cara menyebarkannya secara diam-diam di saluran publik. Para

peneliti telah mengusulkan skema untuk menyamarkan jaringan steganografi sebagai model pembelajaran mesin biasa, yang memungkinkan terjadinya komunikasi rahasia. Steganografi terus bermanfaat dalam skenario komunikasi modern, melindungi ketidakamanan saluran komunikasi nirkabel dan berkontribusi pada pengembangan protokol keamanan untuk berbagai aplikasi. Oleh karena itu dikembangkanlah cabang ilmu yang mempelajari tentang cara-cara penyembunyian pesan rahasia atau dikenal dengan istilah steganografi.

#### **2.1.3.3 Terminologi dan Konsep Dasar Steganografi**

penulis akan menjelaskan beberapa istilah penting dalam bidang Steganografi yang akan sering penulis gunakan dalam penulisan. Berikut adalah istilah-istilah penting tersebut:

a. *Stegoimage*

*Stegoimage* adalah gambar yang menyembunyikan pesan rahasia melalui teknik steganografi, memastikan informasi yang tersembunyi tidak terlihat oleh orang yang tidak berwenang (Li et al., 2023).

b. *Coverimage*

Seperti yang dijelaskan oleh Baldha (2023) *coverimage* merupakan media citra penampung dari pesan rahasia yang dapat digunakan dalam LSB steganografi.

Namun selain dari steganografi gambar, seperti yang sudah Kose et al., (2020) jelaskan dalam format digital terdapat tipe steganografi lainnya seperti steganografi video, audio, serta GIF.

#### **2.1.3.2 Tujuan Steganografi**

Setidaknya ada tiga tujuan utama dari ilmu steganografi ini yaitu (Milosav et al., 2023):

a. Transparansi Persepsi

Transparansi persepsi dalam steganografi mengacu pada kemampuan untuk menyematkan informasi tersembunyi dalam media sampul (seperti *file* gambar, audio, atau video) tanpa menyebabkan perubahan nyata pada tampilan atau kualitas aslinya, tidak berubah. Mempertahankan transparansi persepsi sangat penting untuk memastikan bahwa modifikasi steganografi tidak terlihat oleh indra manusia. Jika perubahannya terlihat jelas, hal ini dapat menimbulkan kecurigaan dan menggagalkan tujuan komunikasi rahasia.

b. *Robustness* (kekokohan)

Kekokohan dalam steganografi melibatkan kemampuan informasi tersembunyi untuk menahan berbagai jenis transformasi, serangan, atau distorsi yang mungkin terjadi selama transmisi atau penyimpanan media penutup. Metode steganografi yang kuat memastikan bahwa informasi yang tersembunyi tetap utuh dan dapat dipulihkan bahkan ketika ada operasi pemrosesan sinyal umum, kompresi, *noise*, atau serangan yang disengaja. Hal ini meningkatkan keandalan komunikasi rahasia.

c. Kapasitas Penyematan yang Besar

Kapasitas medium berarti jumlah informasi tersembunyi yang dapat dikirimkan oleh *coverimage*. Hal ini dinyatakan dalam jumlah absolut dalam byte atau secara relatif dalam persentase sehubungan dengan ukuran medium itu sendiri. Kapasitas penyematan yang besar mengacu pada jumlah informasi rahasia yang dapat ditanamkan secara efektif ke dalam media penutup tanpa menurunkan kualitas atau menimbulkan kecurigaan secara signifikan. Metode steganografi dengan kapasitas penyematan yang besar memungkinkan transmisi sejumlah besar data tersembunyi.

Hal ini sangat penting terutama dalam skenario di mana sejumlah besar informasi perlu disembunyikan. Namun, mencapai keseimbangan dengan transparansi persepsi sangat penting untuk menghindari deteksi.

#### **2.1.3.4 Algoritma Steganografi**

Terdapat berbagai jenis algoritma kompresi bit yang dapat digunakan dalam steganografi, contohnya adalah algoritma steganografi *Least Significant Bit* (LSB). Algoritma LSB dalam steganografi digunakan untuk menyembunyikan informasi di dalam gambar dengan mengganti bit paling tidak signifikan dari nilai piksel dengan data rahasia. Teknik ini umumnya digunakan karena mengubah bit terkecil tidak mempengaruhi kualitas visual gambar secara signifikan. Metode LSB dapat dikombinasikan dengan teknik lain seperti kriptografi untuk meningkatkan keamanan informasi yang tersebunyi. Misalnya, sebuah penelitian mengusulkan penggunaan urutan acak yang diperoleh dari peta logistik untuk menentukan jumlah bit yang digunakan dalam metode LSB, sehingga menghasilkan gambar tersebunyi yang sulit dibuka dengan mudah (Al Maki et al., 2023). Studi lain meningkatkan algoritma LSB dengan menggunakan urutan yang dihasilkan secara acak untuk menyematkan informasi, menghasilkan kualitas gambar yang lebih baik dan ketahanan yang lebih kuat terhadap serangan (Imanda et al., 2023). Algoritma Least Significant Bit (LSB) digunakan dalam steganografi untuk menyembunyikan informasi dalam gambar dengan sedikit mengurangi kualitas visualnya, dan dapat digabungkan dengan teknik lain seperti kriptografi untuk meningkatkan keamanan.

#### **2.1.4 Algoritma Bit Kompresi**

Algoritma Kompresi Bit adalah teknik yang digunakan untuk mengurangi ukuran data dengan mengganti representasi asli dengan simbol, sehingga mengurangi redundansi statistik (Agrawal et al., 2015). Ini

melibatkan penetapan satu bit (0 atau 1) untuk mengkodekan elemen yang paling sering muncul, sambil menyimpan posisi elemen lainnya. Contoh dari algoritma bit kompresi yaitu *Run-Length Encoding*, *Huffman Coding*, *Arithmetic Coding*.

### **2.1.5 Algoritma *Run-Length Encoding***

*Run-Length Encoding (RLE)* adalah salah satu metode paling sederhana dari kompresi data *lossless* yang menggantikan urutan (*run*) dari elemen yang sama dengan kode elemen tunggal dan jumlah elemen dalam run (panjang urutan simbol). Kode elemen dan panjang proses yang dikodekan terkait disebut paket *RLE* (Mados et al., 2021). Teknik ini memungkinkan pengurangan ukuran data tanpa kehilangan informasi asli.

Sejarah penggunaan RLE dimulai pada tahun 1967, ketika RLE telah digunakan dalam transmisi sinyal televisi. Data yang akan dikodekan dapat berupa aliran data satu dimensi (1D), atau mungkin multidimensi seperti dalam kasus gambar yang merupakan kisi-kisi piksel dua dimensi (2D) atau data volumetrik yang merupakan tiga dimensi - kisi dimensi voxel (3D). Penerapan RLE dalam berbagai format data menunjukkan fleksibilitas dan kegunaannya dalam berbagai konteks.

Dalam langkah-langkah kompresi, pertama-tama, nilai saat ini diperiksa dengan nilai tetangga. Jika nilai saat ini sama dengan nilai tetangga, maka nilai-nilai tersebut digabungkan menjadi satu dan sebuah penghitung (counter) ditambahkan ke dalam nilai tersebut. Selanjutnya, langkah kedua adalah melanjutkan ke nilai berikutnya. Jika nilai saat ini tidak sama dengan nilai tetangga, maka nilai saat ini disimpan dan langkah pertama diulangi kembali. Setelah proses langkah 1 dan 2 selesai, hasil kompresi disimpan (Hardi et al., 2019).

### **2.1.6 Algoritma *Huffman Coding***

Pengkodean Huffman adalah algoritma kompresi data dasar yang algoritma ini memberikan kode pada karakter sedemikian rupa sehingga

panjang kode tergantung pada frekuensi relatif dari karakter yang dimaksud (alias, simbol input). Kode *Huffman* memiliki panjang yang bervariasi dan *prefix-free* (bebas awalan), bebas awalan berarti tidak ada kode yang merupakan awalan dari kode lain. Setiap kode biner bebas awalan dapat divisualisasikan sebagai sebuah pohon biner (disebut pohon Huffman) dengan karakter yang disandikan tersimpan di daunnya (Tian et al., 2021).

Pengkodean *Huffman* adalah mekanisme kompresi yang telah diadaptasi secara luas, yang digunakan oleh para peneliti dan industri untuk mengompresi data probabilistik secara efisien ke dalam kode yang dipersingkat secara optimal untuk tujuan efisiensi penyimpanan dan komunikasi. Secara khusus, dalam kasus perangkat jaringan dengan keterbatasan sumber daya tertentu, pengkodean Huffman dapat memainkan peran penting dengan mengurangi jumlah bit yang digunakan untuk transmisi, penyimpanan, dan pemrosesan data. Di sisi lain, sama pentingnya untuk melindungi privasi perangkat jaringan tersebut selama komunikasi, penyimpanan, pemrosesan, dan analisis data yang dikumpulkan darinya (Hassan et al., 2023).

### 2.1.7 Algoritma *Arithmetic Coding*

Algoritma *Arithmetic Coding* adalah metode kompresi yang menggantikan satu baris simbol *input* dengan angka floating point. Ide dasar dari *Arithmetic Coding* adalah membuat garis peluang dari 0 sampai 1 dan memberikan interval untuk setiap karakter dari teks *input* berdasarkan peluang kemunculannya. Semakin tinggi peluang sebuah karakter, maka semakin besar interval yang akan didapatkan .

Keluaran dari pengkodean aritmatik adalah angka yang lebih kecil dari angka 1 dan lebih besar dari atau sama dengan 0. Angka ini dapat diterjemahkan secara unik untuk menghasilkan deretan simbol yang digunakan untuk menghasilkan angka tersebut. Untuk menghasilkan angka keluaran, setiap simbol yang akan dikodekan diberikan satu set nilai probabilitas (Silitonga & Morina, 2019).

### **2.1.8 Algoritma Lempel-Ziv-Welch (LZW)**

Algoritma kompresi Lempel-Ziv-Welch (LZW) adalah metode kompresi berbasis kamus. Ia bekerja dengan mengganti pola karakter yang berulang dengan kode yang lebih pendek. LZW dikenal dengan rasio kompresinya yang tinggi dan umum digunakan dalam berbagai aplikasi. Ini telah diimplementasikan menggunakan pengkodean VHDL untuk meningkatkan tingkat kompresi dan persentase penghematan ruang (Bille et al., 2023). LZW juga dapat digunakan dalam kombinasi dengan algoritma enkripsi lainnya, seperti Columnar Transposition Cipher dan Caesar Cipher, untuk menjamin kerahasiaan, integritas, dan keamanan data (Mohey et al., 2021).

### **2.1.9 Algoritma J-bit Encoding**

*J-Bit Encoding* (JBE) adalah algoritma kompresi data yang mengurangi ukuran data dengan memanipulasi setiap bit data. Ini membagi data menjadi dua keluaran dan kemudian menggabungkannya kembali menjadi satu keluaran. Algoritma telah dimodifikasi di beberapa makalah untuk menghilangkan simbol nol dan satu dari keluaran pertama, sehingga keluaran pertama berisi data asli (tidak termasuk nol dan satu) dan keluaran kedua berisi nilai dua bit yang menggambarkan posisi nol, , satu, dan byte lainnya. Kinerja berbagai kombinasi algoritma telah dibandingkan, dan rasio kompresi terbaik telah dicapai dengan menggunakan skema tertentu (Naibaho, 2020).

### **2.1.10 Algoritma Deflate Compression**

Algoritma kompresi Deflate merupakan salah satu algoritma kompresi data lossless yang umum digunakan pada berbagai aplikasi. Ia dikenal karena efisiensi dan kemampuannya mencapai rasio kompresi yang tinggi. Berbeda dengan algoritma kompresi lain yang menggunakan beberapa teknik secara bersamaan, Deflate mampu mengompresi data sendiri tanpa memerlukan algoritma tambahan (Dey, 2022). Secara keseluruhan, Deflate adalah algoritma kompresi yang banyak digunakan

yang menawarkan kemampuan kompresi data yang efisien dan efektif di berbagai domain.

### **2.1.11 Algoritma Least Significant Bit (LSB)**

Algoritma *Least Significant Bit* (LSB) adalah teknik yang digunakan dalam steganografi untuk menyembunyikan pesan rahasia di dalam objek penutup, seperti gambar, audio, video, dan teks. Ia bekerja dengan mengganti bit paling tidak signifikan dari data objek sampul dengan bit pesan rahasia, tanpa mempengaruhi kualitas visual atau pendengaran objek sampul secara signifikan. Metode ini banyak digunakan dalam steganografi gambar, dimana pesan rahasia disembunyikan di dalam piksel suatu gambar. Algoritma LSB dapat dikombinasikan dengan teknik enkripsi, seperti AES atau Vigenere Cipher, untuk menambahkan lapisan keamanan ekstra pada pesan tersembunyi (Imanda et al., 2023). Metode LSB juga dapat ditingkatkan dengan menggunakan urutan acak yang diperoleh dari peta logistik untuk menentukan jumlah bit yang digunakan untuk menyembunyikan pesan sehingga lebih sulit untuk dideteksi (Yanti & Budayawan, 2023).

### **2.1.12 Metode Perbandingan Eksponensial**

Metode perbandingan eksponensial adalah teknik yang digunakan untuk menetapkan urutan prioritas dari berbagai alternatif keputusan berdasarkan beberapa kriteria. Rumus dari metode eksponensial adalah sebagai beriku (Hasibuan, 2022):

$$TNi = \sum_{j=1}^m (Vij) \times Bj \quad (2. 1)$$

Keterangan:

$TNi$  = Total nilai alternatif ke - i

$Vij$  = Derajat kepentingan relatif kriteria j pada keputusan i

$Bj$  =

Derajat kepentingan kriteria keputusan yang dinyatakan  
dengan bobot

$m$  = Jumlah kriteria keputusan

$n$  = Jumlah pilihan keputusan

$j = 1,2,3, \dots, m$  = Jumlah kriteria

$i = 1,2,3, \dots, n$  = Jumlah pilihan alternatif

## 2.2 Penelitian Terdahulu

Penelitian yang relevan dengan penelitian ini pernah dilakukan oleh beberapa peneliti terdahulu, antara lain:

Dalam penelitian oleh Djebbar et al., (2012) tentang studi komparasi tentang teknik steganografi audio digital di mana di dalam penelitian ini dibandingkan berbagai macam teknik dan pendekatan steganografi audio digital yang berbeda dalam rangka mengetahui kelebihan dan kekurangan dari tiap-tiap teknik. Setiap teknik bergantung pada intensi dari pengaplikasianya. Misalnya teknik domain temporal, secara umum, bertujuan untuk memaksimalkan kapasitas penyembunyian, lalu metode domain transformasi dengan memanfaatkan properti *masking* untuk membuat kebisingan hasil dari penyematan data tidak terdeksi. Kemudian di sisi lain metode domain terenkripsi memastikan integritas data pada aplikasi waktu nyata. Hasil pengkajian dari penelitian ini disajikan dalam bentuk evaluasi atas kinerja dari tiap-tiap teknik yang ditinjau. Dengan demikian didapatkanlah bahwa dengan keberagaman teknik steganografi audio digital yang ada, keuntungan yang didapatkan dari satu teknik

terhadap teknik lainnya bergantung pada batasan aplikasi yang digunakan dan kebutuhannya terhadap kapasitas, keamanan dari data yang disematkan, serta ketahanan atas serangan yang dihadapi.

Dalam penelitian Kose et al., (2020) yaitu ulasan atas teknik steganografi yang dapat dikelompokkan secara umum berdasarkan jenis media yang digunakan (teks, audio, gambar, video, protokol, dll.) dan jenis teknik yang digunakan, mulai dari modifikasi paling sederhana hingga transformasi kompleks dan pemrosesan. Tujuan dari klasifikasi ini adalah untuk menggambarkan variasi teknik penyembunyian informasi, baik yang sederhana maupun kompleks, serta menghalangi upaya penyerang untuk menemukan pesan yang tersembunyi. Upaya tersebut melibatkan penggunaan bidang-bidang cadangan dan transformasi struktur stego-media agar pesan yang disembunyikan tidak dapat dipulihkan, secara efektif menggagalkan komunikasi yang tidak diinginkan.

Dalam penelitian Albayati & Ali, (2021) pada studi komparasi mengenai steganografi pada media gambar dengan menggunakan metode-metode deteksi tepi. Penelitian ini pendekatan penyamaran informasi dengan menggabungkan penggantian LSB dengan deteksi tepi. Untuk menghindari dampak HVS saat lebih banyak bit tersembunyi dalam piksel, piksel penutup diklasifikasikan menjadi wilayah tepi dan wilayah non-tepi. Piksel yang termasuk dalam wilayah tepi digunakan untuk mentransmisikan lebih banyak bit rahasia. Selain itu, digunakan metode di mana informasi tepi diidentifikasi oleh bit-bit paling penting (MSBs) dari gambar penutup, menghasilkan peningkatan beban dan kualitas optik. Eksperimen membuktikan bahwa skema ini dapat mencapai beban yang lebih tinggi dan kualitas optik yang lebih baik.

Dalam penelitian oleh Jenifer, (2021) tentang studi komparasi atas steganografi video. Tujuan dari penelitian yang dilakukan ini adalah memberikan gambaran umum tentang berbagai teknik steganografi video, meliputi karya terkait, kekuatan steganografi, jenis-jenis steganografi, dan

teknik steganografi video khusus. Di dalamnya juga disoroti analisis perbandingan berbagai teknik steganografi video.

Dalam penelitian oleh Modupe et al., (2021) telah melakukan penelitian terhadap analisis algoritma LSB, MSB dan PVD pada steganografi dengan media gambar. Dalam penelitian ini, dilakukan analisis perbandingan steganografi gambar dengan menggunakan teknik *Least Significant Bit* (LSB), *Most Significant Bit* (MSB), dan *Pixel Value Differencing* (PVD) pada gambar berskala abu-abu dan berwarna. Tiga gambar penutup berbeda digunakan untuk menyembunyikan pesan rahasia. Analisis kinerja perbandingan dari metode LSB, MSB, dan PVD dalam steganografi gambar dilakukan menggunakan peak signal to noise ratio (PSNR), mean square error (MSE), dan structural similarity index (SSIM) sebagai metrik kinerja. Teknik LSB memberikan nilai PSNR dan SSIM yang lebih tinggi daripada metode MSB dan PVD, dengan MSE yang lebih rendah dari kedua teknik lainnya.

Selanjutnya sebagai bahan pertimbangan atas pemilihan algoritma *Run-Length Encoding* yang sebelumnya pernah diteliti oleh Hardi et al., (2019) bahwa algoritma ini dapat memberikan hasil kompresi yang cukup baik. Rata-rata dari hasil uji coba menunjukkan nilai 69.355% pada data yang digunakan. Hal ini memberikan harapan bahwa algoritma ini setidaknya dapat digunakan terutama dalam hal kompresi dengan ukuran yang lebih kecil seperti paragraf yang dimuat dalam teks buku.

Untuk *arithmetic coding* sendiri seperti yang telah Shanmugasundaram dan Lourdusamy (2011) lakukan peforma arithmetic coding dalam teknik kompresi statis *arithmetic coding* memiliki peforma kompresi terbaik dengan rata-rata bit per karakter sepenelitir 5. hal ini menunjukkan performa yang lebih baik bahkan bagi algoritma yang paling dekat mendekati angka 1,15% dari *arithmetic coding*. Algoritma ini juga mengungguli sejauh 35,06% dari algoritma yang dibandingkan lainnya dalam pengukuran bit per karakter. Dalam penelitian ini juga didapatkan

algoritma yang mengurangi redundansi pesan berulang variasi Lempel-Ziv yaitu Lempel-Ziv-Welch yang mendapatkan kompresi

Kemudian untuk *J-Bit Encoding* seperti yang dilakukan oleh Naibaho (2020) bahwa implementasi *J-Bit Encoding* dapat menjadi suatu alternatif dalam proses kompresi yang bersifat *lossless*. Didapati bahwa dalam kompresi suatu file ditemukan kompresi ini dapat memberikan Tingkat kompresi yang cukup tinggi yaitu berada di angka 20%. Pemanfaatan J-Bit Encoding ini yang unik dapat diujikan untuk mencari apakah teknik yang digunakan dapat memberikan perubahan yang signifikan terhadap teknik lainnya.

Terakhir ada seperti yang telah diteliti oleh Gupta et al., (2017) Algoritma *Deflate*, menggunakan kombinasi teknik kompresi *LZ77* dan pengkodean *Huffman*. Algoritma ini dirancang untuk format *file .zip* dan telah digunakan dalam berbagai format *file* lainnya seperti *GZIP* dan *7-zip*. *Deflate* menawarkan keunggulan dalam kecepatan kompresi dan dekompresi yang tinggi, menjadikannya pilihan yang baik untuk aplikasi yang memerlukan pemrosesan data cepat. Meskipun rasio kompresinya lebih rendah dibandingkan dengan algoritma modern lainnya seperti *LZMA* atau *PPMd*, kecepatan pemrosesan *Deflate* yang tinggi membuatnya sangat efisien dalam lingkungan dengan batasan waktu kompresi dan dekompresi yang ketat.

Berdasarkan penelitian terdahulu yang sudah dijelaskan, keenam algoritma tersebut dipilih karena menggunakan pendekatan kompresi yang berbeda serta hasil kompresi yang cukup baik. Peneliti berasumsi bahwa dengan menggunakan pendekatan cara kerja algoritma yang berbeda, kinerja yang dihasilkan akan bervariatif dan lebih komprehensif. Tidak setiap penelitian yang menyertakan perhitungan waktu juga dapat menjadi pertimbangan dalam pemilihan algoritma-algoritma ini sehingga dapat menambah tingkat komprehensi dalam penelitian ini.

Tabel 2. 1 Tabel Penelitian Terdahulu

No.	Paper Judul (Penulis, Tahun Publikasi)	Ringkasan	Research Gap
1.	<i>Comparative study of digital audio steganography techniques</i> Oleh (Djebbar et al., 2012)	Dari hasil pembahasan dalam penelitian ini didapati bahwasannya teknik steganografi bergantung pada intensi penggunaan dan kebutuhannya terhadap kapasitas penyimpanan, dibahas pula mengenai LSB serta kelemahannya terhadap kapasitas penyimpanan	Perlakuan Teknik LSB dilakukan tanpa kompresi sehingga memakan banyak tempat
2.	<i>Review and Test of Steganography Techniques</i> Oleh (Kose et al., 2020)	Pada penelitian dibahas mengenai ulasan berbagai teknik steganografi termasuk di dalamnya adalah steganografi dengan citra digital	Pembahasan dilakukan secara konseptual sehingga dapat dilakukan pengujian secara aplikasi
3.	<i>A Comparative Study of Image Steganography Based on Edge Detection</i> Oleh (Albayati & Ali, 2021)	Di dalam penelitian ini dibahas mengenai steganografi dengan menyisipkan pesan ke dalam citra digital	Pesan yang disisipkan tidak memiliki intensi untuk berkomunikasi atau informasional

No.	Paper Judul (Penulis, Tahun Publikasi)	Ringkasan	Research Gap
4.	<i>An Analysis on Video Steganography Techniques</i> Oleh (Jenifer, 2021)	Pembahasan steganografi yang dilakukan merupakan perbandingan antara teknik steganografi video yang berbeda dengan menyisipkan citra digital rahasia ke dalamnya	Kompresi gambar terhadap video dan bukan teks terhadap medium citra digital
5.	<i>A Comparative Analysis of LSB, MSB and PVD Based Image Steganography</i> Oleh (Modupe et al., 2021)	Dibandingkan teknik steganografi yang berbeda yaitu LSB, MSB dan PVD dengan menggunakan citra digital	Analisis Teknik LSB dilakukan dengan medium citra berskala abu-abu
6	<i>Comparative Analysis Run-Length Encoding Algorithm and Fibonacci Code Algorithm on Image Compression</i> Oleh (Hardi et al., 2019)	<i>Run-Length Encoding</i> mendapatkan nilai kompresi 69.355%	Analisa yang dilakukan menggunakan <i>string</i> teks

No.	Paper Judul (Penulis, Tahun Publikasi)	Ringkasan	Research Gap
7.	<i>Comparative study between LM-DH technique and Huffman coding technique</i> (Odat et al., 2015)	<i>Huffman coding</i> dapat memberikan nilai kompresi yang stabil di angka 90%	Masukan yang digunakan akan menggunakan <i>string text</i>
8.	<i>A Comparative Study Of Text Compression Algorithms</i> (Shanmugasundaram & Lourdusamy, 2011)	<i>Arithmetic coding</i> peformanya dalam mengompresi teks mendapat rata-rata BPC 5.15 atau kompresi 64.3% serta Izw sebesar 61.25%	Hasil dari kompresi teks akan diujicobakan ke dalam algoritma LSB dan berupa <i>binary string</i>
9	<i>Implementasi Algoritma J-Bit Encoding Pada Kompresi File PDF</i> (Naibaho, 2020)	<i>J-Bit Encoding</i> pada berkas dengan ekstensi PDF memberikan kompresi sebesar 78%	Data yang diujikan menggunakan teks <i>string</i> .

No.	Paper Judul (Penulis, Tahun Publikasi)	Ringkasan	Research Gap
10.	<i>Modern Lossless Compression Techniques: Review, Comparison and Analysis</i> (Gupta et al., 2017)	<i>deflate coding</i> mendapatkan hasil kompresi 84.7%	Akan diujikan dengan algoritma menggunakan <i>coding</i> dan bukan aplikasi <i>open source</i> yang menggunakan algoritma <i>deflate</i>

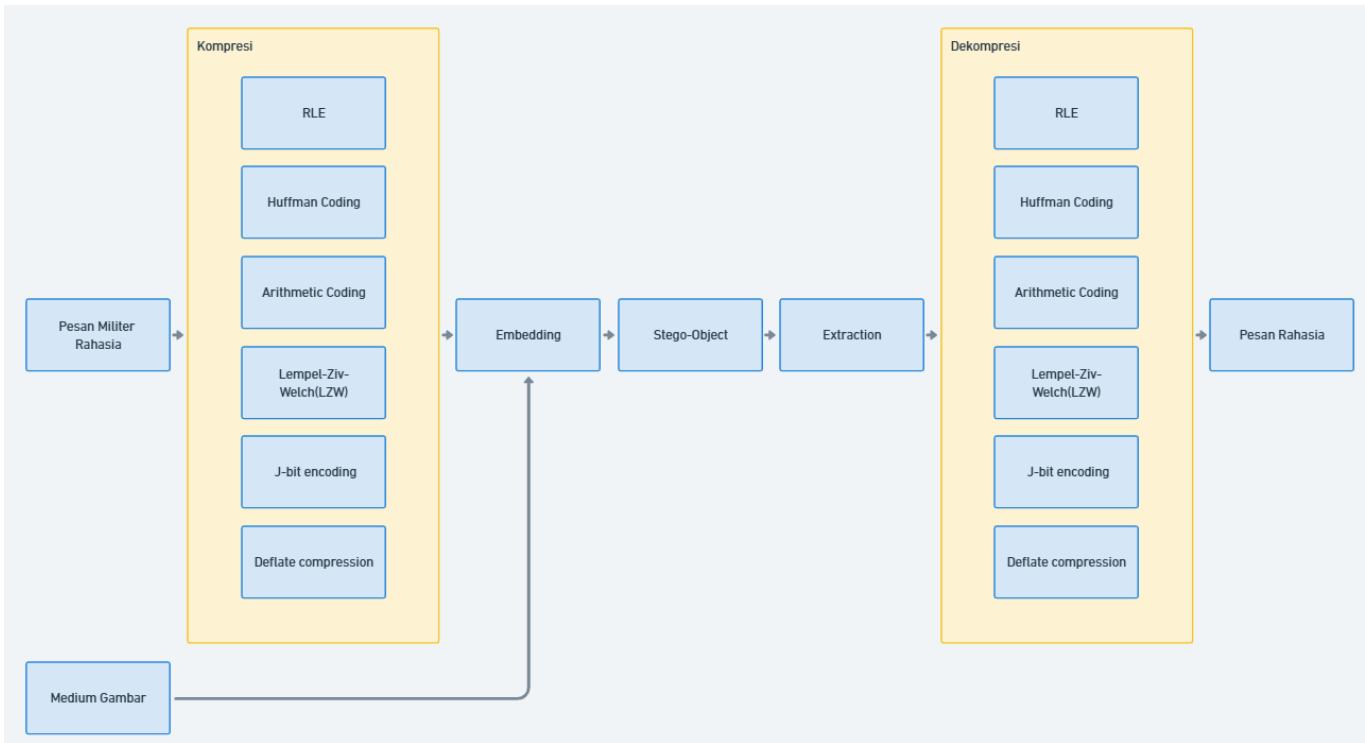
### 2.3 Kerangka Berpikir

Penelitian ini mencoba mencari tahu penggunaan algoritma kompresi yang paling efisien dalam melakukan kompresi terhadap pesan rahasia. Untuk dapat melakukan hal tersebut dicarilah parameter beserta algoritma kompresi terkait yang dapat digunakan dalam penelitian ini. Untuk pemenuhan informasi yang diperlukan studi literatur yang didapatkan merupakan penelitian yang melakukan penelitian terkait dengan penggunaan algoritma kompresi serta studi komparasi terhadap penggunaan teknik atau pun metode steganografi yang dapat menjadi rujukan dalam pelaksanaan penelitian yang penulis kerjakan.

Terdapat 2 proses yang memerlukan algoritma tertentu yaitu dalam proses kompresi data dan penyisipan pesan. Dalam penelitian ini digunakan 6 algoritma yang digunakan untuk melakukan kompresi yaitu *Run-Length Encoding*, *Huffman Coding*, *Arithmetic Coding*, *Lempel-Ziv-Welch* (LZW), *J-bit Encoding*, dan *Deflate Compression*. Algoritma-algoritma tersebut digunakan untuk meng-kompresi pesan yakni berupa teks bertipe data *string* yang dimasukan oleh pengguna. Sedangkan algoritma LSB digunakan untuk melakukan penyisipan pesan ke dalam medium gambar begitu pula dalam pengekstrasi pesan.

Algoritma-algoritma yang digunakan dalam proses kompresi tersebut, dipilih karena mereka merupakan algoritma yang cukup sering ditemui dalam steganografi. Selain dari banyaknya penelitian yang mengklaim kinerja algoritma yang baik mereka juga diakui sebagai algoritma yang tergolong *lossless* artinya pesan yang dikompresi dengan algoritma tersebut dapat dipulihkan dengan sempurna kembali ke teks awal.

Adapun perancangan ini dibangun dengan tujuan agar proses steganografi dapat dilaksanakan dalam bentuk yang lebih mudah untuk digunakan dalam proses penyisipan pesan maupun ekstraksi oleh *user* baik dalam proses pengujian maupun pengembangan untuk dimanfaatkan lebih lanjut. Adapun sistem yang dirancang akan berfungsi menjadi sebuah aplikasi/sistem yang digunakan untuk melakukan *embedding* maupun ekstraksi *Stegoimage* tanpa menyimpan data/dokumen apa pun di dalam sistem, hal ini juga bertujuan agar tidak ada informasi yang disimpan dalam sistem sehingga mengurangi risiko kebocoran informasi serta lebih fleksibel untuk pengembangan seterusnya.



**Gambar 2.1 Kerangka Berpikir**

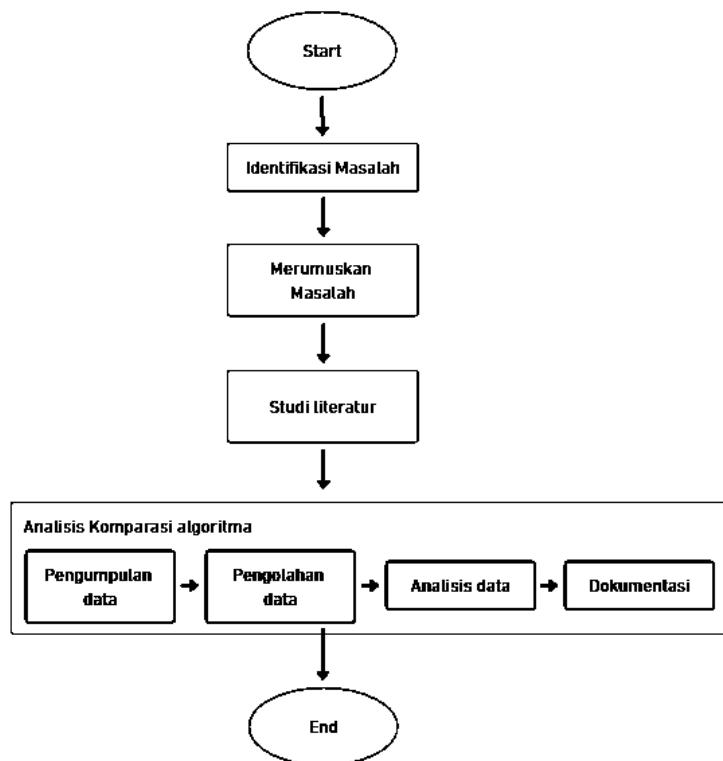
Sumber: diolah oleh peneliti

## BAB III

### METODE PENELITIAN

#### 3.1. Desain Penelitian

Pada penelitian ini, dilakukan pengujian komparatif dengan menggunakan pendekatan kuantitatif terhadap enam algoritma yang telah dibahas pada bagian penelitian terdahulu, yakni algoritma *Run-Length Encoding*, *Huffman Coding*, *Arithmetic Coding*, *Lempel-Ziv-Welch* (LZW), *J-bit Encoding*, dan *Deflate Compression*. Selanjutnya algoritma tersebut akan diujikan terhadap teks yang berbeda. Penelitian ini membandingkan keenam algoritma tersebut dalam meraih nilai efisiensi berdasarkan faktor waktu penggerjaan, ukuran data medium, serta besaran *input*. Perancangan alur penelitian merupakan gambaran umum terkait penelitian yang akan dilakukan dari awal hingga akhir penggerjaan penelitian. Alur kerja penelitian dapat terlihat dari gambar berikut.



**Gambar 3.1 Desain Penelitian**

Sumber: diolah oleh peneliti

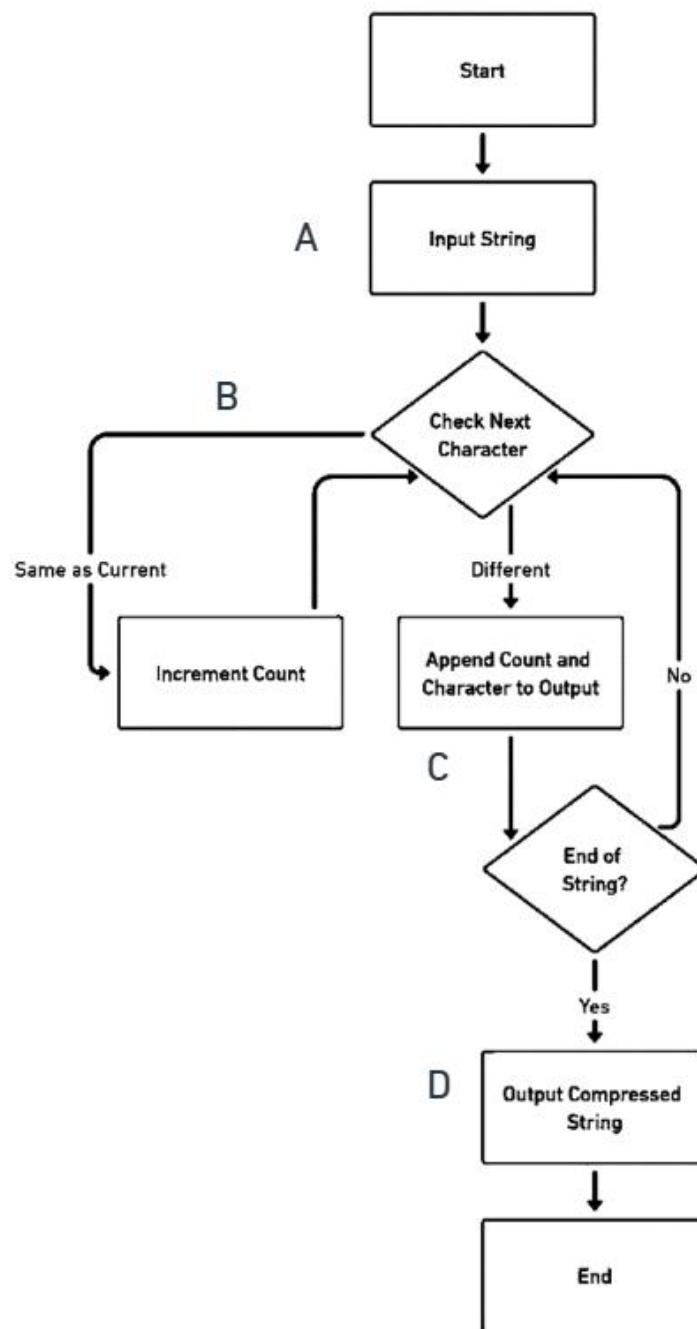
Alur dari penelitian ini diawali dengan mengidentifikasi masalah yang perlu peneliti pecahkan yakni bagaimana menyediakan skema untuk analisis algoritma kompresi bit untuk steganografi dalam distribusi informasi. Kemudian peneliti susun rumusan masalah berdasarkan identifikasi masalah yang telah dicari sebelumnya. Selanjutnya, peneliti lakukan persiapan dalam penelitian yaitu dengan melakukan studi literatur serta melakukan implementasi aplikasi agar dapat memastikan bahwa penerapan algoritma kompresi merupakan solusi yang praktikal.

Setelah itu, proses analisis komparasi algoritma dilakukan. Pertama, pengumpulan data dilakukan pada algoritma kompresi bit. Setelah data berhasil dikumpulkan, data diproses yang selanjutnya hasil dari data yang telah diproses dilakukan analisis untuk mendapat kesimpulan algoritma apa yang efisien untuk digunakan dalam steganografi citra digital. Kemudian dilakukan tahap validasi untuk dapat digeneralisasi apakah hasil yang didapat adalah konsisten. Terakhir, dilakukan dokumentasi atas seluruh tahapan yang telah dilakukan.

### **3.2. Alur Kerja Algoritma Kompresi Bit**

#### **3.2.1. Alur Kerja Algoritma *Run-Length Encoding***

Peneliti akan memberikan gambaran terhadap cara kerja dari algoritma *Run-length Encoding*. Untuk memperjelas proses kerja dari algoritma ini, berikut merupakan diagram alur dari algoritma Run-Length Encoding.



**Gambar 3.2 Flowchart Run-Length Encoding**

Sumber: diolah oleh peneliti

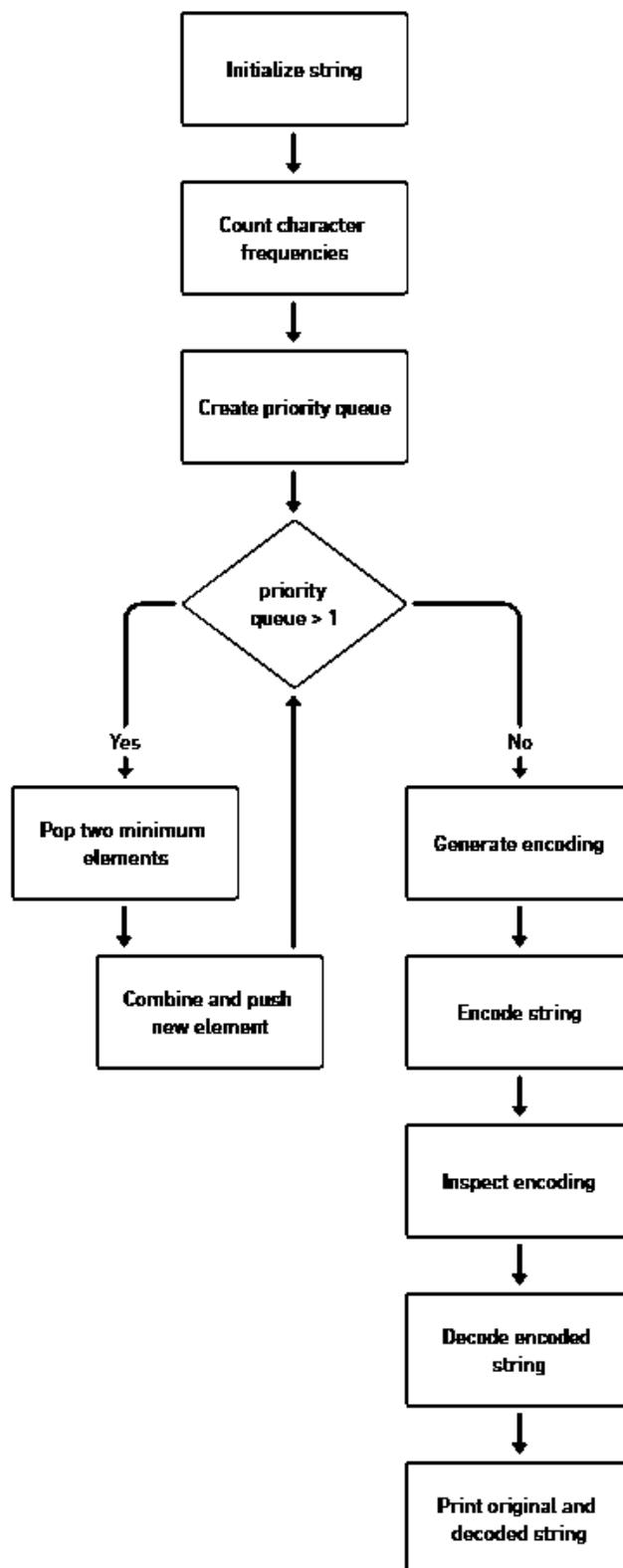
Secara garis besar proses yang terjadi pada *flowchart Run-Length Encoding* yaitu:

- a. Gambar Masukan: Proses dimulai dengan gambar masukan yang perlu dikompres.

- b. Analisis Piksel: Algoritma menganalisis piksel dalam gambar untuk mengidentifikasi rangkaian piksel yang identik.
- c. *Encoding*: Urutan piksel identik dikodekan sebagai nilai tunggal dan hitungan, masing-masing mewakili warna dan panjang proses.
- d. Keluaran Terkompresi: Hasilnya adalah representasi gambar terkompresi, yang memakan lebih sedikit ruang penyimpanan.

### 3.2.2. Alur Kerja Algoritma *Huffman Coding*

Kemudian Peneliti akan memberikan gambaran terhadap cara kerja dari algoritma *Huffman Coding*. Untuk memperjelas proses kerja dari algoritma ini, berikut merupakan diagram alur dari algoritma *Huffman Coding*.



**Gambar 3.3 Flowchart Huffman Coding**

Sumber: diolah oleh peneliti

Secara garis besar proses pada *Huffman Encoding* seperti yang dapat dilihat pada gambar 3.2 yaitu:

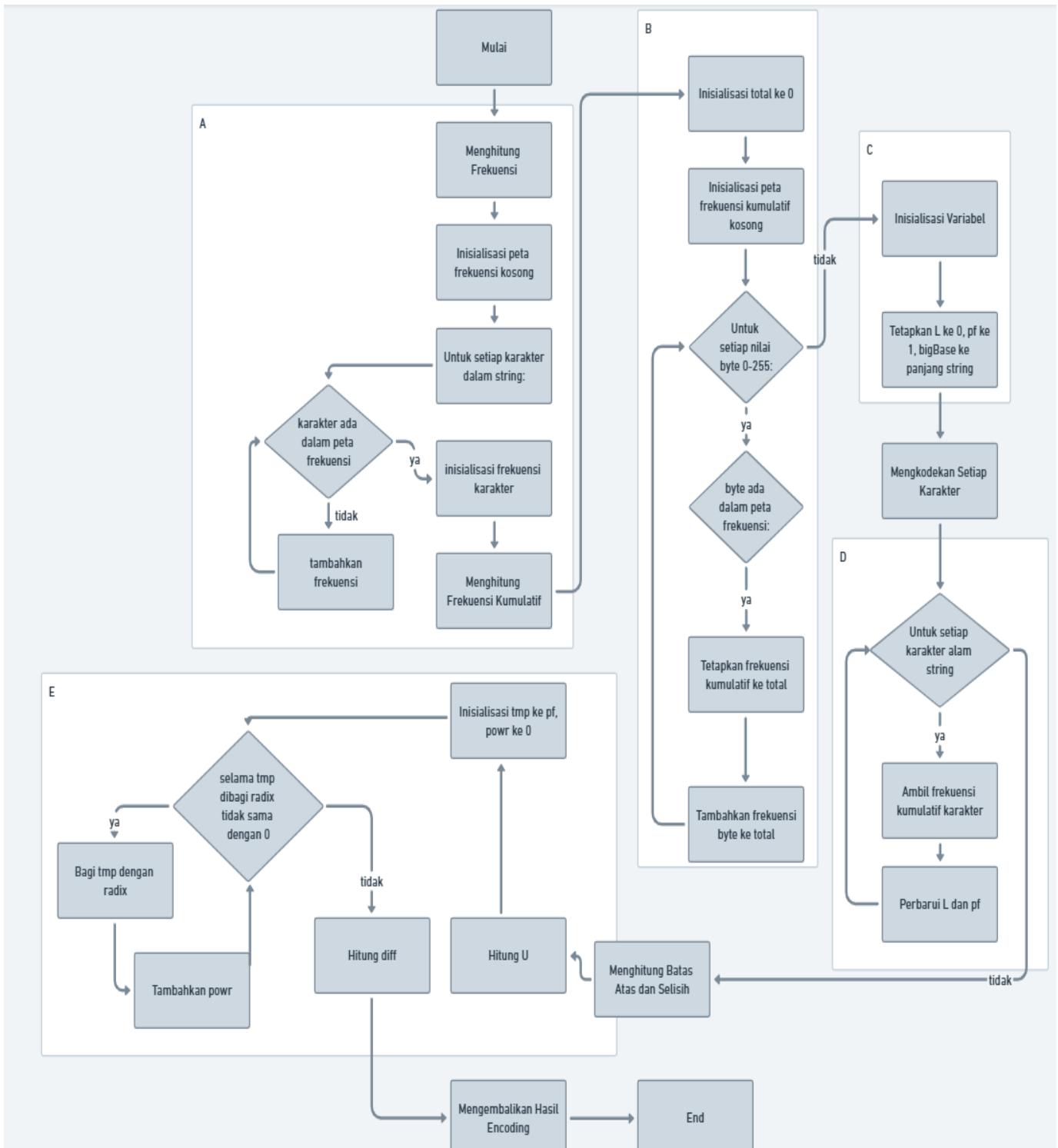
- a. *Initialize string*: Pada tahap ini, inisialisasi *string* yang akan dikompresi menggunakan algoritma *Huffman* dilakukan. Hal ini meliputi penentuan *string* yang akan digunakan sebagai *input*.
- b. *Count character frequencies*: Langkah ini melibatkan penghitungan frekuensi kemunculan setiap karakter dalam *string*. Proses ini dilakukan dengan mengiterasi setiap karakter dalam *string* dan membuat tabel atau *dictionary* yang menyimpan setiap karakter beserta jumlah kemunculannya.
- c. *Create priority queue*: Pada tahap ini, dibuat *priority queue* (antrian prioritas) dari karakter-karakter berdasarkan frekuensinya. Langkah-langkahnya meliputi inisialisasi *priority queue* kosong, kemudian memasukkan setiap karakter ke dalam *priority queue* dengan prioritas sesuai frekuensinya (frekuensi lebih kecil memiliki prioritas lebih tinggi).
- d. *While loop*: Langkah ini mengulangi proses sampai hanya tersisa satu elemen dalam *priority queue*. Cek apakah ukuran *priority queue* lebih dari satu.
- e. *Pop two minimum elements*: Pada tahap ini, dua elemen dengan frekuensi terendah dikeluarkan dari *priority queue*. Langkah-langkahnya meliputi pengambilan elemen pertama (elemen dengan frekuensi terendah) dan elemen kedua (elemen dengan frekuensi terendah berikutnya).
- f. *Combine and push new element*: Langkah ini menggabungkan dua elemen yang diambil dan memasukkan elemen baru ke dalam *priority queue*. Langkah-langkahnya meliputi pembuatan *node* baru yang menggabungkan dua elemen yang diambil dengan menjumlahkan frekuensi mereka, kemudian memasukkan *node* baru ke dalam *priority queue*.

- g. *Generate encoding*: Pada tahap ini, dihasilkan *encoding* (kode Huffman) untuk setiap karakter berdasarkan *tree* yang terbentuk. Langkah-langkahnya meliputi memulai dari *root tree* yang terbentuk, memberikan kode '0' untuk cabang kiri dan '1' untuk cabang kanan, atau sebaliknya, kemudian melakukan rekursif ke bawah *tree* untuk setiap karakter, mencatat kode yang terbentuk.
- h. *Encode string*: Langkah ini melibatkan *encoding string* asli menggunakan kode Huffman yang telah dihasilkan. Langkah-langkahnya meliputi mengganti setiap karakter dalam *string* asli dengan kode Huffman yang sesuai.
- i. *Inspect encoding*: Pada tahap ini, hasil *encoding* diinspeksi untuk verifikasi atau analisis. Langkah-langkahnya meliputi memeriksa panjang atau pola dari *string* yang telah *di-encode* dan membandingkannya dengan *string* asli jika diperlukan untuk verifikasi.
- j. *Decode encoded string*: Langkah ini melibatkan *decoding* *string* yang telah *di-encode* kembali ke bentuk aslinya. Langkah-langkahnya meliputi memulai dari *root tree Huffman*, membaca setiap bit dalam *string* yang *di-encode*, bergerak di *tree* sesuai bit yang dibaca ('0' ke kiri, '1' ke kanan), dan menambahkan karakter yang ditemukan di *leaf node* ke *string* hasil *decoding*.
- k. *Print original and decoded string*: Pada tahap akhir ini, *string* asli dan *string* hasil *decoding* dicetak untuk verifikasi. Langkah-langkahnya meliputi mencetak *string* asli, mencetak *string* hasil *decoding*, dan membandingkan keduanya untuk memastikan mereka sama.

### **3.2.3. Alur Kerja Algoritma *Arithmetical Coding***

Selanjutnya peneliti akan memberikan gambaran terhadap cara kerja dari algoritma *Arithmetical Coding*. Untuk memperjelas proses kerja dari algoritma ini, *flowchart* akan dibagi menjadi dua bagian yaitu *encoding* dan

*decoding*. Berikut merupakan diagram alur *encoding* dari algoritma *Run-Length Encoding*.



**Gambar 3.4 Flowchart Arithmetic Coding**

Sumber: diolah oleh peneliti

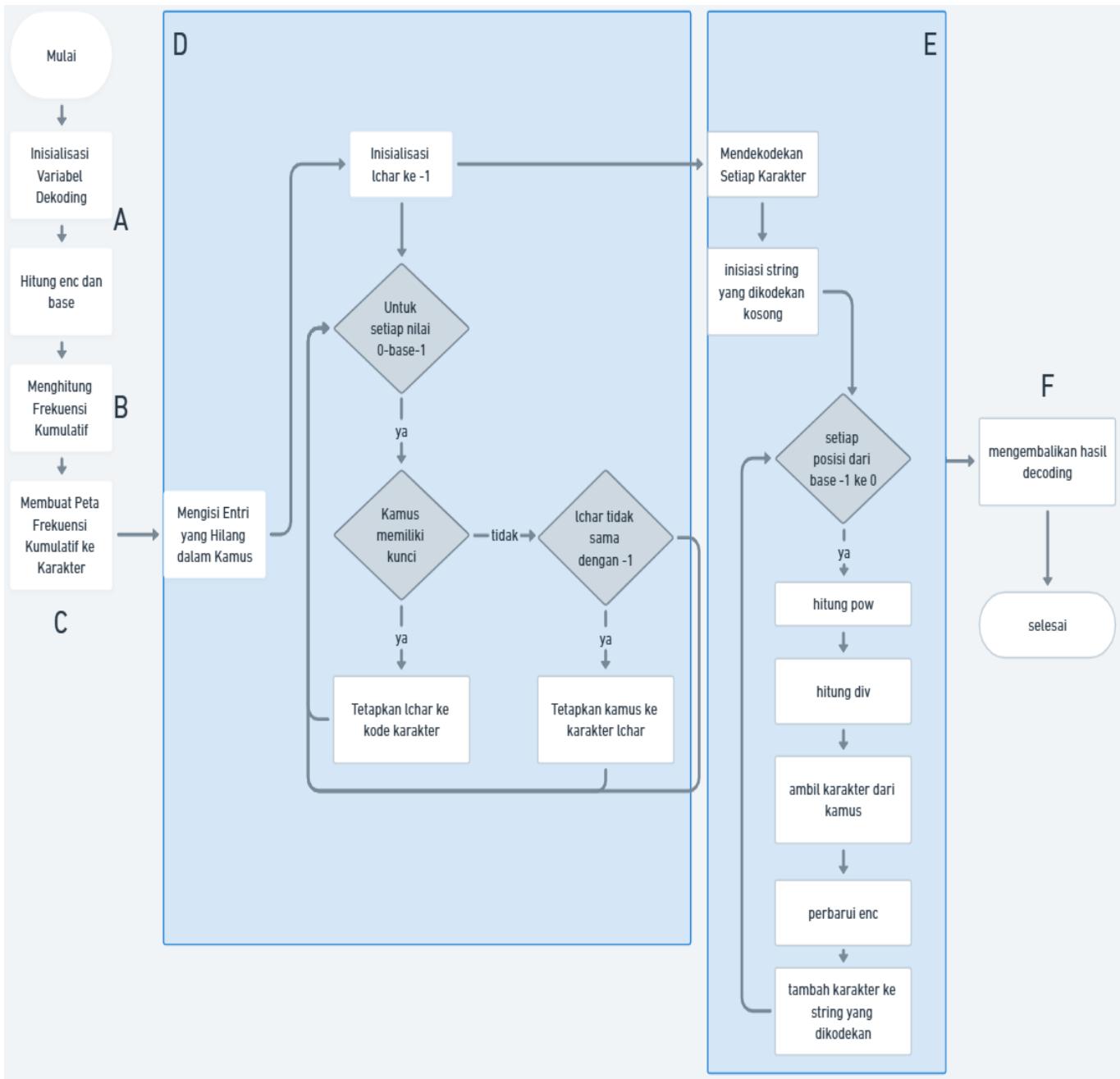
Secara garis besar proses yang terjadi pada *flowchart Run-Length Encoding* yaitu:

- A. Menghitung Frekuensi: Langkah pertama dalam proses encoding adalah menghitung frekuensi setiap karakter dalam string yang akan dikodekan. Peta frekuensi dibuat untuk menyimpan jumlah kemunculan setiap karakter. Dengan cara ini, diperoleh informasi tentang berapa kali setiap karakter muncul dalam *string* yang diberikan.
- B. Menghitung Frekuensi Kumulatif: Setelah peta frekuensi dibuat, langkah selanjutnya adalah menghitung frekuensi kumulatif untuk setiap karakter. Frekuensi kumulatif dari sebuah karakter adalah jumlah frekuensi semua karakter yang mendahuluinya dalam urutan karakter (0-255). Ini membantu dalam menentukan interval probabilitas yang unik untuk setiap karakter.
- C. Inisialisasi Variabel: Variabel-varibel penting diinisialisasi sebelum memulai proses encoding karakter. L diinisialisasi ke 0 sebagai batas bawah dari interval, pf diinisialisasi ke 1 sebagai faktor probabilitas, dan bigBase diinisialisasi ke panjang *string*. Ini adalah persiapan untuk mengkodekan *string* berdasarkan interval probabilitas yang diperoleh.
- D. Mengkodekan Setiap Karakter: Pada langkah ini, setiap karakter dalam string dikodekan satu per satu. Untuk setiap karakter, L diperbarui dengan mengalikan L dengan bigBase dan menambahkan produk dari frekuensi kumulatif karakter dan pf. Kemudian pf diperbarui dengan mengalikan pf dengan frekuensi karakter tersebut. Proses ini mempersempit interval probabilitas hingga mewakili seluruh string.
- E. Menghitung Batas Atas dan Selisih: Setelah semua karakter dikodekan, batas atas interval probabilitas (U) dihitung dengan menambahkan L dan pf. Selanjutnya, eksponen (powr) yang

diperlukan untuk mengkonversi interval ke basis yang diinginkan dicari dengan membagi pf secara berturut-turut dengan radix hingga hasilnya nol. Akhirnya, selisih (diff) antara batas atas dan bawah interval dibagi dengan radix dipangkatkan dengan powr.

- F. Mengembalikan Hasil *Encoding*: Hasil encoding adalah nilai yang dikodekan (diff), eksponen (powr), dan peta frekuensi (freq). Nilai-nilai ini digunakan untuk mendekode string asli kembali. Proses *encoding* selesai dengan mengembalikan hasil-hasil tersebut.

Setelah proses *encoding*, proses selanjutnya adalah proses *decoding*. Berikut merupakan *flowchart* dari proses *decoding* algoritma *Arithmetic Coding*:



**Gambar 3. 5 Flowchart Decoding Arithmetic Coding**

Sumber: diolah oleh peneliti

Secara garis besar proses yang terjadi pada *flowchart Run-Length Encoding* yaitu:

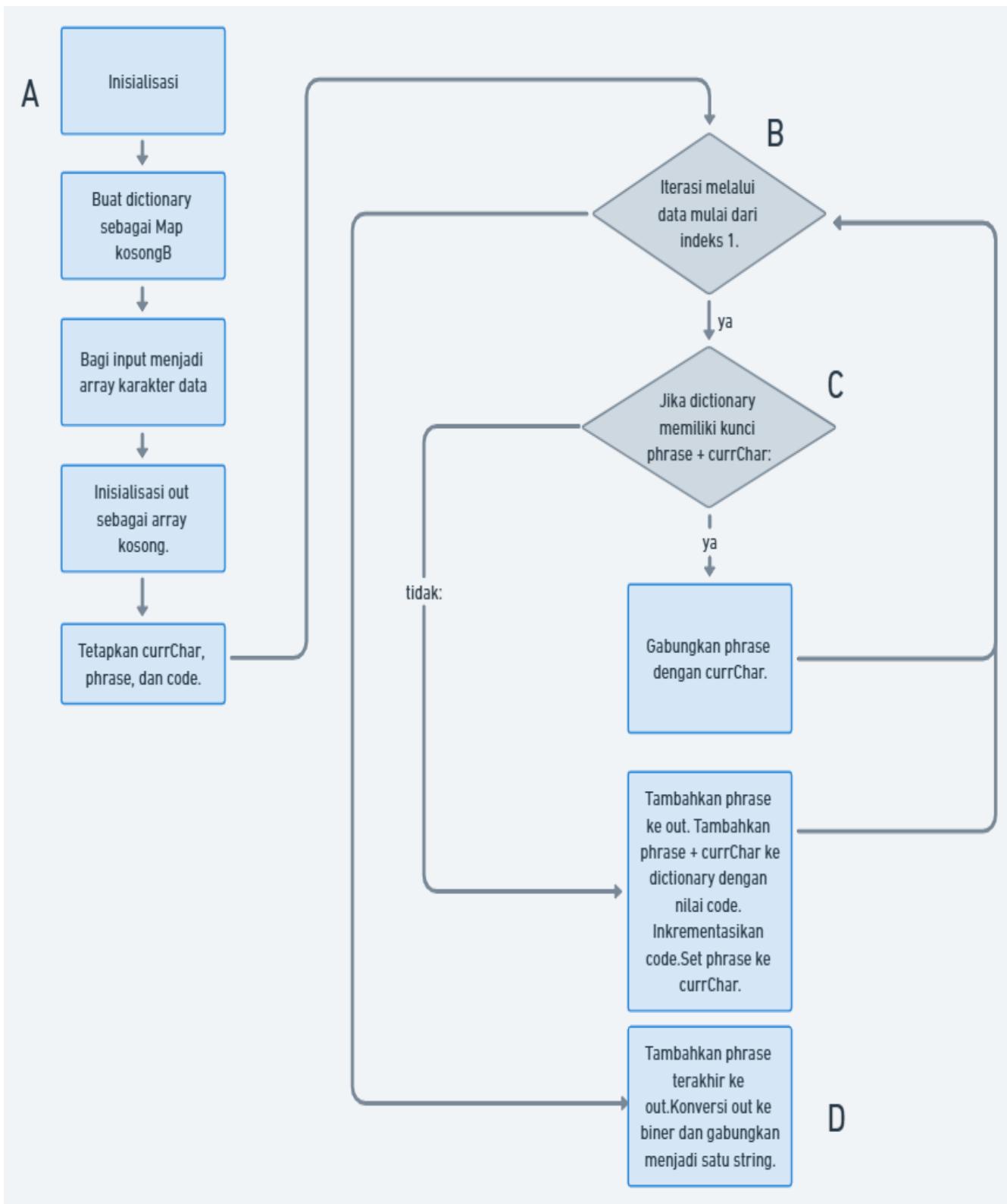
- Inisialisasi Variabel Dekoding:** Proses decoding dimulai dengan menginisialisasi variabel-variabel yang diperlukan. Angka yang

- dikodekan (num) dikonversi menjadi nilai yang dapat didekodekan (enc) dengan mengalikan num dengan radix dipangkatkan dengan pow. Selanjutnya, *base* dihitung sebagai jumlah semua frekuensi dalam peta frekuensi.
- B. Menghitung Frekuensi Kumulatif: Sama seperti dalam proses encoding, frekuensi kumulatif dari peta frekuensi dihitung. Ini membantu dalam menentukan interval probabilitas yang sesuai untuk mendekodekan angka yang dikodekan kembali menjadi *string* asli.
  - C. Membuat Peta Frekuensi Kumulatif ke Karakter: Setelah mendapatkan frekuensi kumulatif, kamus dibuat untuk memetakan nilai kumulatif ke karakter yang sesuai. Ini berarti setiap nilai kumulatif sekarang memiliki karakter yang unik yang dapat digunakan untuk mendekode angka yang dikodekan.
  - D. Mengisi Entri yang Hilang dalam Kamus: Langkah ini memastikan bahwa setiap nilai dalam rentang base memiliki entri dalam kamus. Jika ada nilai yang hilang, nilai tersebut diisi dengan karakter terakhir yang diketahui. Ini penting untuk memastikan bahwa semua nilai dapat dipetakan ke karakter selama proses *decoding*.
  - E. Mendekodekan Setiap Karakter: Dengan kamus yang lengkap, setiap karakter mulai didekodekan satu per satu. Untuk setiap posisi, nilai pembagi (div) dari enc dan pow dihitung. Karakter yang sesuai dengan nilai div diambil dari kamus dan ditambahkan ke string yang didekodekan. enc kemudian diperbarui untuk posisi berikutnya dengan mengurangi produk dari pow dan frekuensi kumulatif karakter, lalu dibagi dengan frekuensi karakter.
  - F. Mengembalikan Hasil Dekoding: Setelah semua karakter berhasil didekodekan, string yang didekodekan dikembalikan

sebagai hasil akhir. Proses ini menghasilkan string asli dari angka yang dikodekan, menyelesaikan proses *decoding*.

### 3.2.4. Alur Kerja Algoritma *Lempel-Ziv-Welch*

Keempat peneliti akan memberikan gambaran terhadap cara kerja dari algoritma *Lempel-Ziv-Welch*. Untuk memperjelas proses kerja dari algoritma ini, *flowchart* akan dibagi menjadi dua bagian yaitu *encoding* dan *decoding*. berikut merupakan diagram alur *encoding* dari algoritma *Lempel-Ziv-Welch*.



**Gambar 3.6 Flowchart Encoding Lempel-Ziv-Welch**

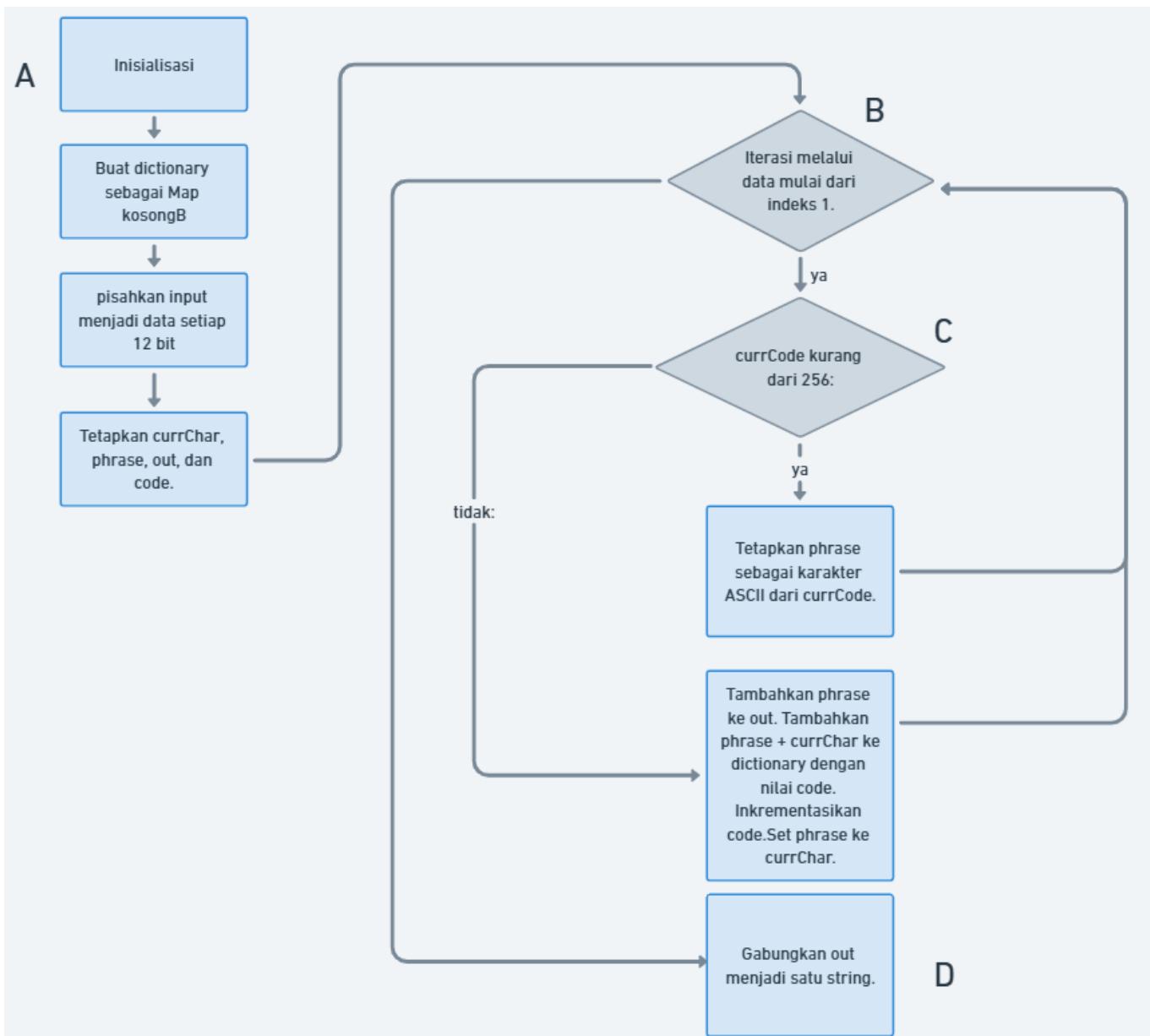
Sumber: diolah oleh peneliti

Secara garis besar proses yang terjadi pada *flowchart encoding Lempel-Ziv-Welch* yaitu:

- A. Inisialisasi Struktur Data: Inisialisasi dimulai dengan pembuatan *dictionary* sebagai sebuah *Map* kosong. Setiap karakter dari input dipisahkan dan disimpan dalam *array* data. Selain itu, *array* *out* disiapkan untuk menampung *output* kompresi. Variabel *phrase* diatur sebagai karakter pertama dari data, dan variabel *code* diinisialisasi dengan nilai 256 untuk melacak kode-kode baru yang akan ditambahkan ke *dictionary*.
- B. Iterasi Melalui *Input*: Melalui perulangan, setiap karakter dalam data diakses mulai dari indeks kedua hingga akhir. Pada setiap iterasi, variabel *currChar* diatur sebagai karakter saat ini. Jika *dictionary* sudah memiliki entri untuk kombinasi *phrase* + *currChar*, maka *phrase* diperbarui dengan menambahkan *currChar*. Hal ini memungkinkan untuk membangun *string* lebih panjang yang mungkin sudah ada dalam *dictionary*.
- C. Memperbarui *Dictionary* dan *Output*: Jika kombinasi *phrase* + *currChar* tidak ditemukan dalam *dictionary*, maka nilai kode untuk *phrase* dimasukkan ke dalam *array* *out*. Jika *phrase* memiliki panjang lebih dari satu karakter, nilai yang dimasukkan adalah kode yang terkait dengan *phrase* dalam *dictionary*. Jika tidak, nilai ASCII dari karakter tersebut yang dimasukkan. Kombinasi *phrase* + *currChar* kemudian ditambahkan ke *dictionary* dengan nilai *code*, yang kemudian diinkrementasikan. Setelah itu, *phrase* diatur ulang menjadi *currChar* untuk memulai proses kembali.
- D. Menangani Sisa Data: Setelah *loop* selesai, *phrase* terakhir harus ditambahkan ke *out*. Proses ini mirip dengan langkah sebelumnya, di mana nilai kode dari *phrase* dimasukkan ke *out*. Terakhir, *output* *array* *out* dikonversi menjadi representasi biner

menggunakan fungsi `toBinary` dan hasil akhirnya digabungkan menjadi satu *string*, menghasilkan data terkompresi.

Setelah proses *encoding*, proses selanjutnya adalah proses *decoding*. Berikut merupakan *flowchart* dari proses *decoding* algoritma *Arithmetic Coding*:



**Gambar 3.7 Flowchart Decoding Lempel-Ziv-Welch**

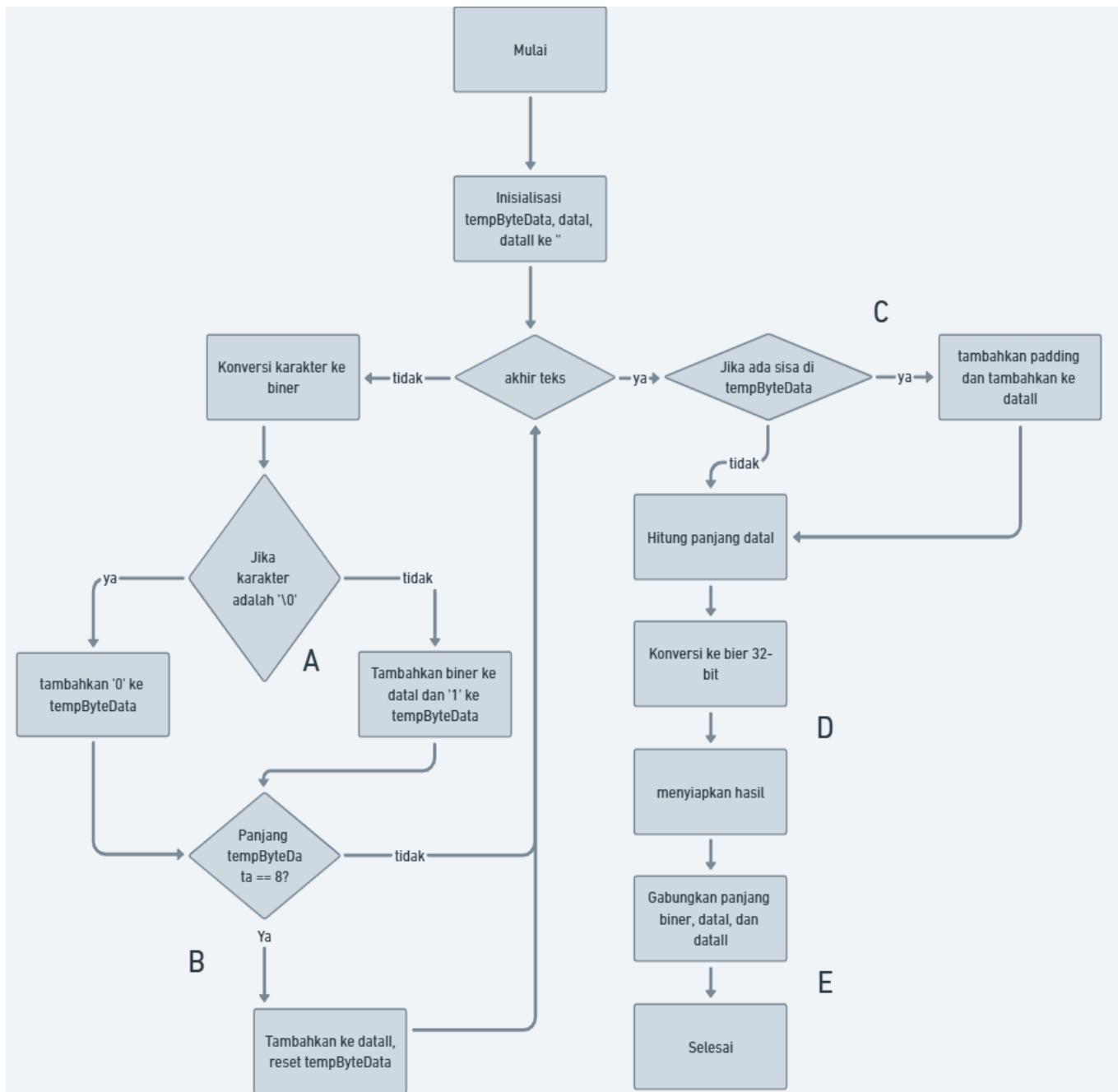
Sumber: diolah oleh peneliti

Secara garis besar proses yang terjadi pada *flowchart encoding Lempel-Ziv-Welch* yaitu:

- A. Inisialisasi Struktur Data: Proses dekompresi dimulai dengan inisialisasi *dictionary* sebagai sebuah *Map* kosong dan *array* data untuk menampung kode-kode terkompresi yang diekstraksi dari *input* biner. Input biner dipecah setiap 12 bit untuk mendapatkan representasi *integer* dari kode-kode tersebut. Karakter pertama diubah kembali dari kode ASCII ke karakter asli dan disimpan dalam *currChar* dan *oldPhrase*. *Output array* *out* diinisialisasi dengan *currChar*, dan *code* diatur pada nilai 256.
- B. Iterasi Melalui Kode Terkompresi: Pada setiap iterasi, *currCode* diatur sebagai nilai *integer* dari 12 bit berikutnya dalam data. Jika *currCode* kurang dari 256, *phrase* diatur sebagai karakter yang sesuai dengan nilai ASCII dari *currCode*. Jika *currCode* lebih besar atau sama dengan 256, *phrase* diatur sebagai nilai dari *dictionary* jika *currCode* ditemukan. Jika tidak ditemukan, *phrase* diatur sebagai *concatenation* dari *oldPhrase* dan *currChar*.
- C. Memperbarui *Dictionary* dan *Output*: *phrase* kemudian ditambahkan ke *array* *out*. Karakter pertama dari *phrase* disimpan dalam *currChar*. Kombinasi *oldPhrase* + *currChar* kemudian ditambahkan ke *dictionary* dengan nilai *code*, yang kemudian diinkrementasikan. *oldPhrase* diperbarui dengan *phrase* untuk digunakan dalam iterasi berikutnya.
- D. Menggabungkan Hasil Dekompresi: Setelah *loop* selesai, *array* *out* yang berisi karakter-karakter hasil dekompresi digabungkan menjadi satu *string*, menghasilkan data yang telah didekompresi kembali ke bentuk aslinya. Proses ini memungkinkan untuk mengembalikan data terkompresi ke bentuk asli tanpa kehilangan informasi.

### **3.2.5. Alur Kerja Algoritma *J-bit Encoding***

Kelima peneliti akan memberikan gambaran terhadap cara kerja dari algoritma *J-bit Encoding*. Untuk memperjelas proses kerja dari algoritma ini, *flowchart* akan dibagi menjadi dua bagian yaitu *encoding* dan *decoding*. Berikut merupakan diagram alur *encoding* dari algoritma *J-bit Encoding*.



**Gambar 3.8 Flowchart Encoding J-bit Encoding**

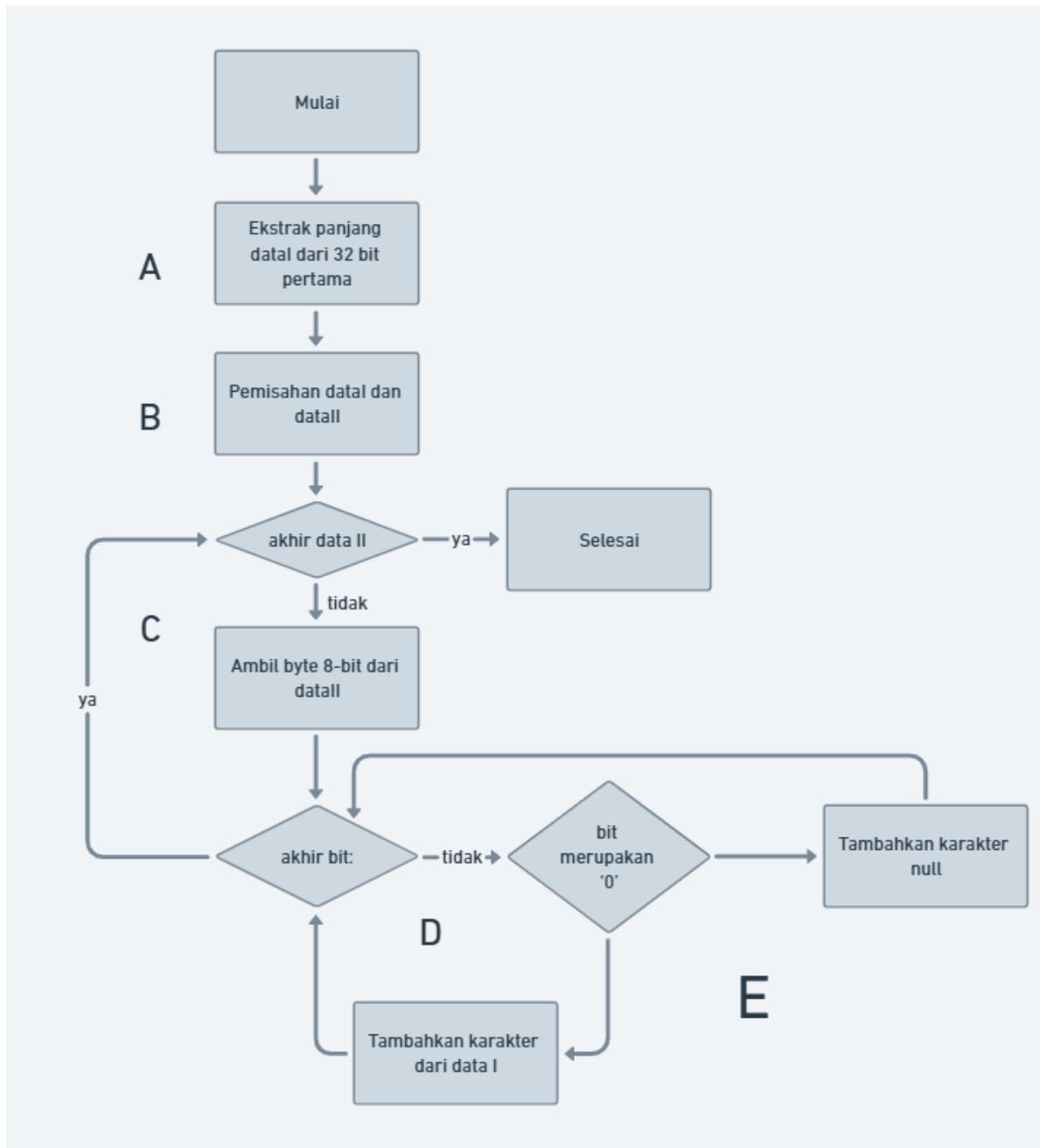
Sumber: diolah oleh peneliti

Secara garis besar proses yang terjadi pada *flowchart encoding J-bit Encoding* yaitu:

- A. Pembentukan *String* tempByteData dan data I: Dua string, tempByteData dan datal, dibentuk secara paralel selama iterasi teks. Untuk setiap karakter, jika karakter tersebut adalah null ('\0'), '0' ditambahkan ke tempByteData. Jika karakter bukan null, representasi biner dari karakter tersebut ditambahkan ke datal, dan '1' ditambahkan ke tempByteData. Proses ini membantu membedakan antara karakter null dan karakter lain dalam representasi biner.
- B. Pengumpulan *String* data II: Setiap kali panjang tempByteData mencapai 8 bit, string ini ditambahkan ke datall, dan tempByteData direset menjadi string kosong. Jika pada akhir teks, panjang tempByteData kurang dari 8 bit, string tersebut diisi dengan padding '0' hingga mencapai 8 bit sebelum ditambahkan ke datall. Langkah ini memastikan bahwa datall selalu terdiri dari byte yang lengkap, yang esensial untuk proses decoding nantinya.
- C. Penggabungan Panjang data I ke Dalam Hasil Akhir: Panjang dari datal dikonversi ke representasi biner 32-bit dan ditempatkan di awal string hasil akhir. Ini dilakukan untuk memastikan bahwa informasi tentang panjang data I tersimpan dan dapat diakses dengan mudah selama proses decoding, sehingga byte biner yang relevan dapat diambil dengan akurat.
- D. Penggabungan *String* Akhir: Hasil akhir dari proses *encoding* adalah penggabungan dari panjang datal dalam bentuk biner 32-bit, diikuti oleh data I, dan diakhiri dengan data II. Proses ini menciptakan *string* biner panjang yang menggabungkan semua informasi yang diperlukan untuk merekonstruksi teks asli,

dengan memanfaatkan informasi tentang keberadaan karakter *null* dan karakter lainnya dalam urutan biner.

Setelah proses *encoding*, proses selanjutnya adalah proses *decoding*. Berikut merupakan *flowchart* dari proses *decoding* algoritma *Arithmetic Coding*:



**Gambar 3.9 Flowchart Decoding J-bit Encoding**

Sumber: diolah oleh peneliti

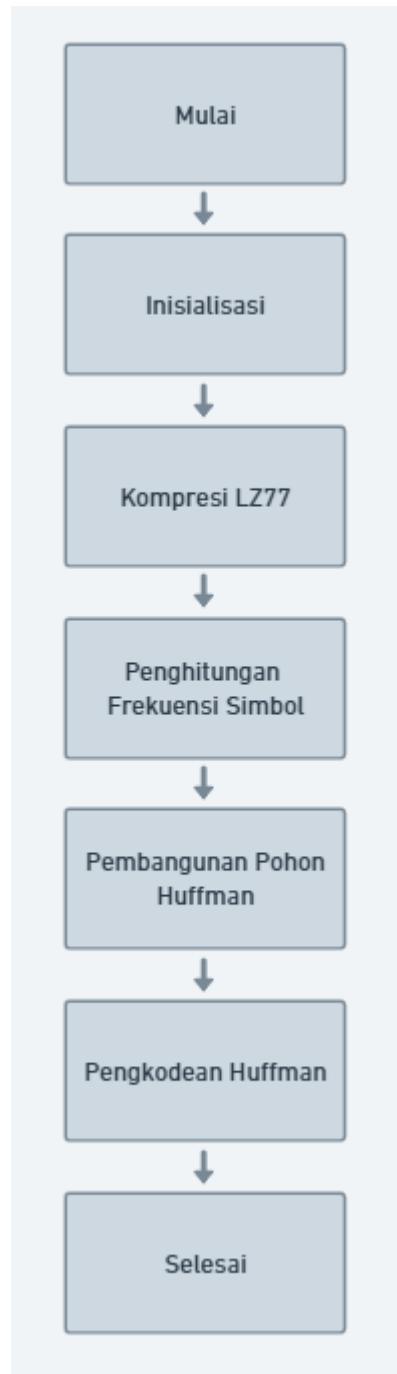
Secara garis besar proses yang terjadi pada *flowchart decoding J-bit Encoding* yaitu:

- A. Ekstraksi Panjang datal: Proses *decoding* dimulai dengan mengekstraksi panjang datal dari 32 bit pertama *string* data. Panjang ini dikonversi dari representasi biner ke *integer*. Langkah ini penting untuk menentukan batasan dari datal dalam *string* data, memungkinkan identifikasi yang jelas dari segmen yang relevan dalam proses *decoding*.
- B. Pemisahan data I dan data II: Setelah panjang data I diekstraksi, *string* data dibagi menjadi dua bagian yaitu data I dan data II. Data I terdiri dari bit biner yang sesuai dengan panjang yang diekstraksi, sementara sisa *string* merupakan data II. Pemisahan ini memungkinkan pemrosesan paralel dari kedua bagian, yang masing-masing memainkan peran berbeda dalam merekonstruksi teks asli.
- C. Iterasi dan *Decoding String* data II: Iterasi dilakukan melalui data II dalam segmen 8-bit. Setiap segmen 8-bit diproses satu per satu. Untuk setiap bit dalam segmen tersebut, jika bit adalah '1', karakter yang sesuai diambil dari datal menggunakan metode `binaryToChar` dan ditambahkan ke *output*. Jika bit adalah '0', karakter `null` ('\0') ditambahkan ke *output*. Proses ini memastikan bahwa setiap karakter dalam teks asli, termasuk karakter `null`, dapat direkonstruksi dengan akurat berdasarkan informasi yang terkandung dalam data II.
- D. Dekode Karakter dari data I: Untuk setiap bit '1' dalam data II, karakter yang sesuai diambil dari datal menggunakan metode `binaryToChar`. Metode ini mengkonversi string biner 8 bit kembali menjadi karakter dengan menggunakan `parseInt` untuk mengubah biner menjadi kode ASCII untuk mendapatkan

karakter yang sesuai. Proses ini memungkinkan rekonstruksi yang tepat dari karakter asli yang bukan *null*.

### 3.2.6. Alur Kerja Algoritma *Deflate Compression*

Terakhir peneliti akan memberikan gambaran terhadap cara kerja dari algoritma *Deflate Comression*. Untuk memperjelas proses kerja dari algoritma ini, *flowchart* akan dibagi menjadi dua bagian yaitu *encoding* dan *decoding*. Berikut merupakan diagram alur *encoding* dari algoritma *Deflate Comression*.



**Gambar 3.10 Flowchart Encoding Deflate Compression**

Sumber: diolah oleh peneliti

Secara garis besar proses yang terjadi pada *flowchart encoding Deflate Compression* yaitu:

- A. Inisiasi: Algoritma *Deflate* dimulai dengan inisialisasi ukuran *buffer* pencarian (searchBufSize) dan *buffer* tampilan ke depan (lookaheadBufSize). *Buffer* pencarian menyimpan data yang telah dikompresi sementara *buffer* tampilan ke depan menyimpan data yang akan dicari kecocokannya. Variabel *output* diinisialisasi sebagai array kosong yang akan menyimpan hasil kompresi, dan pos diinisialisasi ke 0 untuk melacak posisi saat ini dalam *input*.
- B. Kompresi LZ77: Proses kompresi LZ77 berjalan melalui *input* dengan mencari kecocokan terpanjang antara *buffer* pencarian dan tampilan ke depan. Untuk setiap posisi dalam *input*, algoritma mencari kecocokan terpanjang dengan membandingkan *substring* dalam *buffer* pencarian dengan *substring* dalam *buffer* tampilan ke depan. Jika kecocokan dengan panjang minimal tiga karakter ditemukan, pasangan (*distance*, *length*) ditambahkan ke *output*, dan pos diperbarui. Jika tidak ada kecocokan ditemukan, karakter tunggal dari *input* ditambahkan ke *output*.
- C. Penghitungan Frekuensi Simbol: Setelah kompresi LZ77 selesai, frekuensi setiap simbol dalam *output* dihitung. Simbol dapat berupa karakter tunggal atau pasangan (*distance*, *length*). Frekuensi ini kemudian digunakan untuk membangun pohon *Huffman*. Setiap simbol dalam *output* diiterasi dan jumlah kemunculannya dicatat dalam *objek frequencies*.
- D. Pembangunan Pohon Huffman: Berdasarkan frekuensi simbol yang telah dihitung, pohon Huffman dibangun. Setiap simbol dan frekuensinya dimasukkan ke dalam antrian prioritas. Dua *node* dengan frekuensi terendah diambil dari antrian, dan sebuah *node* baru dengan frekuensi gabungan mereka dibuat. Proses ini diulangi hingga hanya ada satu *node* yang tersisa dalam antrian, yang merupakan akar dari pohon *Huffman*. Kode biner untuk

setiap simbol diperoleh dengan menelusuri pohon *Huffman* dari akar ke daun.

- E. Pengkodean Huffman: Dengan pohon Huffman yang telah dibangun, *output* yang berisi hasil kompresi LZ77 dikodekan menjadi *string* biner menggunakan kode *Huffman*. Setiap simbol dalam *output* diganti dengan kode biner yang sesuai. Simbol karakter tunggal dikodekan menggunakan kode *Huffman* mereka, sedangkan pasangan (*distance*, *length*) dikodekan sebagai satu kesatuan menggunakan kode *Huffman* mereka. *String* biner yang dihasilkan kemudian disatukan menjadi satu *string* panjang.

Setelah proses *encoding*, proses selanjutnya adalah proses *decoding*. Berikut merupakan *flowchart* dari proses *decoding* algoritma *Deflate Compression*:



**Gambar 3.11 Flowchart Encoding Deflate Compression**

Sumber: diolah oleh peneliti

Secara garis besar proses yang terjadi pada *flowchart decoding Deflate Compression* yaitu:

- A. Inisialisasi: Proses dekompresi *Deflate* dimulai dengan inisialisasi string biner yang dikompresi dan kode *Huffman* yang digunakan untuk pengkodean. Inisialisasi juga mencakup

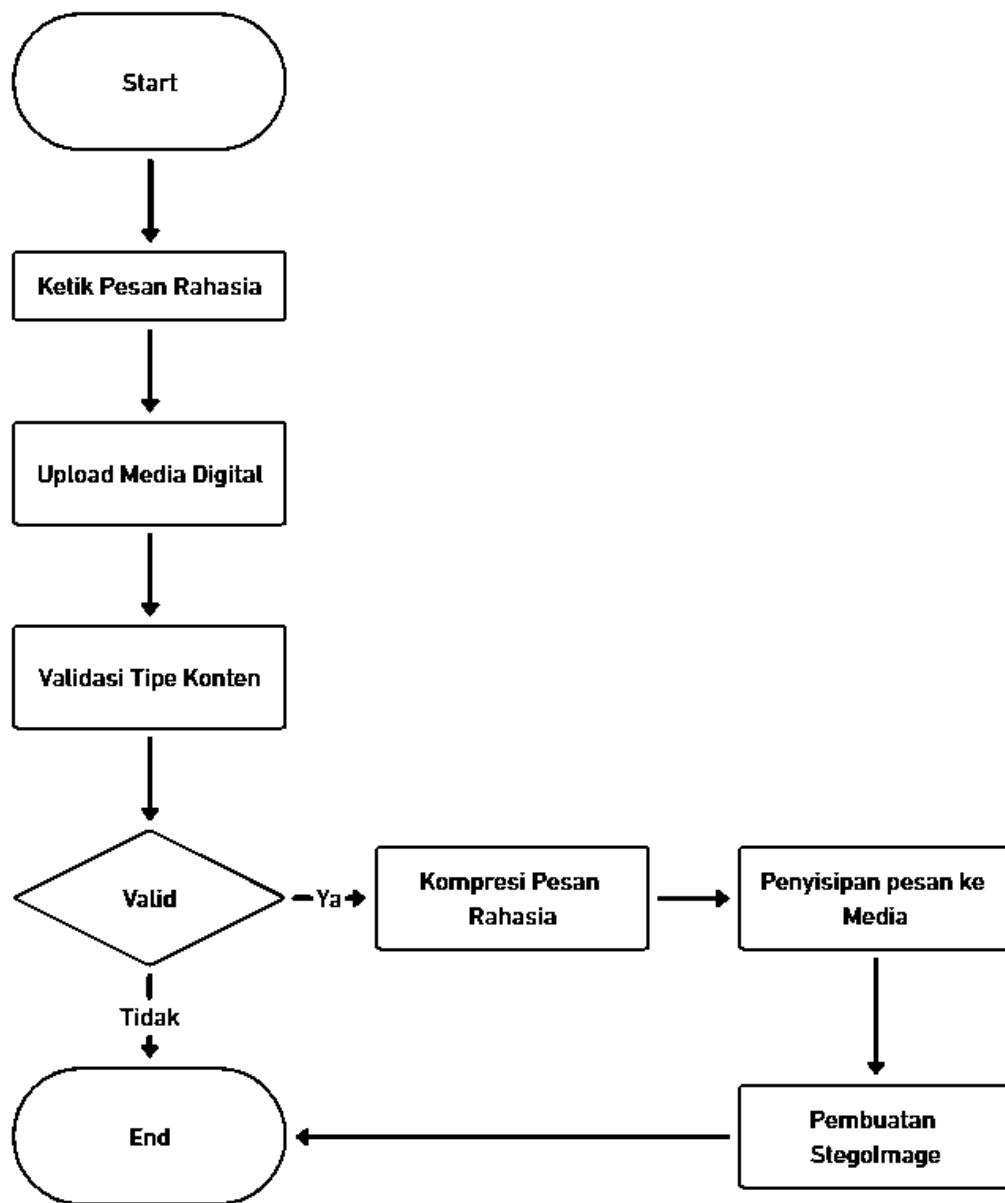
- pembuatan objek `invertedCodes` yang memetakan kode biner kembali ke simbol aslinya, mempermudah proses dekompresi *Huffman*.
- B. **Pembalikan Kode *Huffman*:** Kode *Huffman* yang telah dibangun selama proses kompresi dibalik untuk memetakan kode biner kembali ke simbol asli. Pembalikan ini menghasilkan objek `invertedCodes` yang digunakan untuk memudahkan pencocokan selama proses dekompresi. Setiap kode biner dalam `codes` dipetakan ke simbol asli, memungkinkan deteksi langsung dari kode biner selama dekompresi.
  - C. **Dekompresi Huffman:** *String* biner yang dikompresi diiterasi bit per bit. Setiap bit ditambahkan ke *string* code yang sementara. Ketika code cocok dengan salah satu entri di `invertedCodes`, simbol yang sesuai ditemukan. Jika simbol yang ditemukan adalah karakter, simbol tersebut ditambahkan ke `decodedItems` sebagai objek `{ char: symbol }`. Jika simbol adalah pasangan (*distance*, *length*), pasangan ini diurai dan ditambahkan ke `decodedItems` sebagai objek `{ distance: x, length: y }`.
  - D. **Dekompresi LZ77:** Dengan `decodedItems` yang berisi karakter dan pasangan (*distance*, *length*), proses dekompresi LZ77 dimulai. Untuk setiap *item* dalam `decodedItems`, jika *item* adalah karakter, karakter tersebut langsung ditambahkan ke *output*. Jika *item* adalah pasangan (*distance*, *length*), pola yang diwakili oleh pasangan ini diekstrak dari *output* yang sudah dibangun dan ditambahkan ke *output*. Proses ini memastikan rekonstruksi data asli yang benar.
  - E. **Mengembalikan Hasil Dekompresi:** Setelah semua *item* dalam `decodedItems` diproses, *string* *output* yang dihasilkan mengandung data asli yang tidak terkompresi. *output* kemudian dikembalikan sebagai hasil akhir dari proses dekompresi Deflate,

yang merekonstruksi data asli dari representasi biner yang terkompresi.

### 3.3. Rancangan Alur Sistem

Sistem *embedding* dan *extraction* pada rancangan rekayasa ini menggunakan algoritma *Least Significant Bits* (LSB) dibagi menjadi 2 proses utama yakni proses enkripsi dan proses dekripsi. Proses enkripsi maupun dekripsi akan melewati proses validasi berkas untuk selanjutnya sistem akan melakukan *embedding* atau *extraction* pada berkas gambar digital yang sudah diunggah. Adapun kedua alur proses tersebut seperti digambarkan sebagai berikut.

Proses *embedding* dimulai dengan pengguna mengunggah berkas berupa dokumen citra digital ke dalam sistem yang mana selanjutnya dimasukan pesan rahasia disertai dengan pemilihan algoritma. Selanjutnya sistem akan melakukan validasi tipe berkas untuk memastikan hanya berkas dokumen digital yang boleh diproses. Jika tipe berkas tidak sesuai maka proses akan berhenti dan sistem akan mengembalikan pesan gagal kepada pengguna, namun jika tipe berkas sesuai selanjutnya sistem akan melakukan proses kompresi pesan dan dilanjutkan proses penyisipan untuk kemudian akan dimasukan pesan terkompresi berupa *string* ke dalam citra yang kemudian siap untuk diunduh menjadi *stegoimage*.

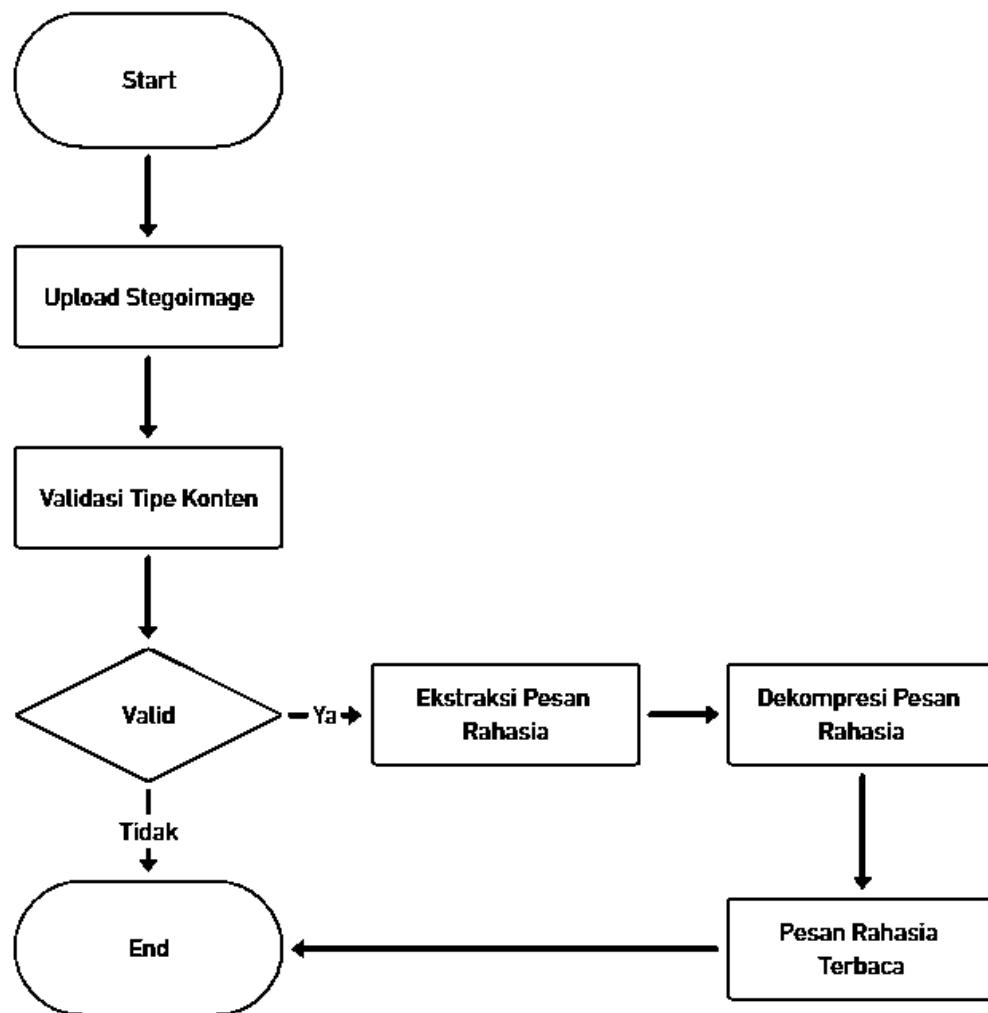


**Gambar 3.12 Diagram Alur Proses Penyisipan**

Sumber: diolah oleh peneliti

Kebalikannya pada saat proses dekripsi, mula-mula pengguna akan mengunggah berkas berupa citra digital, yang mana merupakan *stegoimage*, yang sudah berisi pesan rahasia terkompresi ke dalam sistem. Kemudian sistem akan melakukan validasi tipe berkas untuk memastikan hanya berkas citra digital yang boleh diproses. Jika tipe berkas tidak sesuai maka proses akan berhenti dan sistem akan mengembalikan pesan gagal kepada pengguna, namun jika tipe berkas sesuai selanjutnya sistem akan

menggunakan proses ekstraksi pesan rahasia dengan. Kemudian setelah didapatkan pesan rahasia, akan dilakukan proses dekompresi agar mendapatkan pesan sesungguhnya dari *stegoimage*. Terakhir pesan akan ditampilkan pada layar sistem.



**Gambar 3.13 Diagram Alur Proses Dekripsi**

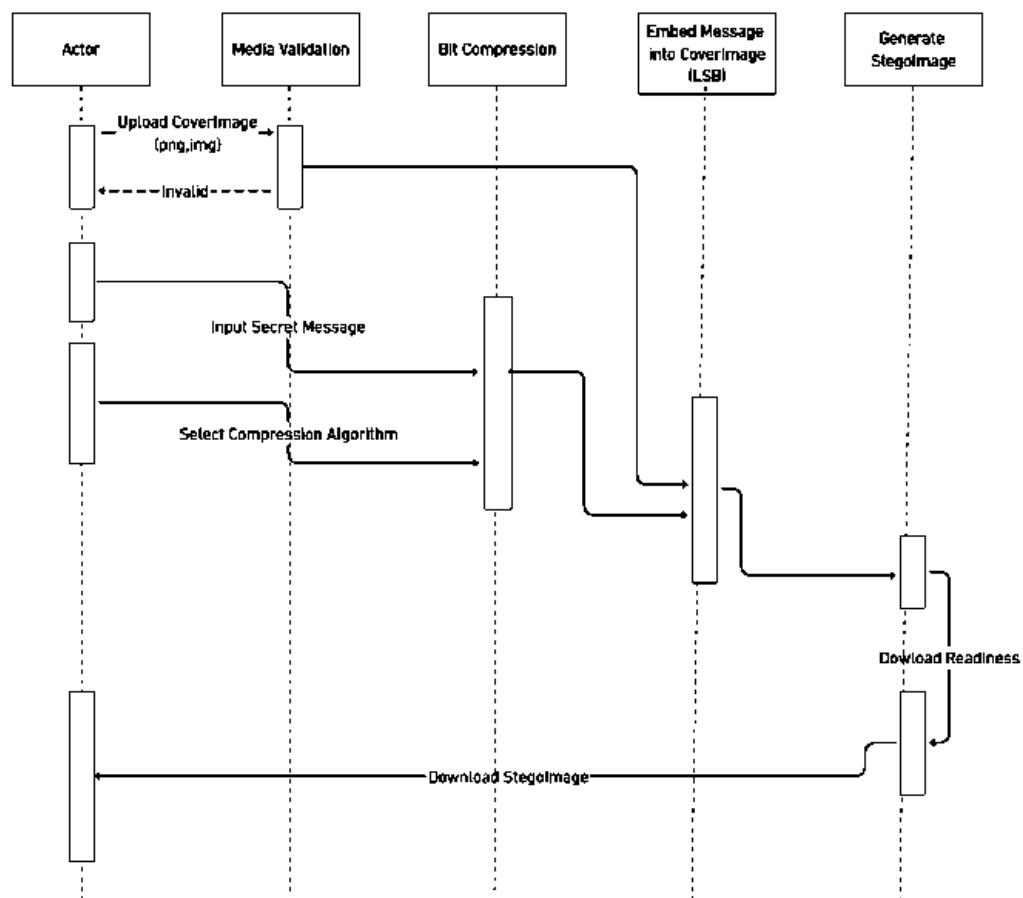
Sumber: diolah oleh peneliti

### 3.3.1 Rancangan Squence Diagram

*Sequence diagram* digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan keluaran tertentu, dan perubahan apa saja yang

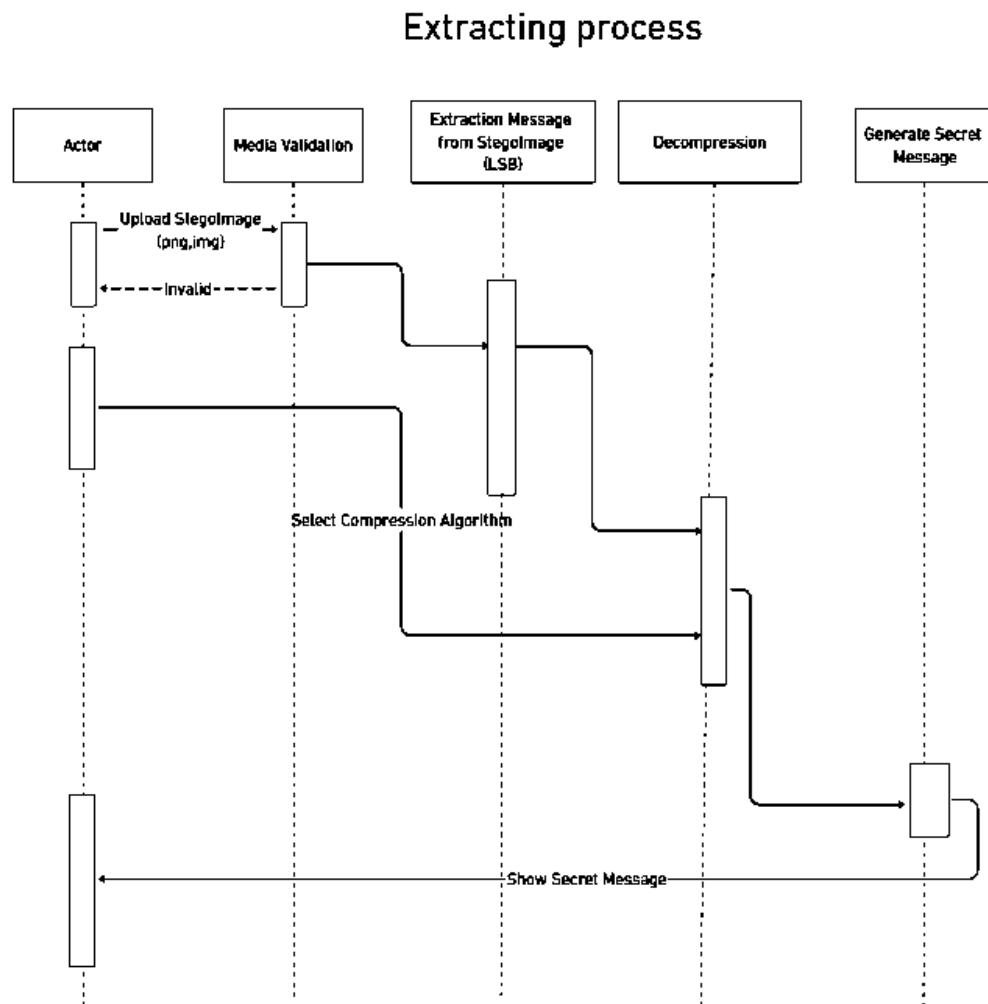
terjadi secara internal dan keluaran apa yang dihasilkan. Berikut adalah *sequence diagram* untuk rancangan sistem yang akan dibuat.

### Embedding process



**Gambar 3.14 Sequence Diagram Proses Embedding**

Sumber: diolah oleh peneliti



**Gambar 3.15 Sequence Diagram Proses Ekstraksi**

Sumber: diolah oleh peneliti

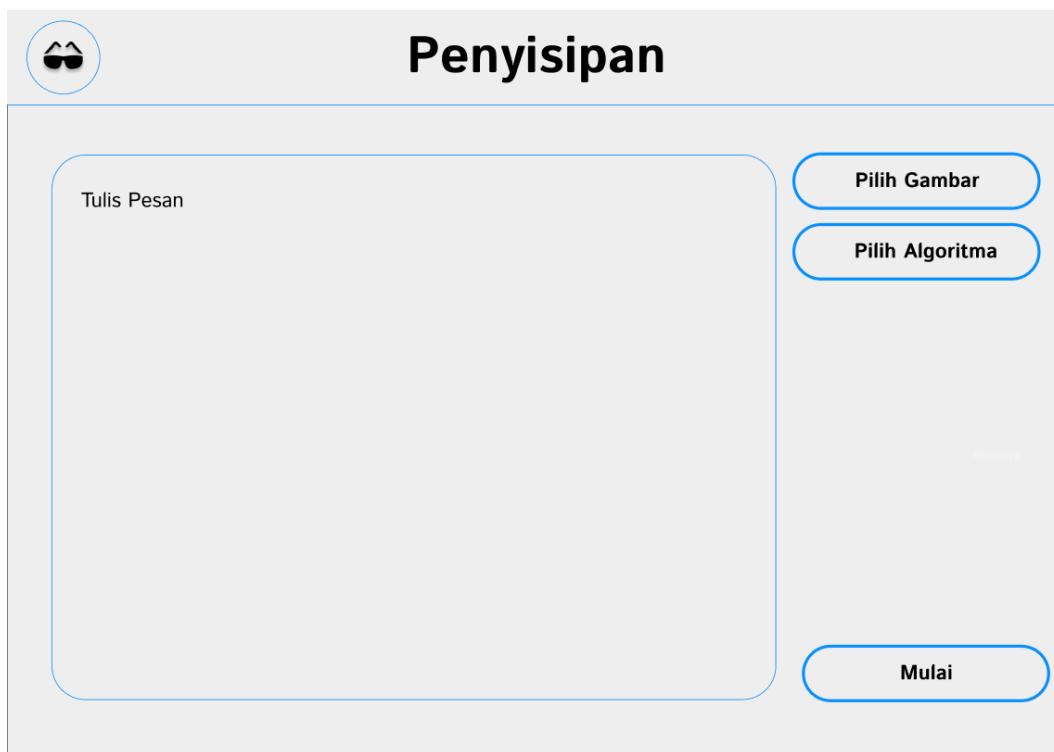
### 3.3.2 Rancangan Layar

Rancangan layar sangat penting dalam pembuatan sesuatu sistem. Itu sebabnya rancangan layar harus mudah dipahami oleh *user*, agar dalam memakai sistem *user* merasa aman serta tidak menghadapi kesusahan dalam mengoperasikannya. Dalam sistem ini, hendak ditafsirkan dengan rancangan layar dua tampilan yaitu rancangan layar *Embedding* dan layar *Extraction* di mana pada layar *Embedding* terdapat fungsi penyisipan pesan dengan memasukan pesan rahasia pada *textfield* yang disediakan lalu memilih *coverimage* yang akan digunakan lalu melakukan serta memilih

algoritma kompresi dengan *drop box* dan menekan *button* untuk memulai proses penyisipan. Serta layar *Extraction* terdapat fungsi pengekstraksian pesan rahasia dengan memilih algoritma kompresi yang digunakan.

### 3.3.2.1 Rancangan Layar *Embedding*

Fungsi layar *Embedding* yaitu, agar *user* dapat menyisipkan pesan rahasia kedalam *coverimage* yang mana nantinya digunakan untuk membuat *stegoimage*. Untuk dapat membuat *stegoimage* dimasukanlah masukan berupa teks rahasia serta gambar sebagai *coverimage* lalu kemudian dipilih algoritma kompresi untuk digunakan. Setelah tombol diklik maka akan langsung memulai proses penyisipan yang mana nanti akan mengunduh foto hasil dari penyisipan.



**Gambar 3.16 Rancangan Layar *Embedding***

Sumber: diolah oleh peneliti

### 3.3.2.2 Rancangan Layar *Extraction*

Fungsi layar *Extraction* yaitu, agar *user* dapat mengekstrak pesan rahasia dari *stegoimage* yang mana nantinya dapat informasi rahasia tersebut dapat terbaca kembali. Untuk kembali membaca pesan rahasia dari *stegoimage* dimasukanlah algoritma kompresi yang digunakan sebagai algoritma kompresi pesan. Lalu setelah algoritma dipilih, dimasukan *stegoimage* untuk nantinya pesan rahasia tersebut dapat terbaca. Setelah tombol diklik maka akan langsung memulai proses pengekstrasiyan yang mana nanti akan nampak pesan rahasia yang berada dalam *stegoimage* tersebut.



**Gambar 3.17 Rancangan Layar *Extraction***

Sumber: diolah oleh peneliti

### 3.4 Metode Perbandingan

#### 3.4.1 Komparasi Efisiensi Waktu Kompresi Dengan Grafik Dan Tabel

Proses pertama yang akan dilakukan adalah perbandingan waktu pengerjaan dari setiap algoritma berdasarkan waktu kompresi dan waktu dekompresi. Perhitungan akan dilakukan sebanyak masing-masing tiga kali untuk mendapatkan hasil yang objektif menggunakan fungsi perhitungan waktu. Selanjutnya hasil pengukuran akan dihitung rata-rata pengerjaannya dan di dapatkan hasil rata-rata yang dapat dicatat untuk dilaporkan. Kemudian hasil perhitungan divisualisasikan menggunakan grafik dan tabel untuk menentukan metode yang paling cepat berdasarkan hasil pengolahan.

#### 3.4.2 Komparasi Persentasi Kompresi Terhadap *Plaintext*

##### Menggunakan Metode Eksponensial

Langkah berikutnya adalah mengukur tingkat kompresi terhadap *plaintext*. Pengukuran ini dilakukan dengan menggunakan metode eksponensial yang mana perhitungannya akan dibantu dengan memberikan beban terhadap parameter yang berhubungan dengan perhitungan hasil kompresi. Parameter yang digunakan terdapat 4 hal yaitu: *ratio of compression*, *compression ratio*, *space savings*, dan *redundancy*. *Ratio of compression* dihitung menggunakan rumus berikut:

$$RC = \left( \frac{USK}{UA} \right) \times 100\% \quad (3.1)$$

Keterangan:

*RC* = *Ratio of compression*

*UA* = *Ukuran asli*

*USK* = *Ukuran Setelah Kompresi*

Pengukuran pada 3.1 ini akan menunjukkan sejauh apa algoritma dapat mengecilkan ukuran dari berkas. Kemudian selanjutnya perhitungan terhadap *compression ratio* yaitu berikut:

$$CR = \left( \frac{UA}{USK} \right) \times 100\% \quad (3. 2)$$

Keterangan:

*CR = compression ratio*

*UA = Ukuran asli*

*USK = Ukuran Setelah Kompresi*

Pengukuran pada 3.2 ini akan menunjukkan seberapa besar hasil algoritma dalam mengurangi ukuran data asli. Kemudian selanjutnya perhitungan terhadap *space savings* yaitu berikut:

$$SS = 100\%(UA) - CR \quad (3. 3)$$

Keterangan:

*CR = Compression Ratio*

*SS = Space Savings*

*UA = Ukuran asli*

Pengukuran pada 3.3 ini akan menunjukkan seberapa besar jumlah ruang yang dapat dihemat dengan menggunakan algoritma kompresi. Kemudian selanjutnya perhitungan terhadap *redundancy* yaitu berikut:

$$R = 100\% - CR \quad (3. 4)$$

Keterangan:

*CR = Compression Ratio*

*SS = Space Savings*

Pengukuran pada 3.4 ini akan menunjukkan seberapa besar jumlah perulangan yang dapat dihilangkan oleh suatu algoritma kompresi. Kemudian terakhir parameter-parameter di atas akan dimasukan ke dalam metode eksponensial berikut:

$$TNi = \sum_{j=1}^m (Vij) \times Bj \quad (3. 5)$$

Keterangan:

$TNi$  = Total nilai alternatif ke - i

$Vij$  = Derajat kepentingan relatif kriteria j pada keputusan i

$Bj$  =

Derajat kepentingan kriteria keputusan yang dinyatakan dengan bobot

$m$  = Jumlah kriteria keputusan

$n$  = Jumlah pilihan keputusan

$j = 1,2,3, \dots, m$  = Jumlah kriteria

$i = 1,2,3, \dots, n$  = Jumlah pilihan alternatif

Dari formula 3.5  $Vij$  merupakan 4 parameter yang sudah dijelaskan di atas serta  $Bj$  adalah beban setiap algoritma di mana beban dari setiap algoritma akan bergantung dari hasil perhitungan masing-masing.

### 3.4.3 Komparasi Persentasi Kompresi terhadap Medium

#### Menggunakan Grafik

Pada tahap ini, perbandingan dilakukan dengan menggunakan media penyimpanan sebagai acuan. Proses ini mirip dengan komparasi terhadap *plaintext*, tetapi perhitungannya dilakukan dengan membandingkan ukuran berkas media sebelum dan sesudah kompresi. Hal ini dilakukan untuk mengetahui efisiensi kompresi saat data disimpan dalam media. Terakhir perbandingan akan ditampilkan dengan menampilkan hasil datanya dalam suatu grafik perbandingan. Berikut formula yang digunakan dalam mengukur persentase kompresi terhadap medium:

$$PKM = \left( \frac{UDK}{UM} \right) \times 100\% \quad (3. 6)$$

Keterangan:

*PKM* =

*Persentase Kompresi Medium*

*UM* = *Ukuran Medium*

*UDK* = *Ukuran Data Kompresi*

Pada formula 3.6, perbandingan dilakukan dengan menggunakan media penyimpanan sebagai acuan. Perhitungan ini mirip dengan komparasi terhadap *plaintext*, tetapi perhitungannya dilakukan dengan membandingkan ukuran berkas media sebelum dan sesudah kompresi.

### 3.5 Tempat dan Waktu Penelitian

#### 3.5.1 Waktu Penelitian

Waktu yang digunakan peneliti untuk penelitian ini dilaksanakan sejak tanggal dikeluarkannya ijin penelitian dalam kurun waktu kurang lebih 6 (enam) bulan, 1 Bulan penggerjaan penelitian, 4 bulan pengumpulan data disertai dengan pengolahan dan pembangunan sistem, dan 1 bulan pengolahan data yang meliputi penyajian dalam bentuk skripsi dan proses bimbingan berlangsung. Berikut adalah timeline waktu penelitian yang penulis ajukan.

Tabel 3.1 Jadwal Penelitian

No	Kegiatan	2023	2024				
		Des	Jan	Feb	Mar	Apr	Mei
1.	Pembuatan Draft Skripsi						
2.	Proses Pembimbingan						
3.	Seminar Skripsi						
4.	Perbaikan Skripsi						
5.	Pemahaman algoritma kompresi						
6.	Penyusunan Algoritma Kompresi						
7.	Pembuatan Frontend						
8.	Pembuatan Backend						
9.	Integrasi Algoritma kompresi dengan Backend						
10.	Pengumpulan dan Pengolahan Data						
11.	Analisis Data						
12.	Penyusunan Laporan Skripsi						
13.	Seminar Hasil						
14.	Seminar Skripsi						
15.	Perbaikan Skripsi						
16.	Penyerahan Skripsi kepada Program Studi						

Sumber: diolah oleh peneliti

### 3.5.2 Tempat Penelitian

Tempat pelaksanaan penelitian ini adalah di lingkungan Fakultas Sains dan Teknologi Pertahanan, Universitas Pertahanan Indonesia. Berikut adalah detail tempat atau lokasi peneliti melakukan penelitian:

Fakultas Sains dan Teknologi Pertahanan

Alamat : Kawasan IPSC Sentul, Sukahati

Kecamatan : Citeureup

Kabupaten : Bogor

Provinsi : Jawa Barat

Kode pos : 16810

Telepon : (021) 87951555

### 3.6 Instrumen Penelitian

Pada penelitian ini membutuhkan instrumen penelitian yang memungkinkan seperti untuk mengumpulkan data, Analisis kebutuhan sistem yang akan dibangun, serta melaksanakan eksperimen dan yang terakhir adalah melakukan analisis hasil. Analisis kebutuhan sistem pada penelitian ini adalah sebagai berikut:

- a. Sistem ini berbasis web, diselenggarakan dengan localhost dalam membangun sistem ini, dan sistem ini akan dibangun menggunakan bahasa pemrograman javascript.
- b. Sistem dapat memberikan layanan *embedding* dan kompresi informasi atau data teks dari citra digital.
- c. Sistem dapat menyediakan layanan untuk *extraction* dan dekompresi informasi data ke dalam bentuk semula.

Tabel berikut menjelaskan persyaratan sistem untuk membangun sistem ini.

Tabel 3.2 Persyaratan Sistem

No	Fungsi	Kebutuhan
1	Layanan penyisipan data ke citra digital	<p>a. <i>Input</i></p> <ol style="list-style-type: none"> <li>1) Berkas penampung/<i>coverimage</i></li> <li>2) Informasi yang dirahasiakan</li> <li>3) Algoritma bit kompresi pilihan</li> </ol> <p>b. Proses</p> <ol style="list-style-type: none"> <li>1) Data rahasia dikompresi dengan algoritma kompresi bit yang telah dipilih.</li> <li>2) Hasil kompresi langsung disembunyikan ke dalam <i>coverimage</i> menggunakan metode LSB.</li> </ol>
		<p>a. <i>Output</i></p> <ol style="list-style-type: none"> <li>1) Pengguna mendapatkan keluaran berupa <i>stegoimage</i> yang berisi data rahasia yang berhasil dikompresi.</li> </ol>
2	Layanan ekstraksi data dari <i>stegoimage</i> .	<p>a. <i>Input</i></p> <ol style="list-style-type: none"> <li>1) <i>Stegoimage</i></li> <li>2) Algoritma bit kompresi pilihan</li> </ol> <p>b. Proses</p> <ol style="list-style-type: none"> <li>1) <i>Stegoimage</i> diekstrak menggunakan metode LSB yang kemudian pesan tersembunyi akan terbaca.</li> <li>2) Data rahasia didekompresi dengan algoritma kompresi bit yang telah dipilih.</li> </ol>

No	Fungsi	Kebutuhan
		<p>c. <i>Output</i></p> <p>1) Pengguna mendapatkan keluaran berupa data rahasia yang berhasil dikompresi.</p>

Sumber: diolah oleh peneliti

### 3.7 Pengumpulan Data

Pengumpulan data dilakukan dengan 6 metode kompresi berbeda di mana pada proses yang terjadi pada kompresi algoritma *Run-Length Encoding* yaitu dengan Masukan untuk proses ini adalah *string*, yang melalui beberapa langkah untuk membuat representasi run-length yang dikodekan. Awalnya, *string input* dipecah menjadi *array* karakter individual. Kemudian, operasi pengelompokan dilakukan menggunakan fungsi "grup", yang mengelompokkan karakter identik yang berurutan. Selanjutnya, setiap grup dipetakan ke sebuah *tuple* dalam bentuk [panjang, karakter], di mana "panjang" mewakili jumlah pengulangan karakter yang berurutan dalam grup tersebut. Keluaran akhirnya adalah *array* yang berisi *tuples* ini, yang secara efektif menyediakan representasi segmen yang disandikan run-length dalam *string* asli.

Kemudian pengumpulan data pada proses yang terjadi pada kompresi algoritma *Huffman Coding* yaitu dengan merubah *string* masukan diproses dengan menghitung frekuensi setiap karakter, yang kemudian digunakan untuk membangun pohon *Huffman*. Pohon ini dibangun dengan membuat antrian prioritas dimana setiap elemen merupakan pasangan yang terdiri dari frekuensi karakter dan karakter itu sendiri. Algoritma ini secara berulang menggabungkan dua elemen dengan frekuensi terendah dalam antrian prioritas untuk membuat elemen baru yang mewakili frekuensi gabungannya dan simpul pohon biner. Pengkodean untuk setiap karakter dihasilkan dengan menelusuri pohon Huffman. Terakhir, *string*

masukan dikodekan menjadi *string* biner menggunakan pengkodean yang dihasilkan. keluarannya adalah *string* panjang hasil dari pohon huffman.

Ketiga pada proses pengumpulan data dalam algoritma kompresi *Arithmetic Coding*, pengambilan dilakukan dari sebuah *Map* yang disebut 'freq' sebagai *input*, di mana kunci-kunci mewakili karakter-karakter dan nilai-nilai yang sesuai menunjukkan frekuensinya. Algoritma ini secara iteratif memeriksa setiap karakter ASCII dalam rentang 0 hingga 255. Untuk setiap karakter dalam rentang ini, algoritma memeriksa apakah karakter tersebut terdapat sebagai kunci dalam *Map* "freq". Jika karakter tersebut ditemukan, algoritma akan menghitung frekuensi kumulatif dengan menambahkan frekuensi karakter saat ini ke total yang sedang diakumulasi. Hasilnya adalah pembentukan sebuah *Map* baru yang disebut "cf", di mana setiap kunci merepresentasikan sebuah karakter dan nilai terkaitnya merepresentasikan frekuensi kumulatif hingga karakter tertentu tersebut. Proses ini pada dasarnya menghasilkan pemetaan karakter ke frekuensi kumulatifnya, yang dapat dimanfaatkan lebih lanjut dalam berbagai aplikasi.

Selanjutnya pada metode yang ke empat ketika melakukan pengumpulan data pada proses kompresi algoritma *Lempel-Ziv-Welch*, langkah pertama adalah menggunakan string masukan. Proses ini dimulai dengan inisialisasi kamus yang memetakan setiap karakter tunggal ke nilai ASCII-nya. Algoritma kemudian mengiterasi melalui string masukan, secara berkelanjutan menggabungkan karakter untuk memeriksa apakah kombinasi tersebut sudah ada dalam kamus. Apabila suatu kombinasi tidak terdaftar dalam kamus, algoritma akan menambahkannya ke kamus dan menginisialisasi kata baru untuk dimulai. Selama proses berlangsung, algoritma melacak urutan bersama dengan kode terkaitnya dalam kamus. *Output* akhir dari proses ini adalah *array* yang berisi kode-kode ini, yang efektif merepresentasikan data terkompresi yang dihasilkan dari string masukan awal.

Yang ke lima pengumpulan data dalam proses kompresi algoritma *J-bit Encoding* dimulai dengan membaca masukan per *byte*. Ketika *byte* yang

dibaca adalah nol, bit '0' ditambahkan ke *temporary byte* data. Jika *byte* yang dibaca bukan nol, *byte* tersebut disalin ke data I, dan bit '1' ditambahkan ke *temporary byte* data. Proses ini berulang hingga *temporary byte* data terisi dengan 8 bit. Setelah itu, nilai *byte* dari *temporary byte* data dipindahkan ke data II, dan *temporary byte* data direset. Langkah-langkah ini diulang hingga mencapai akhir *file*. *Output* dari proses ini melibatkan kombinasi dari data I dan data II, yang terdiri dari panjang masukan asli serta data I dan data II yang terkompresi.

Terakhir pengumpulan data pada proses yang terjadi pada kompresi algoritma *Deflate Compression* yaitu dengan memanfaatkan algoritma LZ77 dan pembuatan pohon Huffman. Kompresi LZ77 melibatkan pemeliharaan jendela geser pada data yang diproses untuk mengidentifikasi pola berulang atau urutan *byte*. Ketika pola tersebut terdeteksi, Deflate menyimpan *pointer* ke kemunculan awal pola di dalam jendela dan panjang urutan yang berulang. Jika tidak ada pengulangan yang ditemukan atau tidak layak dikodekan dengan *pointer*, data disimpan sebagai nilai literal. Setelah langkah LZ77, *Deflate* melakukan analisis frekuensi terhadap semua pasangan literal dan panjang penunjuk. Analisis ini mengarah pada konstruksi pohon Huffman, di mana *item* yang lebih sering diberi kode lebih pendek, sedangkan *item* yang lebih jarang diberi kode lebih panjang. Selanjutnya, pasangan literal dan panjang penunjuk dikodekan menggunakan pohon *Huffman* ini, menyelesaikan proses kompresi.

Kemudian penelitian ini juga akan menggunakan teknik pengumpulan data pengujian eksperimental di mana algoritma kompresi akan dijalankan pada kumpulan data dalam hal ini *string* pesan rahasia serta dokumen *stegoimage*. Setiap algoritma akan diuji dalam kondisi yang sama untuk memastikan perbandingan yang adil. Parameter yang akan diukur meliputi:

- a. Kecepatan: Waktu yang dibutuhkan untuk proses kompresi dan dekompresi.

- b. Persentase Kompresi: Perbandingan antara ukuran *file* asli dan ukuran *file* setelah kompresi.
- c. Jumlah Bit yang Terkompresi: Detail tentang bagaimana data dikompresi, termasuk jumlah bit atau informasi yang dikurangi.

### **3.8 Pengolahan Data**

Dalam bagian ini, peneliti akan menjelaskan langkah-langkah yang digunakan untuk menganalisis data yang diperoleh dari eksperimen pengujian performa berbagai algoritma kompresi bit. Setelah pengumpulan data eksperimental, proses selanjutnya adalah melakukan analisis kuantitatif. Teknik ini melibatkan penggunaan statistik dan metode numerik untuk menganalisis dan menginterpretasikan hasil yang diperoleh. Terdapat beberapa opsi dalam melakukan proses analisa seperti dapat menggunakan grafik, tabel, dan metode statistik untuk menampilkan perbandingan kinerja antar algoritma. Tujuan utama dari pengolahan data ini adalah untuk menentukan algoritma mana yang paling efisien berdasarkan waktu eksekusi, persentase kompresi, dan jumlah bit terhadap medium.

#### **3.8.1 Pengukuran Waktu Eksekusi**

Waktu eksekusi algoritma diukur untuk menentukan efisiensi waktu. Pengukuran ini dilakukan dengan menggunakan fungsi penghitung waktu yang merekam durasi yang dibutuhkan oleh setiap algoritma untuk menyelesaikan proses kompresi data. Durasi waktu yang direkam ini kemudian akan digunakan untuk membandingkan kecepatan relatif dari setiap algoritma kompresi. Kemudian akan dilakukan perekaman 3 kali untuk setiap algoritma baik dalam kompresi maupun dekompresi yang mana hasilnya akan dirata-rata untuk kemudian dijadikan hasil perhitungan akhir dalam penentuan kecepatan waktu eksekusi.

#### **3.8.2 Kalkulasi Persentase Kompresi**

Persentase kompresi dihitung sebagai ukuran efektivitas algoritma dalam mengurangi ukuran data. Ini dihitung dengan membandingkan ukuran data sebelum dan sesudah kompresi. Perhitungan dijalankan dengan mencari nilai *ratio of compression*, *compression ratio*, *space savings*, dan *redundancy* yang kemudian nilai-nilai tersebut akan dimasukkan dalam metode perbandingan eksponensial untuk mendapatkan preferensi yang dapat dijadikan acuan pengambilan kesimpulan algoritma terbaik.

### **3.8.3 Penghitungan Jumlah Bit Terhadap Medium**

Perhitungan terakhir yang dilakukan adalah perhitungan atas jumlah bit yang menjadi acuan atau *benchmark* mau pun data yang sudah terkompresi terhadap medium. Perbandingan ini dilakukan

### **3.8.4 Analisis Data**

Data yang dikumpulkan dari pengujian di atas akan dianalisis menggunakan metode statistik. Analisis ini meliputi:

- a. Visualisasi Data: Membuat grafik dan tabel untuk memvisualisasikan perbandingan performa algoritma.
- b. Analisis Komparatif: Membandingkan performa algoritma berdasarkan metrik yang telah diukur untuk menentukan algoritma mana yang paling efisien dan efektif dalam konteks yang diberikan. Analisis ini dibantu dengan perhitungan metode eksponensial yang akan menentukan tingkat dari setiap algoritma.

### **3.8.5 Dokumentasi**

Seluruh proses pengolahan data, mulai dari pengumpulan data hingga analisis hasil, akan didokumentasikan secara rinci. Dokumentasi ini akan memungkinkan proses pengulangan studi dan akan memastikan transparansi dalam penelitian yang akan dimasukkan pada sub bab Hasil di bab 4.

## **BAB IV**

### **HASIL PENELITIAN DAN PEMBAHASAN**

#### **4.1 Masukan Data Pengujian**

Masukan data merupakan nilai-nilai yang dimasukkan ke dalam sistem sehingga nantinya dapat diolah oleh fungsi-fungsi yang ada di dalam sistem. Nilai-nilai yang akan dimasukkan merupakan kalimat yang mengandung pesan rahasia yang dimaksudkan untuk memberikan perintah atau informasi kepada orang lain dengan Masukan yang diberikan dibagi menjadi 3 variasi yaitu variasi 1 paragraf, 5 paragraf, dan 10 paragraf.

##### **A. 1 Paragraf**

*"It was dark before dawn on a freezing Tokyo winter's night. In silence the soldiers prepared their rifles and bayonets. The officers strapped on their greatcoats and swords. They filed in silence out of their barracks and marched down the snow-covered streets. The contingent halted as they reached a beautiful traditional wooden mansion. There was a guard on the gate, and he tried to stop them, but the soldiers knocked him rudely aside and burst past. The guard telephoned desperately for help, but it was too late. The officer barked an order and the soldiers smashed down the front door. They stormed through the house. In a bedroom two officers found an old man asleep in bed. They drew their swords and pistols."*

Pada variasi ini jumlah bit yang terkompresikan berjumlah 5744 bit. Hal ini merupakan tolok ukur yang akan digunakan dalam menilai kinerja dari algoritma kompresi yang akan diterapkan terhadap teks 1 paragraf ini. Jika hasil lebih rendah, hal tersebut menandakan algoritma kompresi yang digunakan lebih baik dibandingkan tidak menggunakan kompresi. Hal sebaliknya juga berlaku apabila hasil bit lebih besar dibanding tolok ukur pada bit variasi 1 paragraf.

## B. 5 Paragraph

*"It was dark before dawn on a freezing Tokyo winter's night. In silence the soldiers prepared their rifles and bayonets. The officers strapped on their greatcoats and swords. They filed in silence out of their barracks and marched down the snow-covered streets. The contingent halted as they reached a beautiful traditional wooden mansion. There was a guard on the gate, and he tried to stop them, but the soldiers knocked him rudely aside and burst past. The guard telephoned desperately for help, but it was too late. The officer barked an order and the soldiers smashed down the front door. They stormed through the house. In a bedroom two officers found an old man asleep in bed. They drew their swords and pistols.*

*The old man was the venerable statesman Takahashi Korekiyo. He had been Prime Minister of Japan and Minister of Finance seven times. This was the man who prevented his country collapsing into the Great Depression by the combined use of modern fiscal, monetary, and exchange rate policies, perhaps the first in the world to design such a policy package. By 1934, before such policies had been articulated in economic theory, Takahashi had put them into practice. In doing so he had helped Japan build a strong economy. But sadly this economic strength had been used by Japan to build and abuse its military might for war, resulting in the ultimate misery of millions. Takahashi was the only Japanese politician brave enough to stand up to the military and fight against the inexorable slide to militarism in the 1930s, designing a modern disciplined fiscal policy which would put a brake on excessive military spending by 1935. For this he would pay the ultimate price.*

*Takahashi had learned his craft from his experience in the two major conflicts that had disrupted the region at the turn of the century, and had set the scene for the twentieth century ahead. In the first Sino-Japanese War of 1894-5, Japan dealt a humiliating blow to the*

*declining Chinese Qing Empire and dominated Korea as a client state. The second conflict happened a decade later, when in 1904-5 Japan convincingly defeated the Russian fleets, extended its influence in Manchuria, and bolstered its military.*

*Takahashi was a quite unique Japanese gentleman of the time, and his life had been an eventful one. He had been born in Edo (the old name for Tokyo) in 1854 and had an unusual upbringing in a country that was undergoing wrenching changes. His father, Morifusa, was a landscape painter in the shogun court at Edo Castle. He was a big-eating, big-drinking, party-going man, famous for his intake of large amounts of sake. Despite having two wives, several offspring and being advanced in age, he impregnated a pretty 16-year-old family maid named Kim, who gave birth to a boy named Wakiji. Wakiji was adopted out to a nearby family named Takahashi, a member of the lowest rank of the samurai clan (effectively the foot soldiers of the Tokugawa regime). There the boy was renamed Takahashi, and brought up by Kiyoko, the widowed grand-mother of the house, who was to exert an important influence over him.*

*In the year of Takahashi's birth, US Commander Perry's ships had breached the self-imposed Japanese foreign blockade, opening up the country, and sparking off major changes. The ruling shogunate gave way to the reinstatement of the Meiji Emperor, recognizing the need to deal with the outside world and to learn lessons from the West. Grandmother Kiyoko was herself a rational, independent, and modernizing person, and she raised Takahashi in this disrupted world. The young boy grew up cute, precocious, and a prodigious learner. He attracted attention from a modernizing senior samurai who sent him, aged only ten, to study with a missionary's wife. Takahashi learned English very fast and very well, engaging enthusiastically with the foreigners he came into contact with. Still aged only 11 he was*

*appointed as a houseboy in the British-owned Chartered Mercantile Bank in Yokohama, the start of his long and colourful financial career."*

Pada variasi ini jumlah bit yang terkompresikan berjumlah 32408 bit. Hal ini merupakan tolok ukur yang akan digunakan dalam menilai kinerja dari algoritma kompresi yang akan diterapkan terhadap teks 5 paragraf ini. Jika hasil lebih rendah, hal tersebut menandakan algoritma kompresi yang digunakan lebih baik dibandingkan tidak menggunakan kompresi. Hal sebaliknya juga berlaku apabila hasil bit lebih besar disbanding tolok ukur pada bit variasi 5 paragraf.

#### C. 10 Paragraf

*"It was dark before dawn on a freezing Tokyo winter's night. In silence the soldiers prepared their rifles and bayonets. The officers strapped on their greatcoats and swords. They filed in silence out of their barracks and marched down the snow-covered streets. The contingent halted as they reached a beautiful traditional wooden mansion. There was a guard on the gate, and he tried to stop them, but the soldiers knocked him rudely aside and burst past. The guard telephoned desperately for help, but it was too late. The officer barked an order and the soldiers smashed down the front door. They stormed through the house. In a bedroom two officers found an old man asleep in bed. They drew their swords and pistols.*

*The old man was the venerable statesman Takahashi Korekiyo. He had been Prime Minister of Japan and Minister of Finance seven times. This was the man who prevented his country collapsing into the Great Depression by the combined use of modern fiscal, monetary, and exchange rate policies, perhaps the first in the world to design such a policy package. By 1934, before such policies had been articulated in economic theory, Takahashi had put them into practice. In doing so he had helped Japan build a strong economy. But sadly this economic strength had been used by Japan to build and abuse its military might for war, resulting in the ultimate misery of millions.*

*Takahashi was the only Japanese politician brave enough to stand up to the military and fight against the inexorable slide to militarism in the 1930s, designing a modern disciplined fiscal policy which would put a brake on excessive military spending by 1935. For this he would pay the ultimate price.*

*Takahashi had learned his craft from his experience in the two major conflicts that had disrupted the region at the turn of the century, and had set the scene for the twentieth century ahead. In the first Sino-Japanese War of 1894-5, Japan dealt a humiliating blow to the declining Chinese Qing Empire and dominated Korea as a client state. The second conflict happened a decade later, when in 1904-5 Japan convincingly defeated the Russian fleets, extended its influence in Manchuria, and bolstered its military.*

*Takahashi was a quite unique Japanese gentleman of the time, and his life had been an eventful one. He had been born in Edo (the old name for Tokyo) in 1854 and had an unusual upbringing in a country that was undergoing wrenching changes. His father, Morifusa, was a landscape painter in the shogun court at Edo Castle. He was a big-eating, big-drinking, party-going man, famous for his intake of large amounts of sake. Despite having two wives, several offspring and being advanced in age, he impregnated a pretty 16-year-old family maid named Kim, who gave birth to a boy named Wakiji. Wakiji was adopted out to a nearby family named Takahashi, a member of the lowest rank of the samurai clan (effectively the foot soldiers of the Tokugawa regime). There the boy was renamed Takahashi, and brought up by Kiyoko, the widowed grand-mother of the house, who was to exert an important influence over him.*

*In the year of Takahashi's birth, US Commander Perry's ships had breached the self-imposed Japanese foreign blockade, opening up the country, and sparking off major changes. The ruling shogunate gave way to the reinstatement of the Meiji Emperor, recognizing the*

*need to deal with the outside world and to learn lessons from the West. Grandmother Kiyoko was herself a rational, independent, and modernizing person, and she raised Takahashi in this disrupted world. The young boy grew up cute, precocious, and a prodigious learner. He attracted attention from a modernizing senior samurai who sent him, aged only ten, to study with a missionary's wife. Takahashi learned English very fast and very well, engaging enthusiastically with the foreigners he came into contact with. Still aged only 11 he was appointed as a houseboy in the British-owned Chartered Mercantile Bank in Yokohama, the start of his long and colourful financial career.*

*Most Japanese youths were brought up to be quiet and respectful, but not Takahashi. He was very sociable from a young age. This helped him to learn languages, but like his natural father, he also liked to party, and that soon got him into trouble-drinking, gambling, and joining gangs in Tokyo that preyed on prostitutes. He took up with one of these latter, a young geisha. His family tried to discipline him, but young Takahashi proved irrepressible.*

*He yearned to travel to see something of the modern world outside Japan, a most unusual possibility for a young Japanese boy. In 1867, still aged only 13, he convinced one of the samurai clan to arrange for him to travel to the US. His grandmother Kiyoko did her best to prepare him for the journey ahead. She even presented him with his grandfather's samurai dagger and taught him how to commit ritual suicide by disem- bowelling himself, in case he might find this necessary.*

*It was a rough and difficult voyage crossing the Pacific. Takahashi and another young Japanese travelling companion kept up their spirits by playing pranks on other passengers, which frequently got them into trouble aboard the ship. At last they docked in San Francisco. Takahashi had expected that he would get the opportunity to go to school, but that never happened. Instead he ended up working*

*as a houseboy in San Francisco. Then without realizing what he was doing, he signed a contract committing himself as an indentured servant on a farm in Oakland for some years. He soon had a raging argument with a Chinese cook from the farm who tried to kill him with an axe. The young Takahashi was disciplined by his employer. He tried to run away from the farm, only to be told that he was not allowed to leave. Later he was sold on to another family, a life he described as little better than that of a slave.*

*Never one willing to be ordered around by others, Takahashi waited for his opportunity to escape, and at last ran away to the Pacific coast. In 1868 he managed to board a ship and head back home to Japan. There he found a country that was in turmoil because the shogun lords were rising up against the new coalition civilian government. The rebellious young Takahashi and his friends joined a local samurai group, and were caught up in opposition to government. For a time they had to flee Tokyo and hide.*

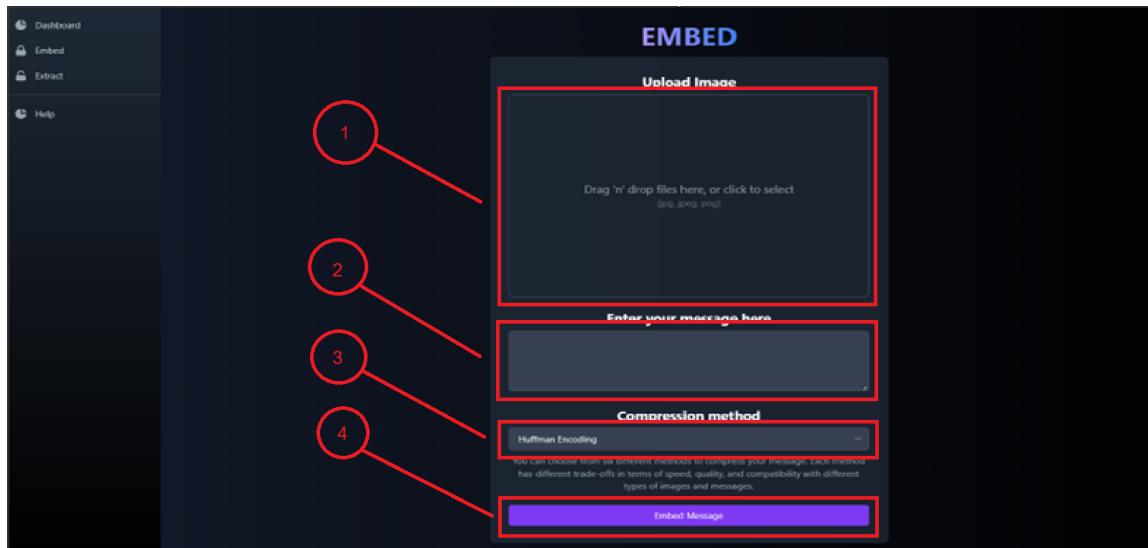
*Eventually Takahashi Korekiyo settled down to life back in Tokyo. He was offered teaching jobs in missionary schools, where he began first by instructing in English, then extending his teaching into other subjects. He was completely untutored himself, but his outgoing manner and linguistic skills made him an excellent teacher, and no new subject was too difficult for him to learn. However his life continued to drift between missionary discipline and bouts of drunkenness with geisha and gangs."*

Pada variasi ini jumlah bit yang terkompresikan berjumlah 55240 bit. Hal ini merupakan tolok ukur yang akan digunakan dalam menilai kinerja dari algoritma kompresi yang akan diterapkan terhadap teks 10 paragraf ini. Jika hasil lebih rendah, hal tersebut menandakan algoritma kompresi yang digunakan lebih baik dibandingkan tidak menggunakan kompresi. Hal sebaliknya juga berlaku apabila hasil bit lebih besar dibanding tolok ukur pada bit variasi 10 paragraf.

## 4.2 Implementasi Web

### 4.2.1 Penyisipan Gambar

Gambar 4.1 menunjukkan laman *embed* dalam aplikasi steganografi berbasis web. Terdapat tiga masukan yang diperlukan yaitu gambar, pesan, dan juga pemilihan algoritma kompresi yang mana setelah memasukan diberikan terdapat tombol *Embed Message* untuk memulai steganografi.



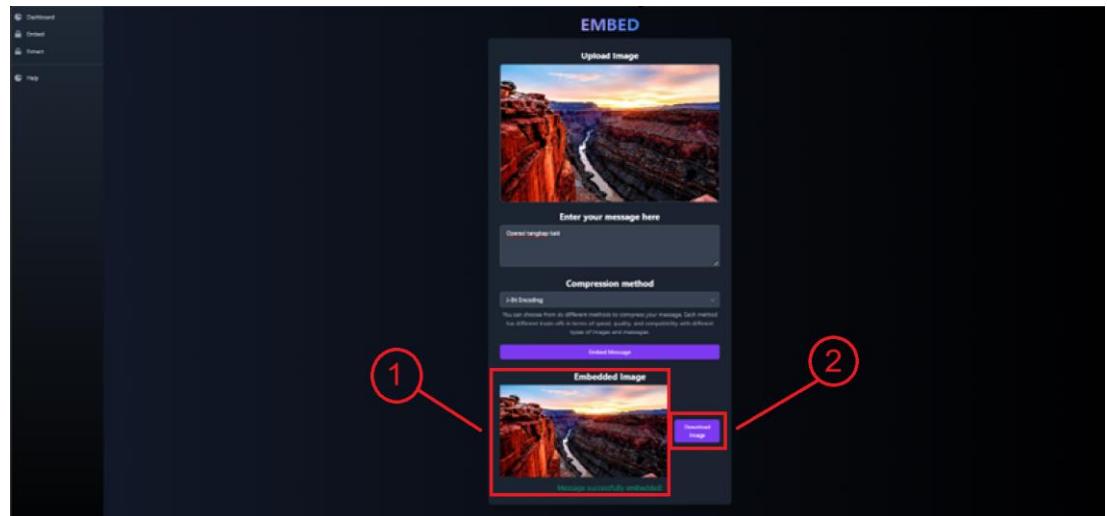
**Gambar 4.1 Gambar Laman Embed**

Sumber: diolah oleh peneliti

Pada gambar 4.1 terdapat beberapa fungsi dari komponen yang ada, berikut penjelasan dari tiap-tiap komponen:

1. Memilih gambar yang akan digunakan untuk disisipkan pesan
2. Memasukkan pesan yang akan disisipkan ke dalam gambar
3. Memilih metode kompresi untuk menyisipkan pesan ke gambar
4. *Submit* untuk memproses penyisipan gambar

Setelah masukan diberikan laman akan berubah seperti pada gambar 4.2. Gambar tersebut menunjukkan terdapat dua bagian yang cukup krusial yaitu bagian contoh gambar yang akan diunduh dan juga tombol *Download Image*. Berdasarkan dari algoritma yang digunakan maka akan muncul kotak tambahan yang akan ditampilkan pada bagian demo.



**Gambar 4.2 Gambar Laman Embed Setelah Klik**

Sumber: diolah oleh peneliti

Pada gambar 4.2 terdapat beberapa fungsi dari komponen yang ada, berikut penjelasan dari tiap-tiap komponen:

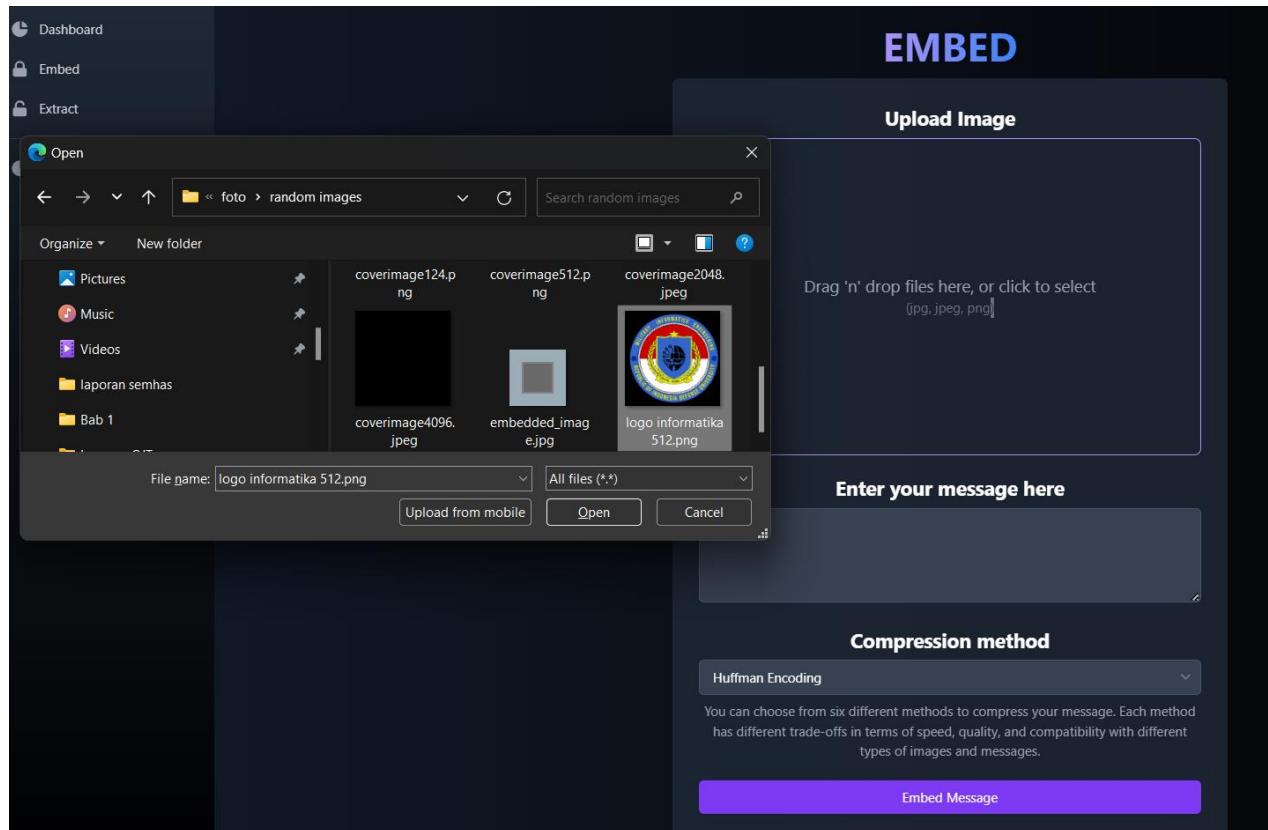
1. Tempat dimunculkan *preview* terhadap gambar hasil penyisipan dan siap diunduh
2. Memulai proses pengunduhan

Berikut merupakan tahapan pengejaan proses steganografi dari mulai mengunggah gambar hingga mengunduhnya ke dalam sistem. Proses akan dibagi menjadi enam metode. Setiap metode yang dikerjakan akan menggunakan gambar dan teks sederhana agar memperjelas detail dari gambar yang dimasukan.

#### **4.2.1.1 Run-Length Encoding**

- 1) Unggah *coverimage*

Tahapan pertama dalam melakukan proses steganografi adalah dengan mengunggah gambar yang akan peneliti pilih sebagai media penampung pesan. Caranya bisa dengan menahan dan menggeser gambar ke kotak *upload image* atau dapat juga dengan mengklik kotak *upload image* dan memilih gambar yang peneliti inginkan.

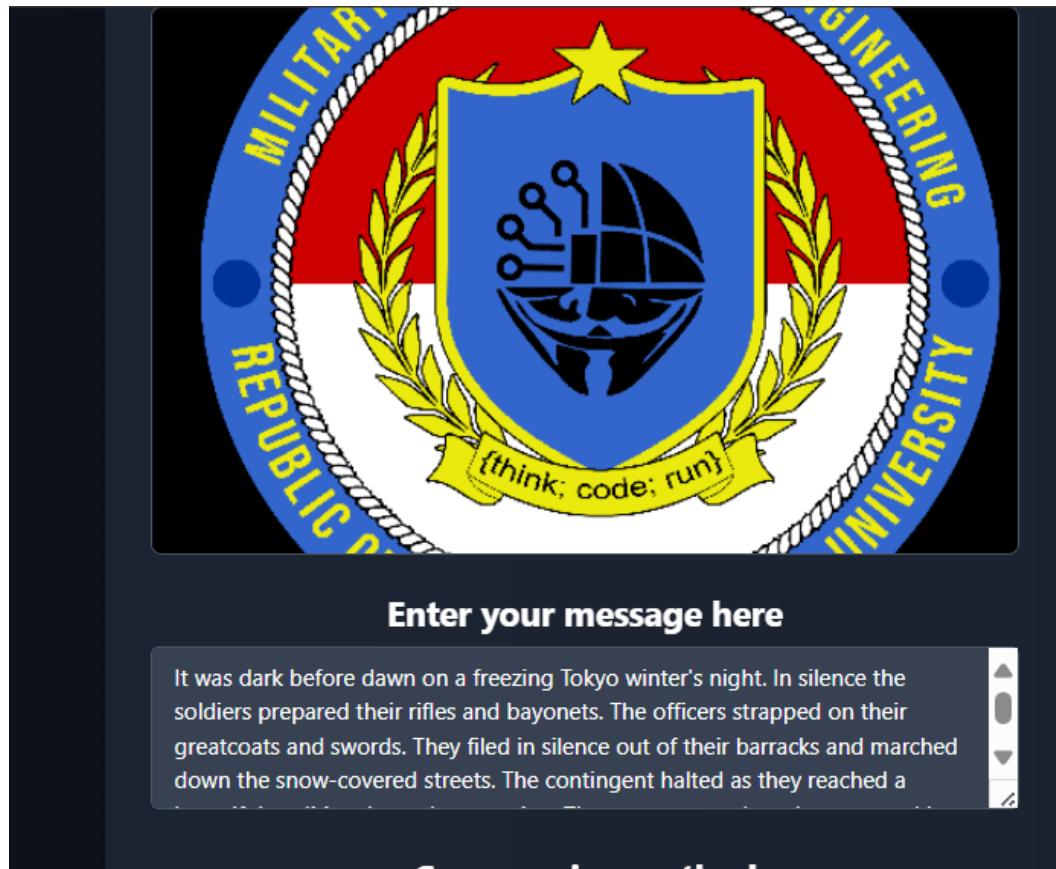


**Gambar 4.3 Gambar Unggah RLE**

Sumber: diolah oleh peneliti

2) Masukkan pesan rahasia

Ketik pesan rahasia yang ingin disisipkan ke dalam kotak di bawah tulisan "*Enter your message here*".

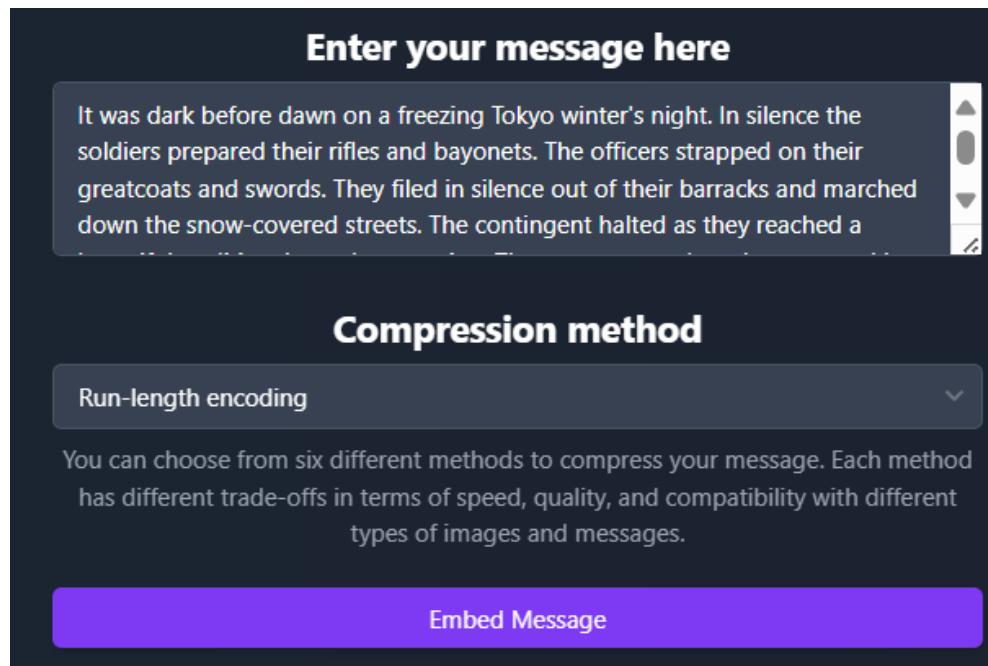


Gambar 4.4 Gambar Pesan Rahasia RLE

Sumber: diolah oleh peneliti

3) Pilih metode kompresi

Pilih metode kompresi yang diinginkan. Pada bagian ini, karena peneliti akan mencoba menggunakan *Run-Length Encoding* maka peneliti pilih metode tersebut di kotak *compression method*. Setelah itu klik tombol *Embed Message*.

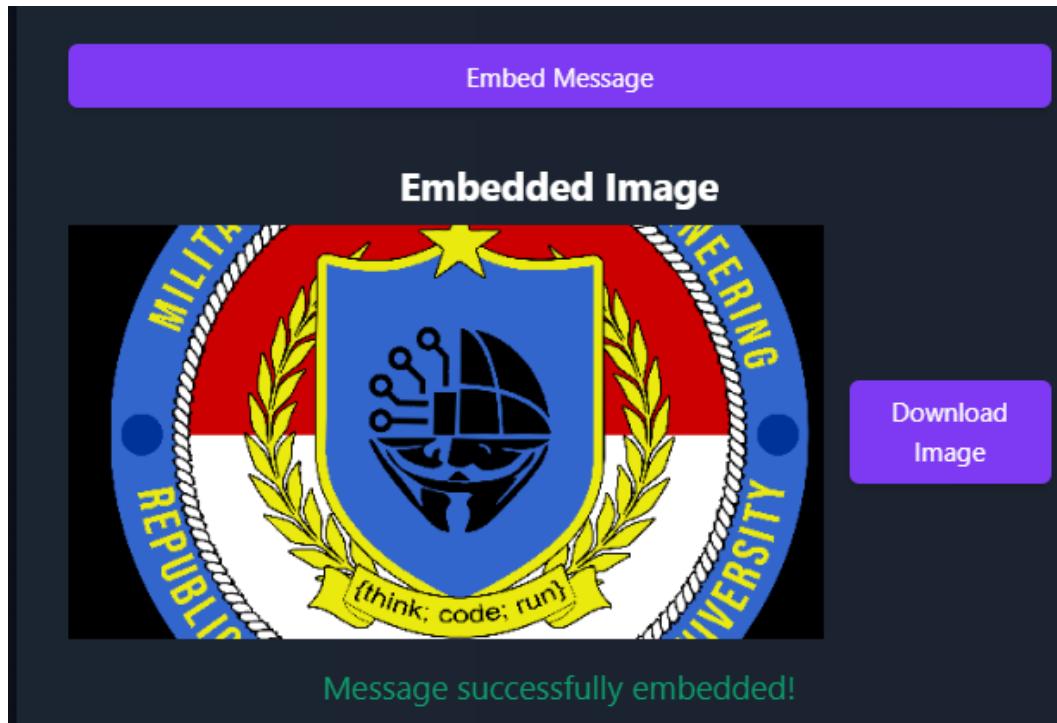


**Gambar 4.5 Gambar Pilih Metode RLE**

Sumber: diolah oleh peneliti

4) Unduh *stegoimage*

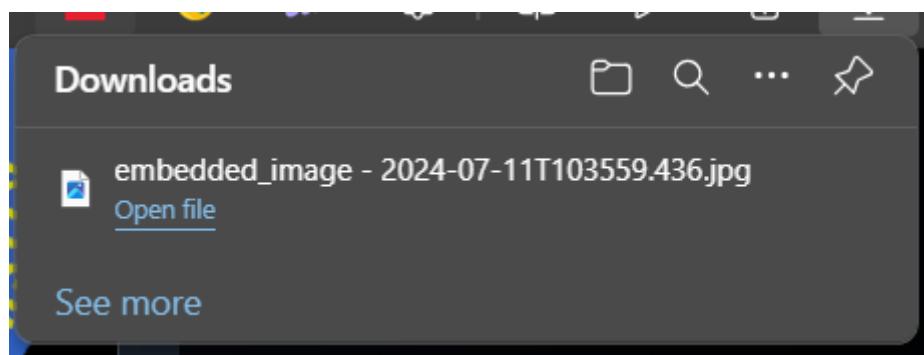
Untuk mengunduh gambar yang sudah disisipkan peneliti hanya perlu mengklik tombol *Download Image* yang berada di kanan ulasan gambar *stegoimage*.



**Gambar 4.6 Gambar Unduh Stegoimage RLE**

Sumber: diolah oleh peneliti

Gambar akan secara langsung terunduh ke dalam gawai yang digunakan.



**Gambar 4.7 Gambar Stegoimage Terunduh RLE**

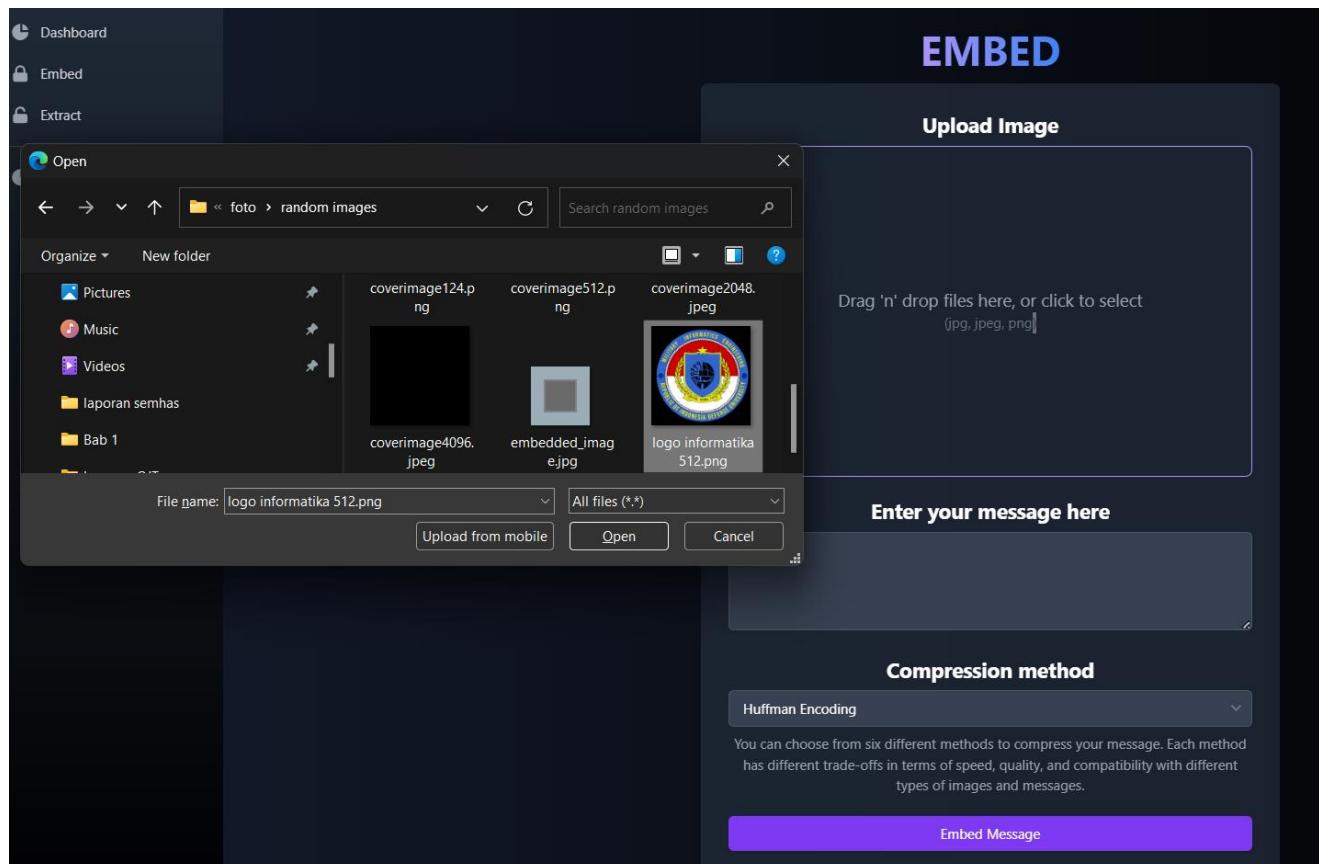
Sumber: diolah oleh peneliti

#### 4.2.1.2 *Huffman Coding*

- 1) Unggah *coverimage*

Tahapan pertama dalam melakukan proses steganografi adalah dengan mengunggah gambar yang akan peneliti pilih sebagai media

penampung pesan. Caranya bida dengan menahan dan menggeser gambar ke kotak *upload image* atau dapat juga dengan mengklik kotak *upload image* dan memilih gambar yang peneliti inginkan.

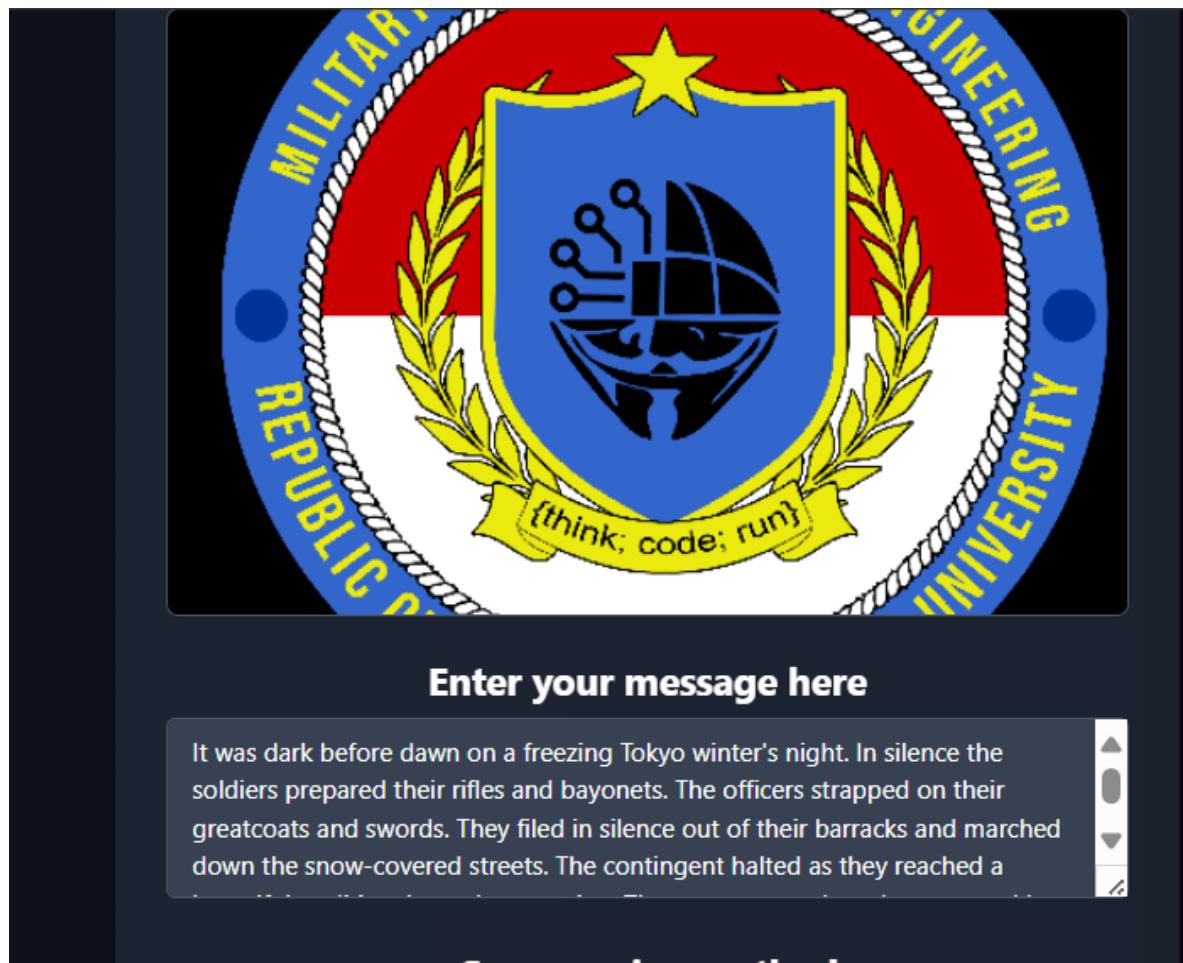


**Gambar 4.8 Unggah Coverimage Huffman**

Sumber: diolah oleh peneliti

2) Masukkan pesan rahasia

Ketik pesan rahasia yang ingin disisipkan ke dalam kotak di bawah tulisan “*Enter your message here*”.

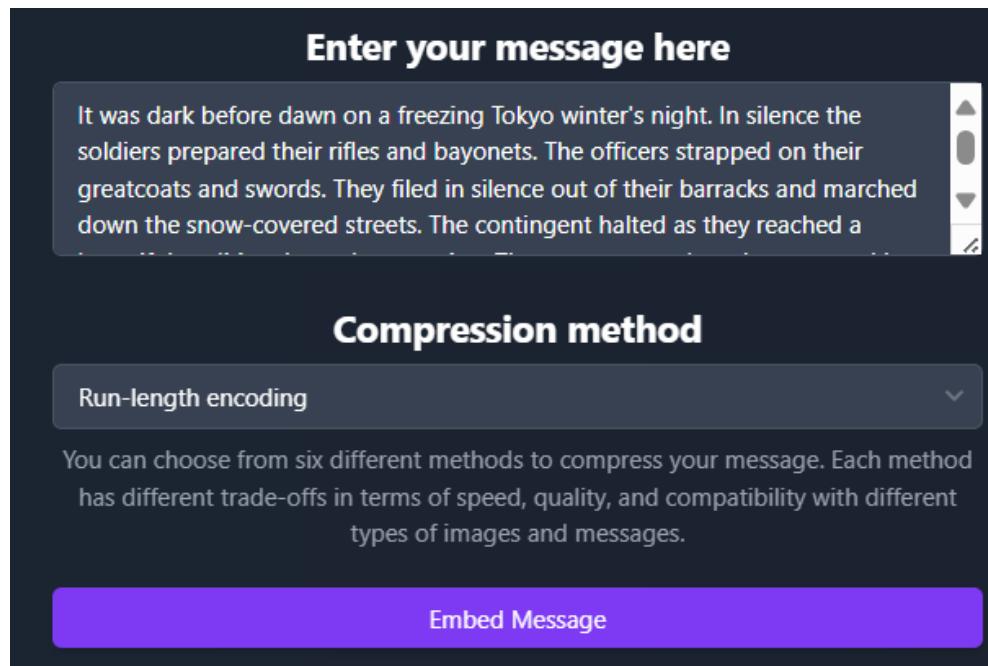


**Gambar 4.9 Masukan Pesan Rahasia Huffman**

Sumber: diolah oleh peneliti

3) Pilih metode kompresi

Pilih metode kompresi yang diinginkan. Pada bagian ini, karena peneliti akan mencoba menggunakan *Huffman Coding* maka peneliti pilih metode tersebut di kotak *compression method*. Setelah itu klik tombol *Embed Message*.



**Gambar 4.10 Pilih Metode Kompresi *Huffman***

Sumber: diolah oleh peneliti

4) Unduh *stegoimage* dan simpan *Huffman tree*

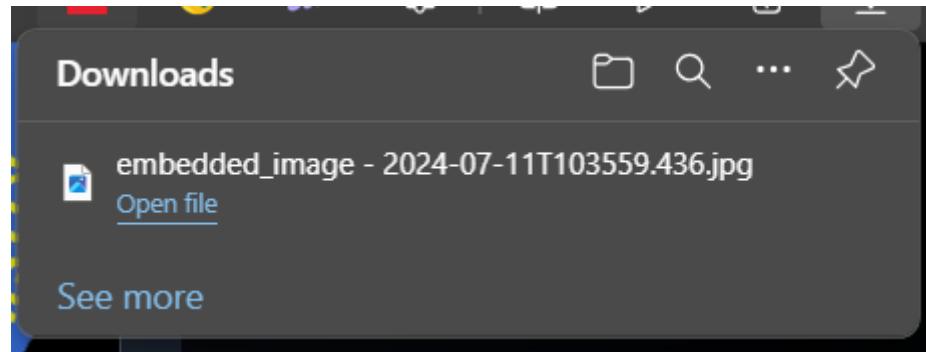
Untuk mengunduh gambar yang sudah disisipkan peneliti hanya perlu mengklik tombol *Download Image* yang berada di kanan ulasan gambar *stegoimage*. Selanjutnya simpan juga *Huffman tree* untuk proses ekstraksi pesan.



**Gambar 4.11 Unduh dan Simpan Pohon Huffman**

Sumber: diolah oleh peneliti

Gambar akan secara langsung terunduh ke dalam gawai yang digunakan.



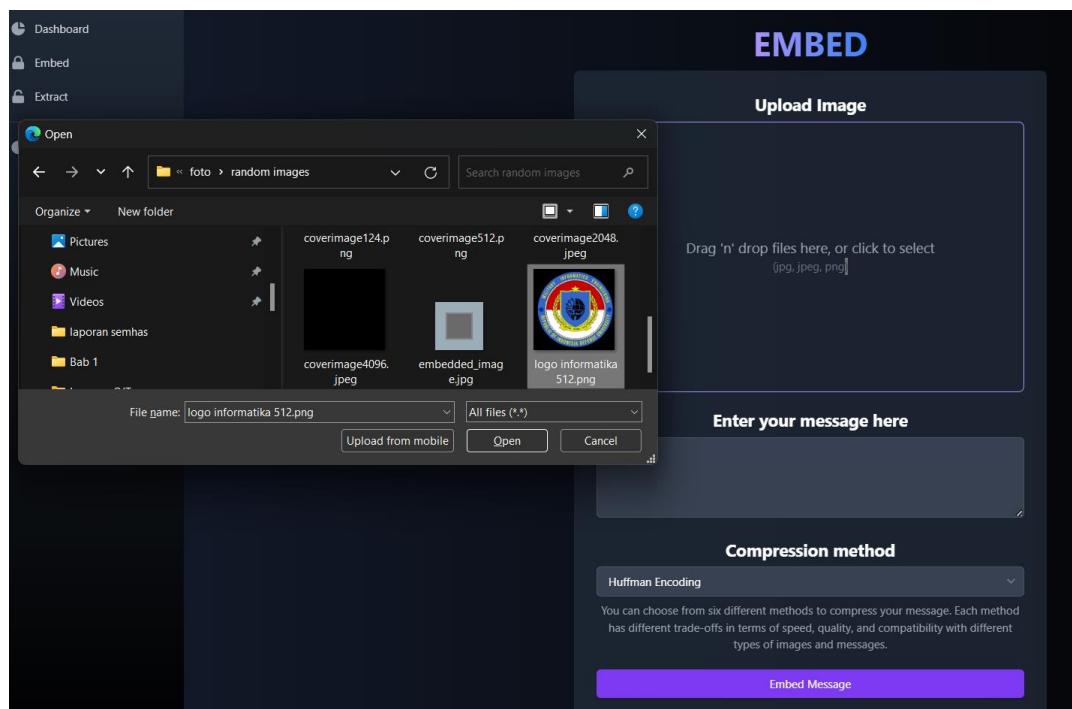
**Gambar 4.12 Unduh Gambar Huffman**

Sumber: diolah oleh peneliti

#### 4.2.1.3 Arithmetic Coding

##### 1) Unggah *coverimage*

Tahapan pertama dalam melakukan proses steganografi adalah dengan mengunggah gambar yang akan peneliti pilih sebagai media penampung pesan. Caranya bisa dengan menahan dan menggeser gambar ke kotak *upload image* atau dapat juga dengan mengklik kotak *upload image* dan memilih gambar yang peneliti inginkan.

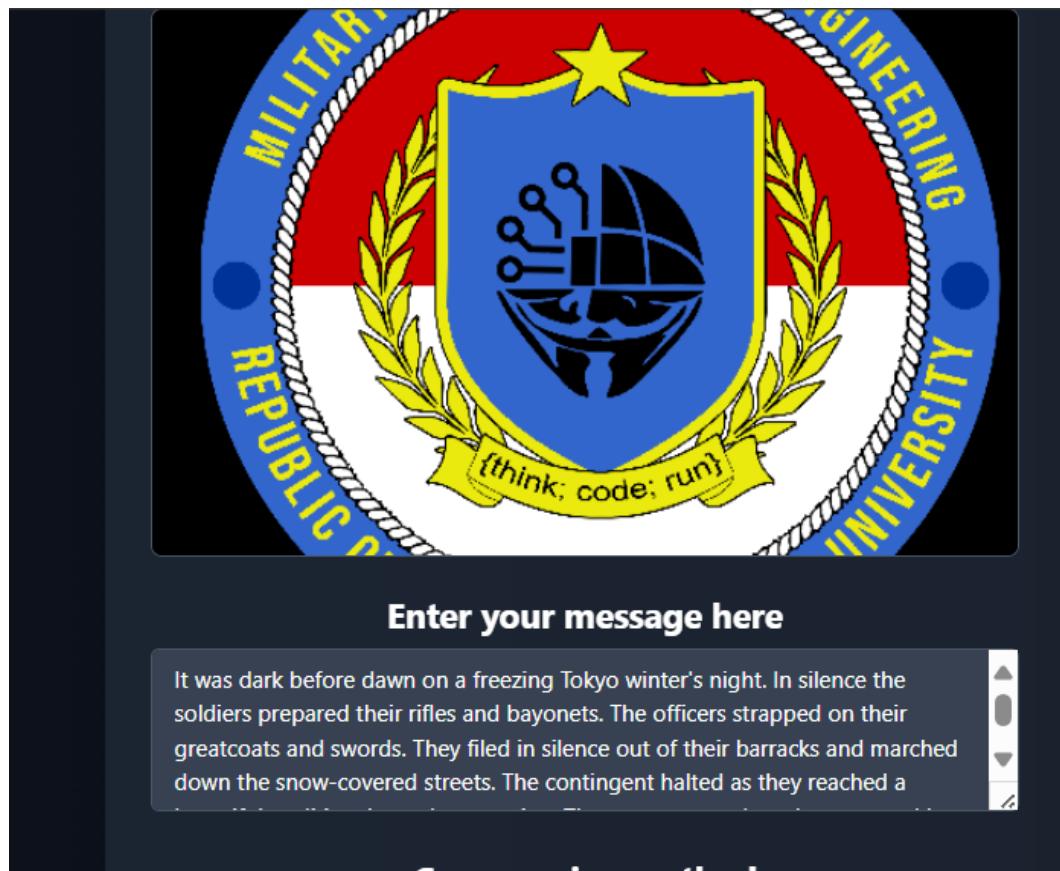


**Gambar 4.13 Unggah Coverimage Arithmetic**

Sumber: diolah oleh peneliti

2) Masukkan pesan rahasia

Ketik pesan rahasia yang ingin disisipkan ke dalam kotak di bawah tulisan “*Enter your message here*”.

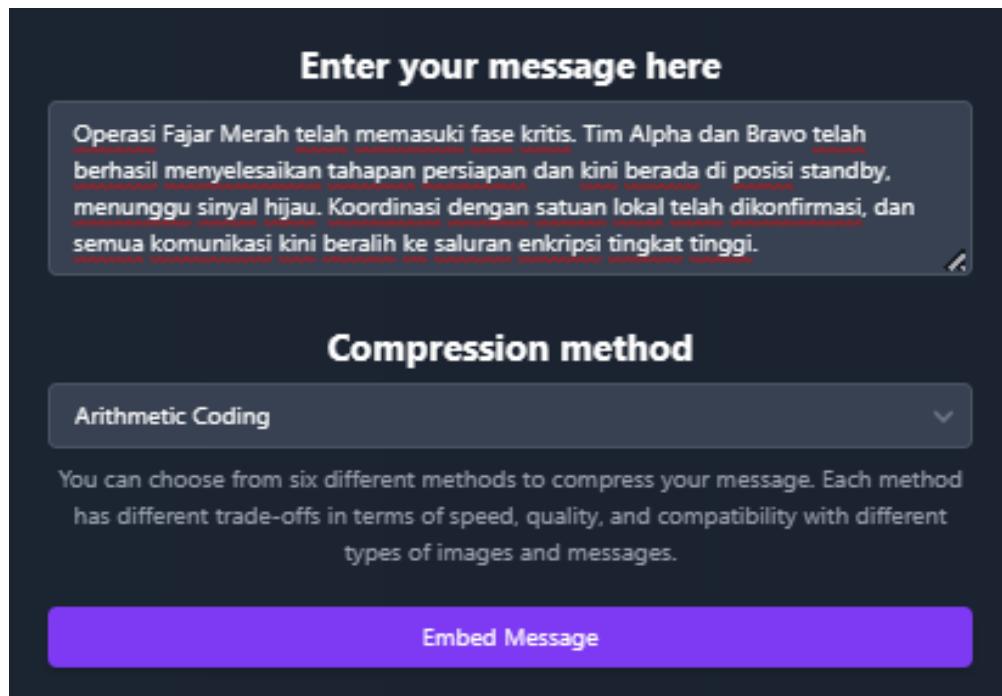


**Gambar 4.14 Masukan Pesan Rahasia *Arithmetic***

Sumber: diolah oleh peneliti

3) Pilih metode kompresi

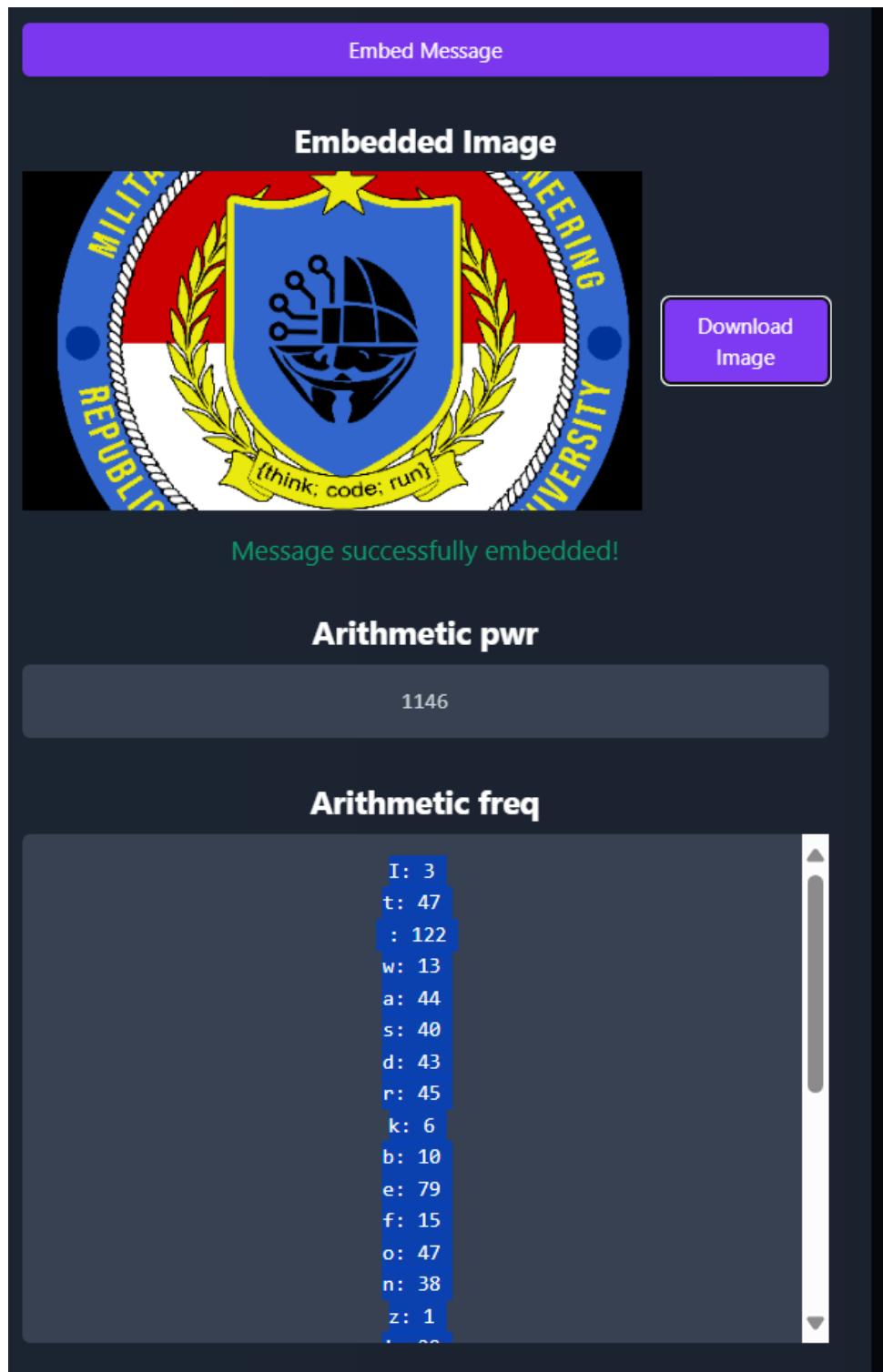
Pilih metode kompresi yang diinginkan. Pada bagian ini, karena peneliti akan mencoba menggunakan *Arithmetic Coding* maka peneliti pilih metode tersebut di kotak *compression method*. Setelah itu klik tombol *Embed Message*.



**Gambar 4.15 Pilih Metode Kompresi *Arithmetic***

Sumber: diolah oleh peneliti

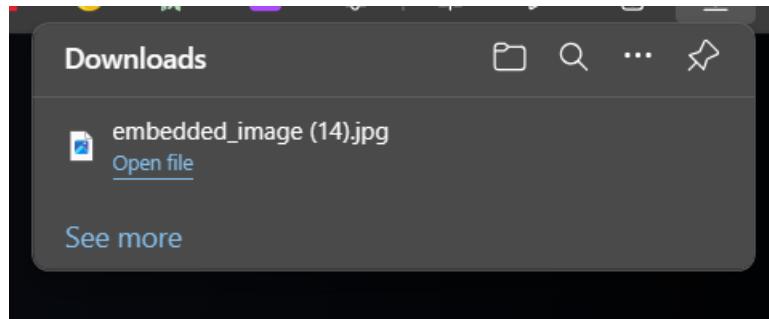
- 4) Unduh *stegoimage* serta simpan pwr dan freq  
Untuk mengunduh gambar yang sudah disisipkan peneliti hanya perlu mengklik tombol *Download Image* yang berada di kanan ulasan gambar *stegoimage*. Selanjutnya simpan juga *Arithmetic freq* dan *Arithmetic pwr* untuk proses ekstraksi pesan.



Gambar 4.16 Unduh Stegoimage, Pwr, Freq

Sumber: diolah oleh peneliti

Gambar akan secara langsung terunduh ke dalam gawai yang digunakan.



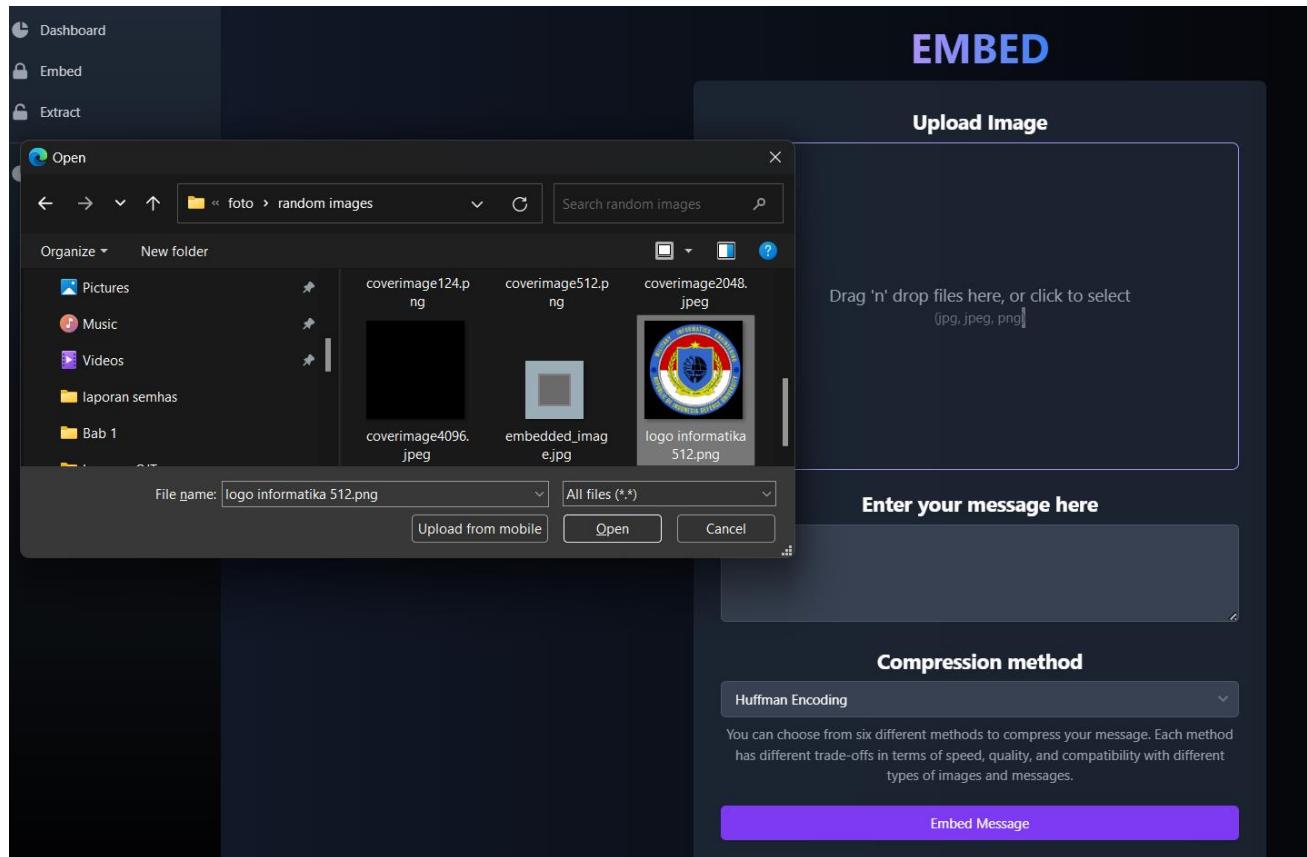
**Gambar 4.17 Unduh Stegoimage Arithmetic**

Sumber: diolah oleh peneliti

#### 4.2.1.4 Lempel-Ziv-Welch

##### 1) Unggah *coverimage*

Tahapan pertama dalam melakukan proses steganografi adalah dengan mengunggah gambar yang akan peneliti pilih sebagai media penampung pesan. Caranya bisa dengan menahan dan menggeser gambar ke kotak *upload image* atau dapat juga dengan mengklik kotak *upload image* dan memilih gambar yang peneliti inginkan.

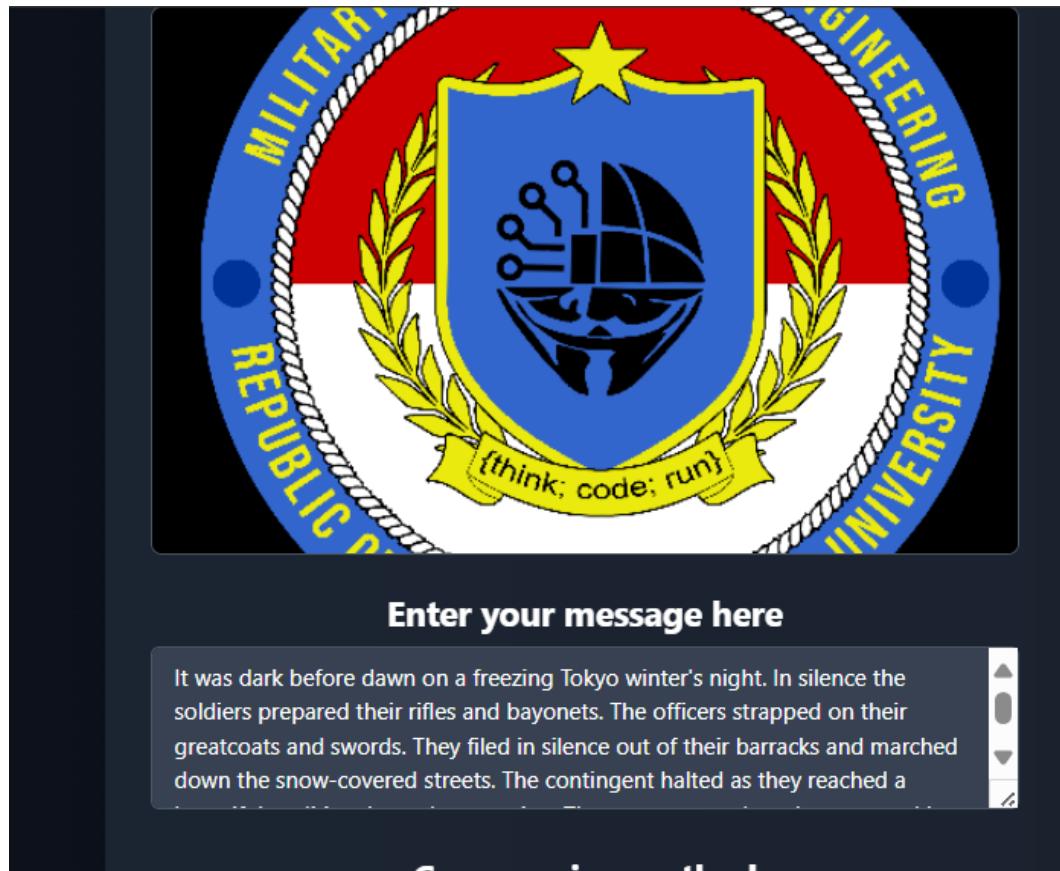


**Gambar 4.18 Unggah Coverimage LZW**

Sumber: diolah oleh peneliti

2) Masukkan pesan rahasia

Ketik pesan rahasia yang ingin disisipkan ke dalam kotak di bawah tulisan "*Enter your message here*".

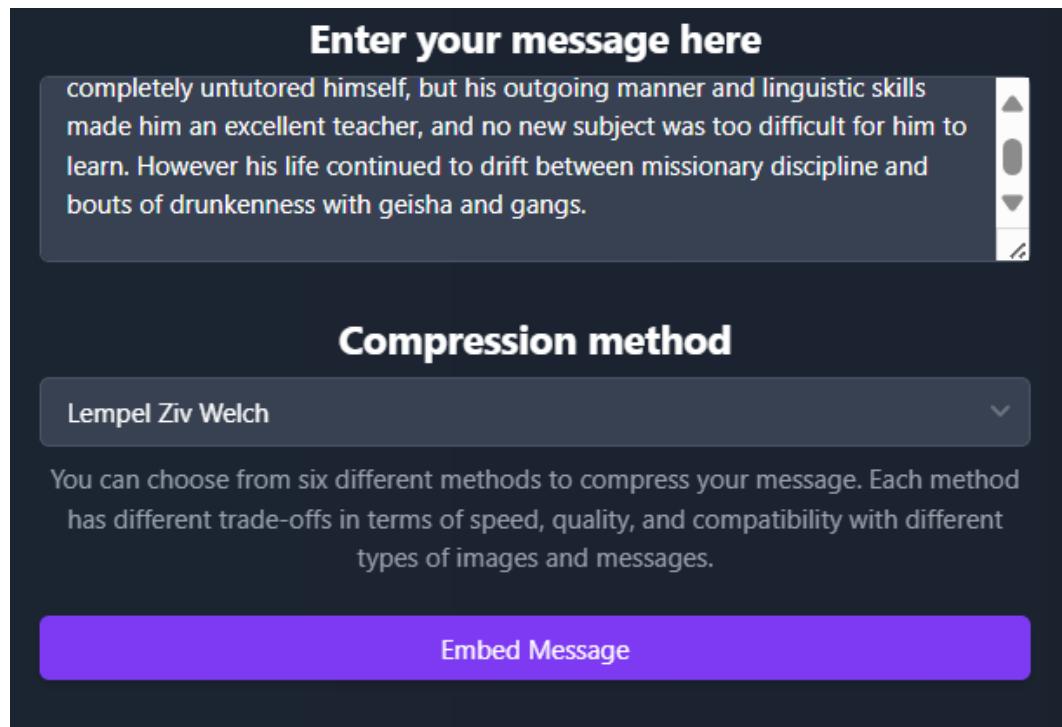


**Gambar 4.19 Masukkan Pesan Rahasia LZW**

Sumber: diolah oleh peneliti

3) Pilih metode kompresi

Pilih metode kompresi yang diinginkan. Pada bagian ini, karena peneliti akan mencoba menggunakan *Lempel-Ziv-Welch* maka peneliti pilih metode tersebut di kotak *compression method*. Setelah itu klik tombol *Embed Message*.

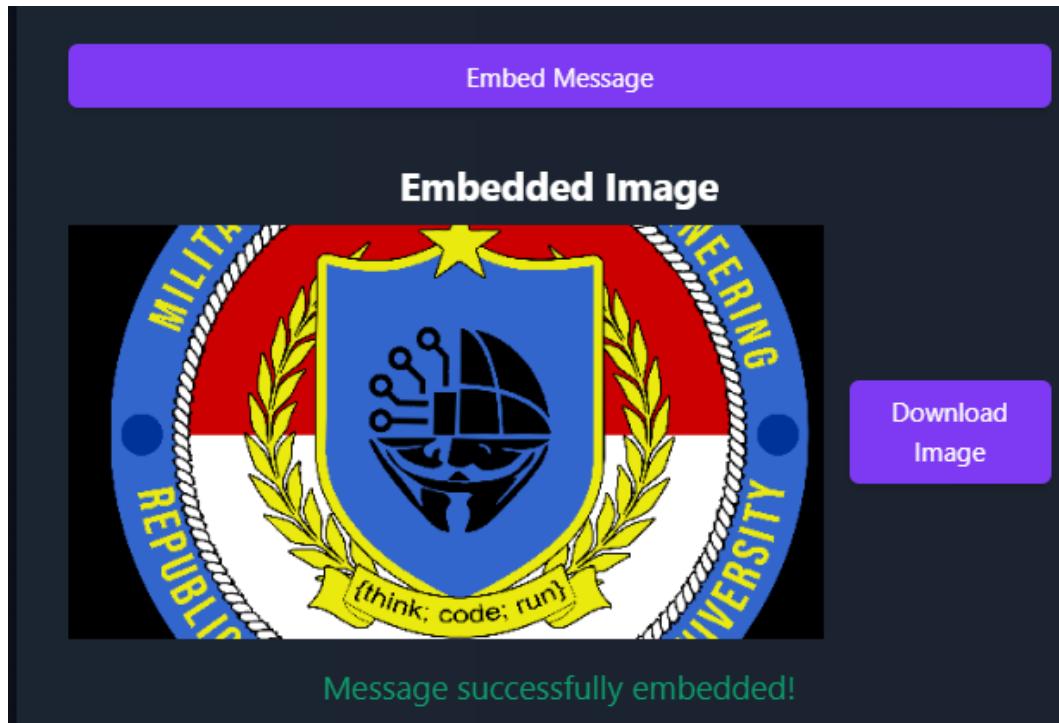


**Gambar 4.20 Pilih Metode Kompresi LZW**

Sumber: diolah oleh peneliti

4) Unduh *stegoimage*

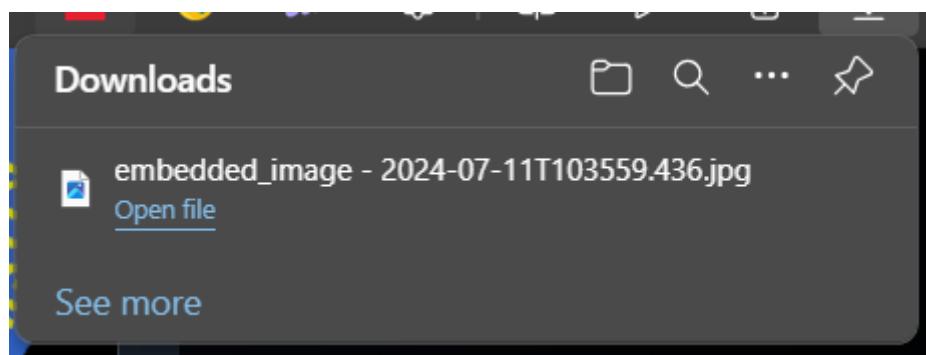
Untuk mengunduh gambar yang sudah disisipkan peneliti hanya perlu mengklik tombol *Download Image* yang berada di kanan ulasan gambar *stegoimage*.



**Gambar 4.21 Unduh Stegoimage**

Sumber: diolah oleh peneliti

Gambar akan secara langsung terunduh ke dalam gawai yang digunakan.



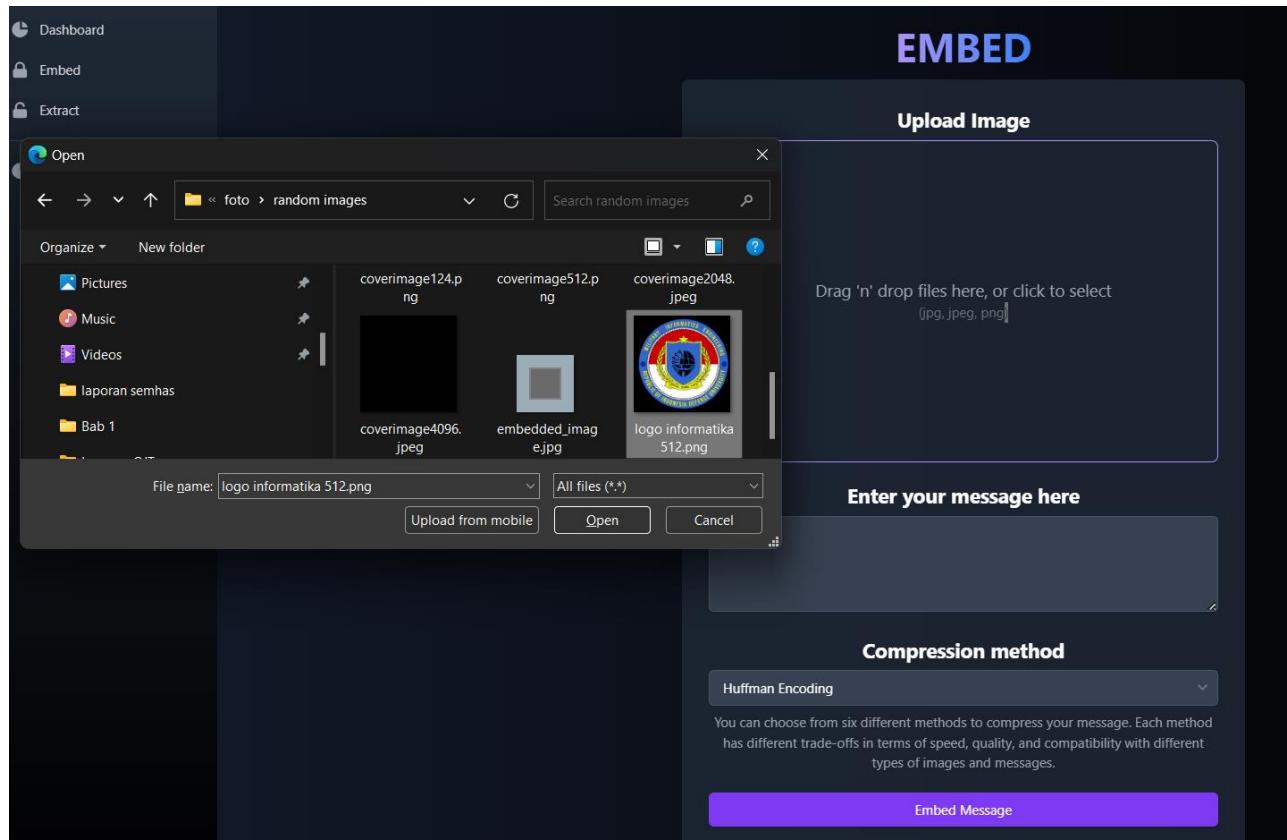
**Gambar 4.22 Stegoimage Terunduh LZW**

Sumber: diolah oleh peneliti

#### **4.2.1.5 J-bit Encoding**

- 1) Unggah *coverimage*

Tahapan pertama dalam melakukan proses steganografi adalah dengan mengunggah gambar yang akan peneliti pilih sebagai media penampung pesan. Caranya bisa dengan menahan dan menggeser gambar ke kotak *upload image* atau dapat juga dengan mengklik kotak *upload image* dan memilih gambar yang peneliti inginkan.

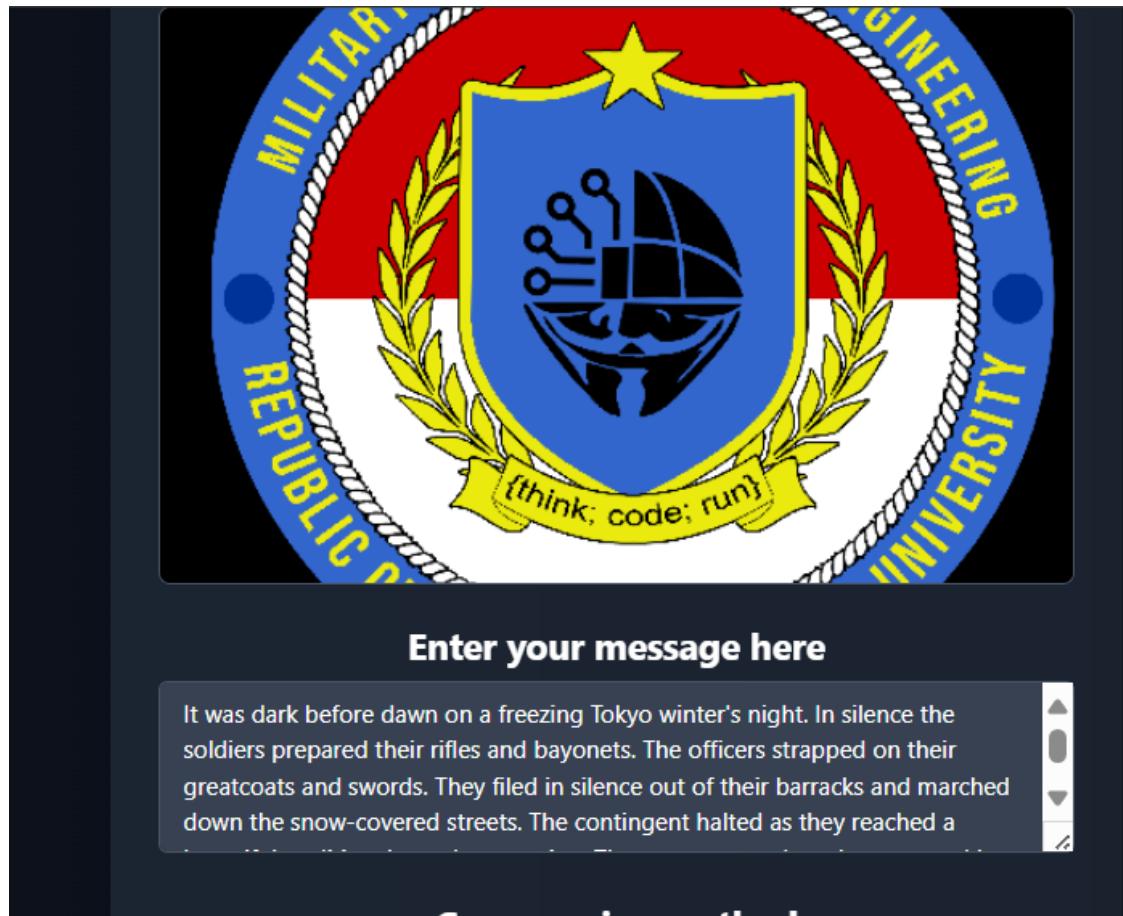


**Gambar 4.23 Unggah Coverimage JBE**

Sumber: diolah oleh peneliti

2) Masukkan pesan rahasia

Ketik pesan rahasia yang ingin disisipkan ke dalam kotak di bawah tulisan “*Enter your message here*”.

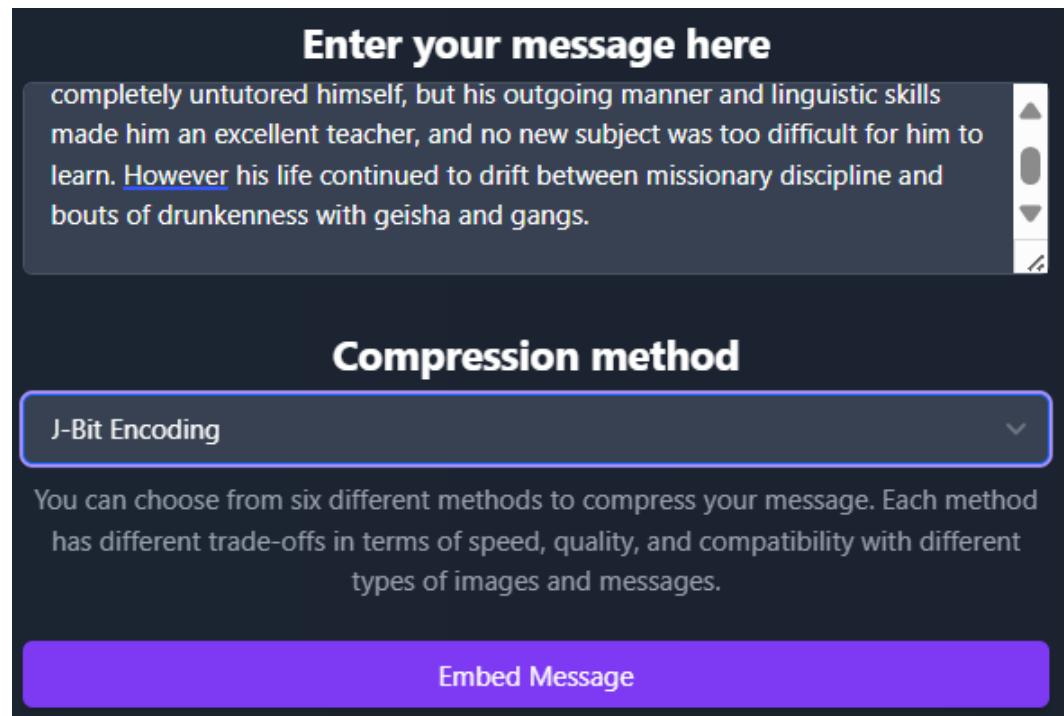


Gambar 4.24 Masukkan Pesan Rahasia JBE

Sumber: diolah oleh peneliti

3) Pilih metode kompresi

Pilih metode kompresi yang diinginkan. Pada bagian ini, karena peneliti akan mencoba menggunakan *J-bit Encoding* maka peneliti pilih metode tersebut di kotak *compression method*. Setelah itu klik tombol *Embed Message*.

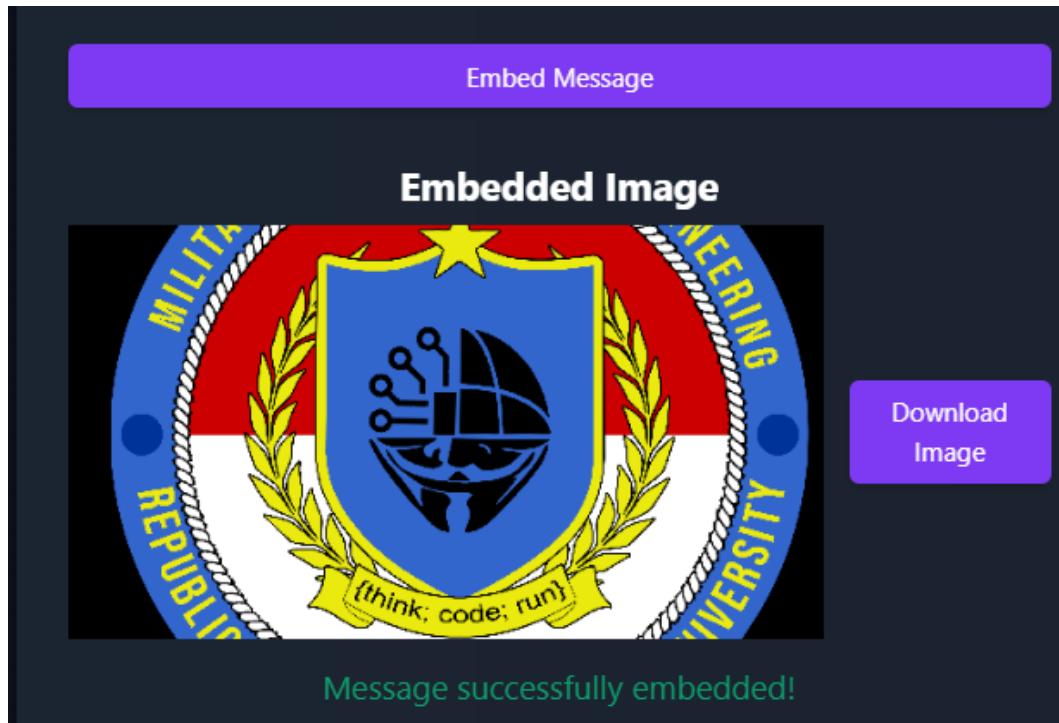


**Gambar 4.25 Pilih Metode Kompresi JBE**

Sumber: diolah oleh peneliti

4) Unduh *stegoimage*

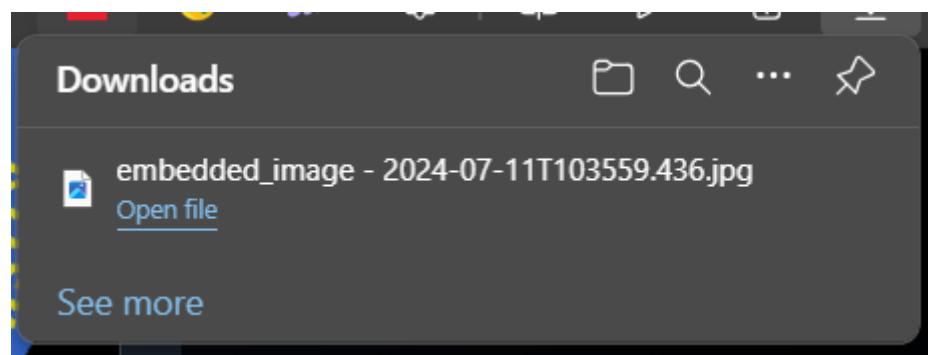
Untuk mengunduh gambar yang sudah disisipkan peneliti hanya perlu mengklik tombol *Download Image* yang berada di kanan ulasan gambar *stegoimage*.



**Gambar 4.26 Unduh Stegoimage JBE**

Sumber: diolah oleh peneliti

Gambar akan secara langsung terunduh ke dalam gawai yang digunakan.



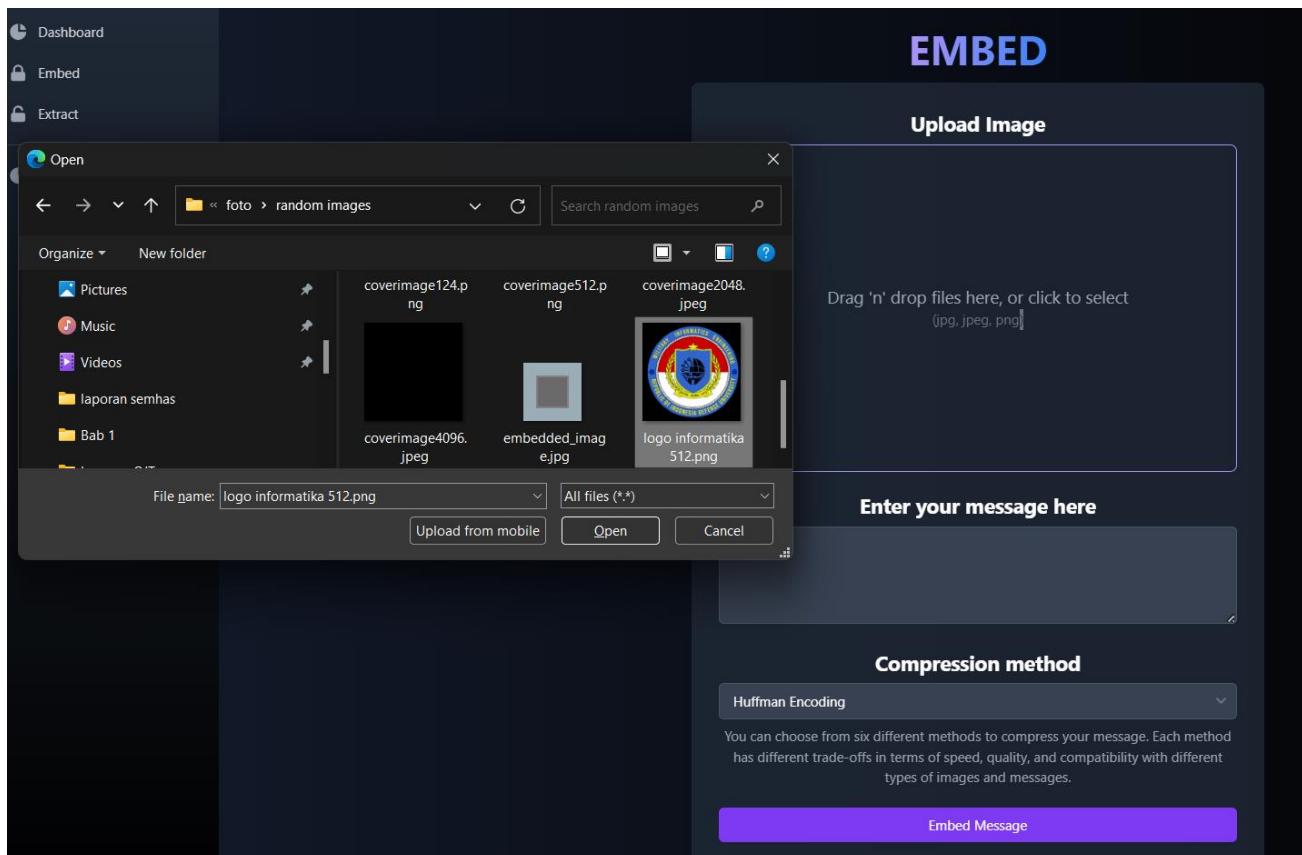
**Gambar 4.27 Stegoimage Terunduh JBE**

Sumber: diolah oleh peneliti

#### **4.2.1.6 Deflate Compression**

- 1) Unggah coverimage

Tahapan pertama dalam melakukan proses steganografi adalah dengan mengunggah gambar yang akan peneliti pilih sebagai media penampung pesan. Caranya bisa dengan menahan dan menggeser gambar ke kotak *upload image* atau dapat juga dengan mengklik kotak *upload image* dan memilih gambar yang peneliti inginkan.

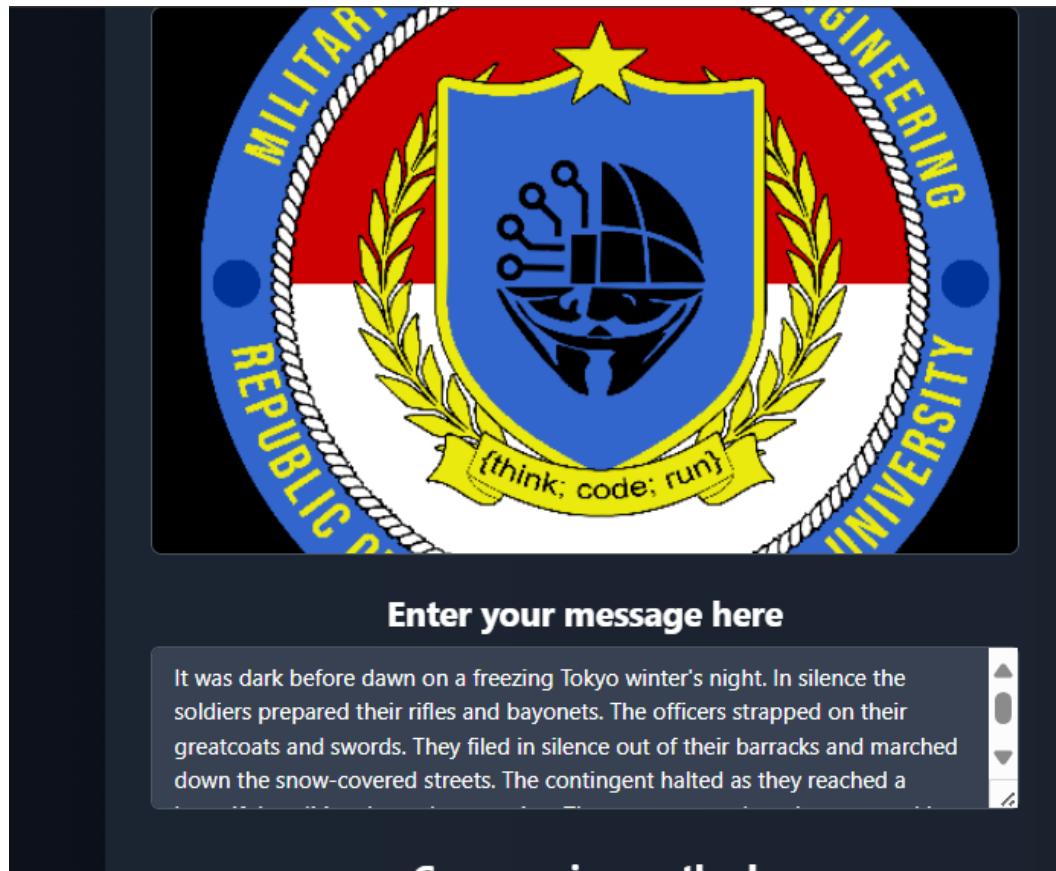


**Gambar 4.28 Unggah Coverimage Deflate**

Sumber: diolah oleh peneliti

2) Masukkan pesan rahasia

Ketik pesan rahasia yang ingin disisipkan ke dalam kotak di bawah tulisan “*Enter your message here*”.

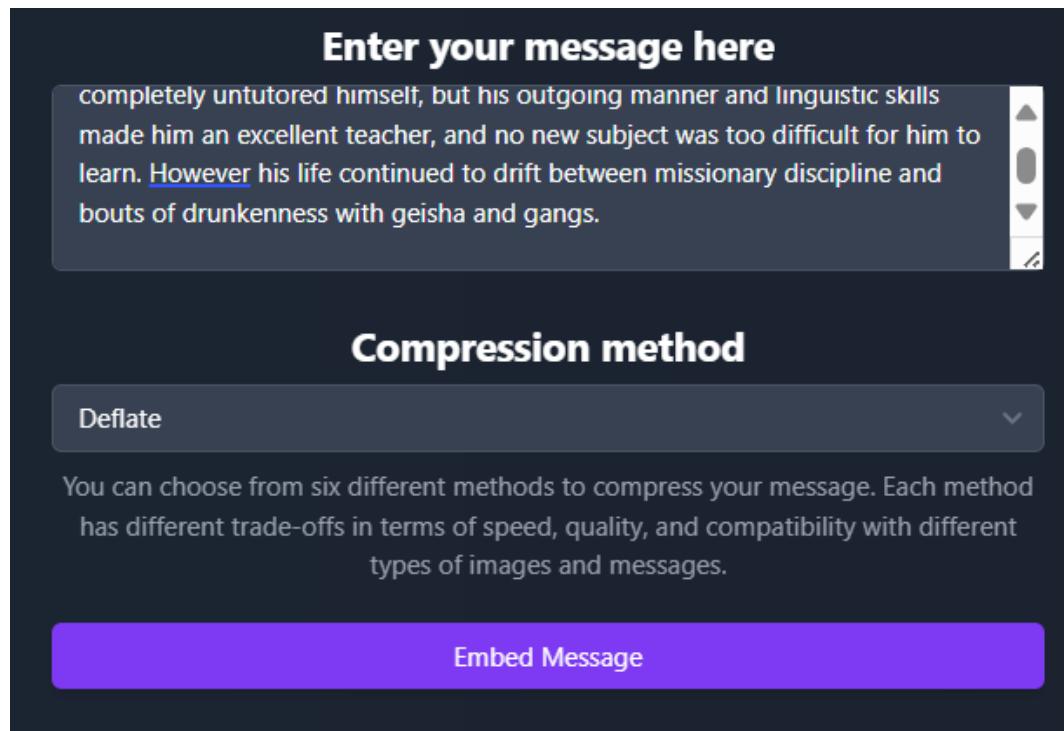


**Gambar 4.29 Masukkan Pesan Rahasia *Deflate***

Sumber: diolah oleh peneliti

3) Pilih metode kompresi

Pilih metode kompresi yang diinginkan. Pada bagian ini, karena peneliti akan mencoba menggunakan *Deflate Compression* maka peneliti pilih metode tersebut di kotak *compression method*. Setelah itu klik tombol *Embed Message*.

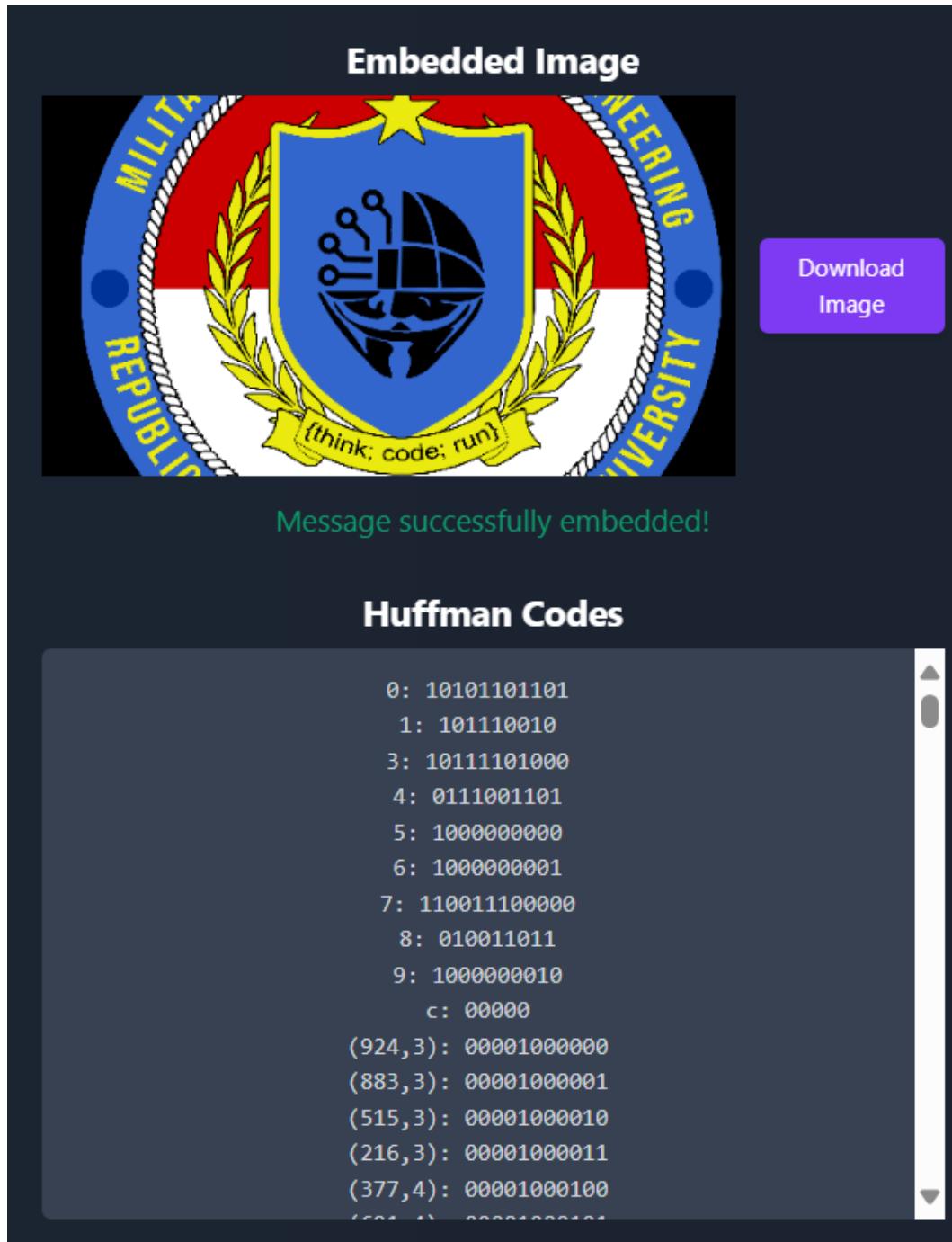


**Gambar 4.30 Pilih Metode Kompresi *Deflate***

Sumber: diolah oleh peneliti

4) Unduh *stegoimage* dan simpan *Huffman tree*

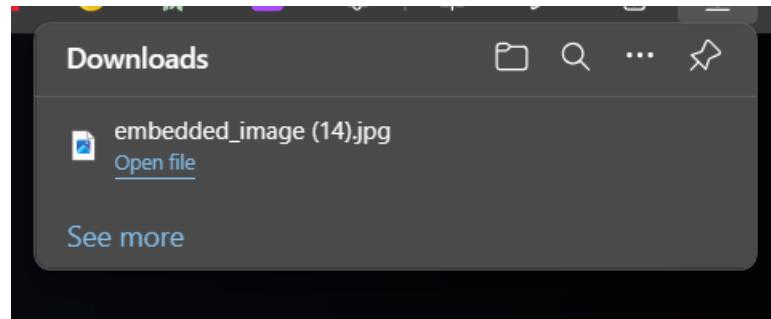
Untuk mengunduh gambar yang sudah disisipkan peneliti hanya perlu mengklik tombol *Download Image* yang berada di kanan ulasan gambar *stegoimage*. Selanjutnya simpan juga *Huffman tree* untuk proses ekstraksi pesan.



**Gambar 4.31 Unduh Stegoimage dan Pohon Huffman Deflate**

Sumber: diolah oleh peneliti

Gambar akan secara langsung terunduh ke dalam gawai yang digunakan.

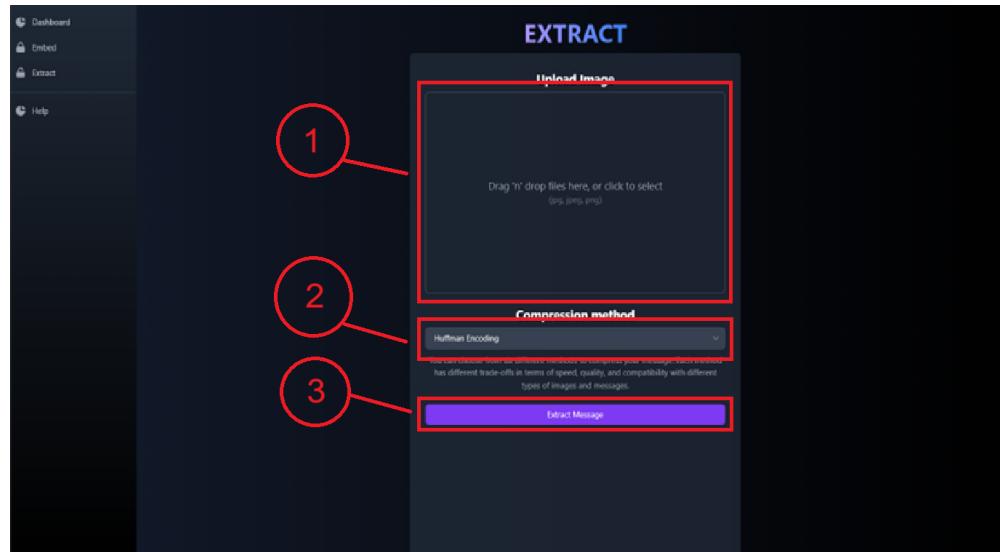


**Gambar 4.32 Gambar Terunduh Deflate**

Sumber: diolah oleh peneliti

#### 4.2.2 Ekstraksi Gambar

Gambar 4.3 adalah gambaran laman *extract* dalam aplikasi steganografi berbasis web. Terdapat dua hal yaitu contoh gambar yang akan diunggah dan juga tombol pilihan algoritma. Berdasarkan dari algoritma yang digunakan maka akan muncul kotak tambahan yang akan ditampilkan.



**Gambar 4.33 Gambar Laman Extract**

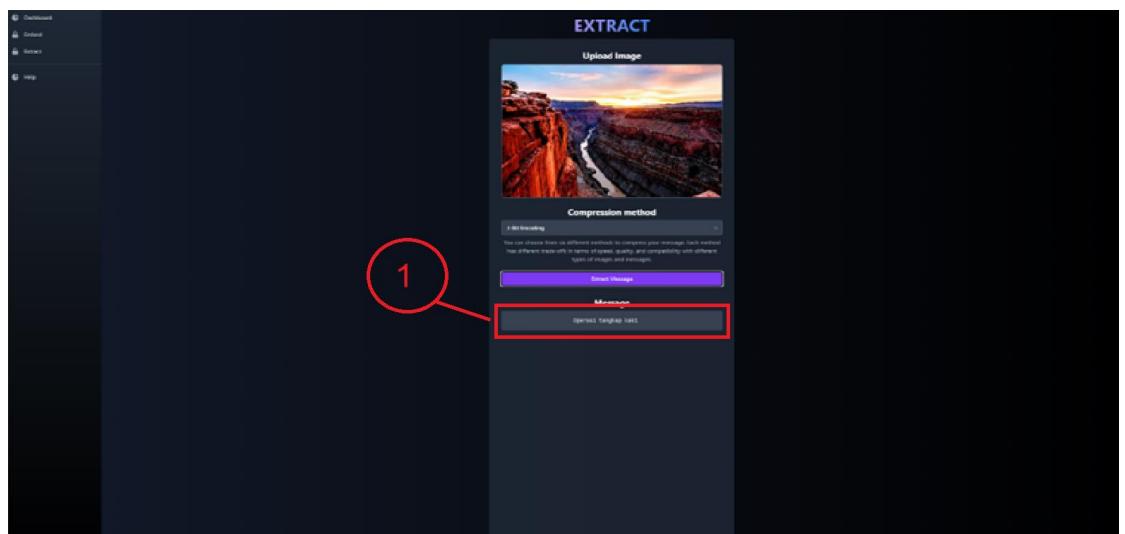
Sumber: diolah oleh peneliti

Pada gambar 4.3 terdapat beberapa fungsi dari komponen yang ada, berikut penjelasan dari tiap-tiap komponen:

1. Memilih gambar yang akan digunakan untuk mengekstrak pesan

2. Memilih metode kompresi yang digunakan untuk mengekstrak pesan dari gambar
3. *Submit* untuk memproses pengekstrakan gambar

Gambar 4.4 merupakan gambaran laman *extract* setelah klik dalam aplikasi steganografi berbasis web setelah tombol mulai diklik. Dapat dilihat di sini akan muncul pesan rahasia yang terdapat di dalam gambar *stegoimage* yang diunggah.



**Gambar 4.34 Gambar Laman Extract Setelah Klik**

Sumber: diolah oleh peneliti

Pada gambar 4.4 terdapat beberapa fungsi dari komponen yang ada, berikut penjelasan dari tiap-tiap komponen:

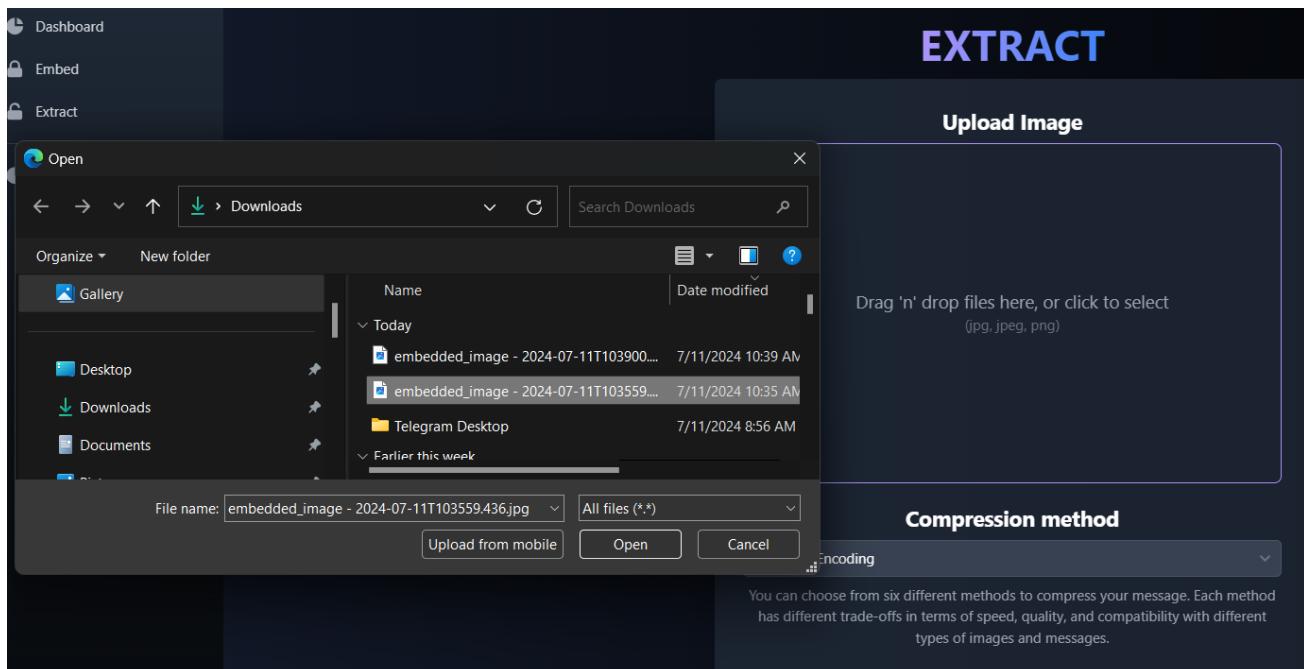
1. Pesan hasil ekstraksi muncul

Berikut merupakan tahapan pengejaan proses steganografi dari mulai mengunggah gambar hingga mengunduhnya ke dalam sistem. Proses akan dibagi menjadi enam metode. Setiap metode yang dikerjakan akan menggunakan gambar dan teks sederhana agar memperjelas detail dari gambar yang dimasukan.

#### **4.2.2.1 Run-Length Encoding**

- 1) Unggah *stegoimage*

Selanjutnya untuk membaca kembali pesan yang telah disisipkan ke dalam *stegoimage* peneliti perlu mengunggah gambar yang ingin diekstrak pada laman *extract*.

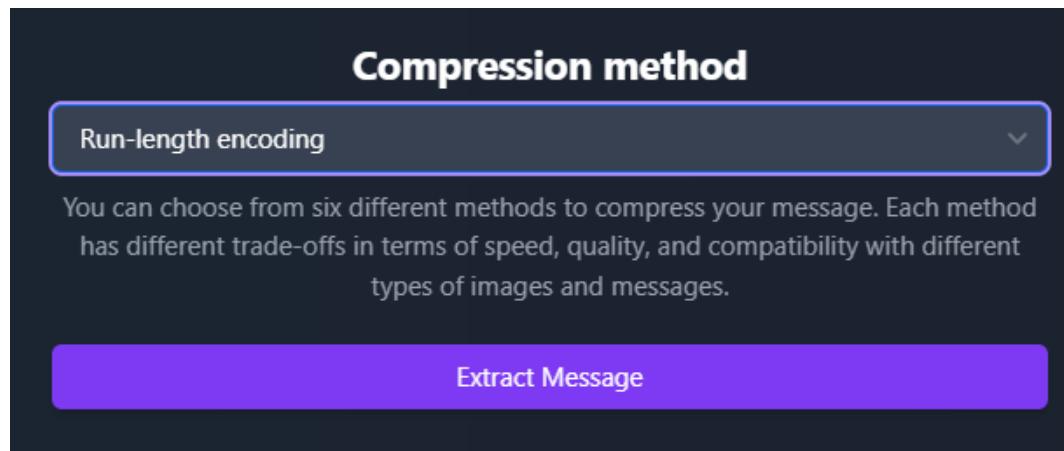


**Gambar 4.35 Gambar Unggah Stegoimgae RLE**

Sumber: diolah oleh peneliti

## 2) Pilih metode kompresi

Pilih metode kompresi yang diinginkan. Pada bagian ini, karena peneliti akan mencoba menggunakan *Run-Length Encoding* maka peneliti pilih metode tersebut di kotak *compression method*. Setelah itu klik tombol *Extract Message*.

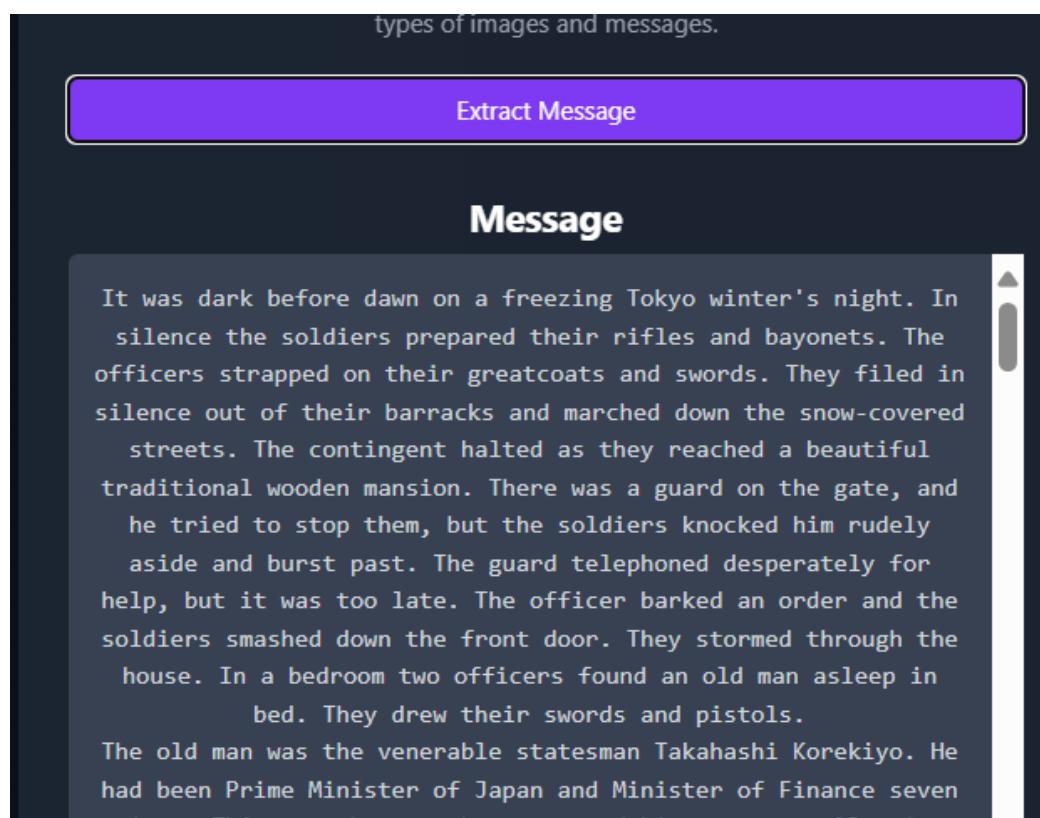


**Gambar 4.36 Gambar Pilih Metode RLE**

Sumber: diolah oleh peneliti

3) Membaca hasil pesan

Setelah proses ekstraksi selesai, tulisan pesan tersembunyi akan muncul pada kotak di bawah tombol *Extract Message*.



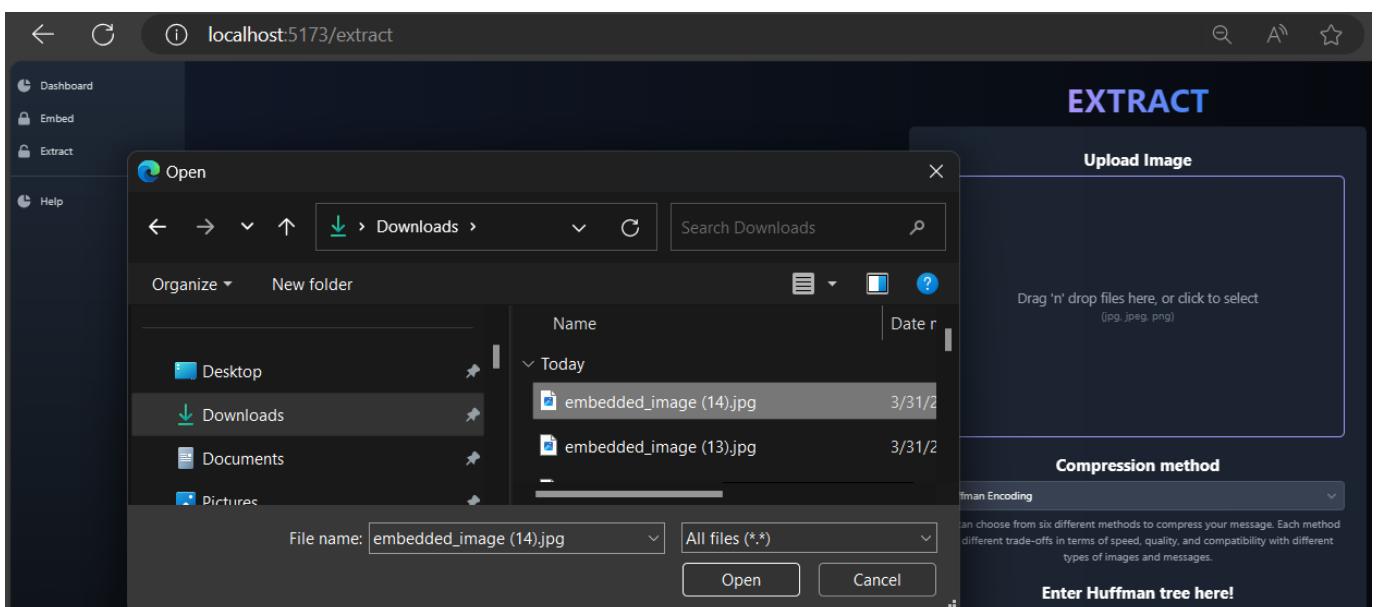
**Gambar 4.37 Baca Pesan RLE**

Sumber: diolah oleh peneliti

#### 4.2.2.2 Huffman Coding

1) Unggah *stegoimage*

Selanjutnya untuk membaca kembali pesan yang telah disisipkan ke dalam *stegoimage* peneliti perlu mengunggah gambar yang ingin diekstrak pada laman *extract*.

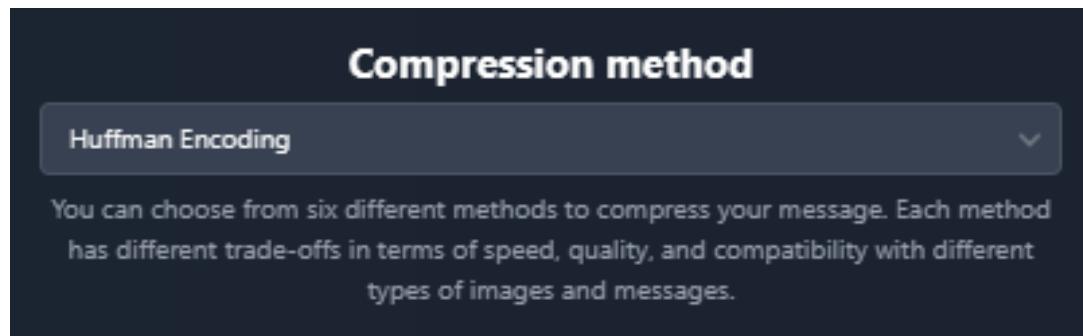


**Gambar 4.38 Unggah Stegoimage Huffman**

Sumber: diolah oleh peneliti

2) Pilih metode kompresi

Pilih metode kompresi yang diinginkan. Pada bagian ini, karena peneliti akan mencoba menggunakan *Huffman Coding* maka peneliti pilih metode tersebut di kotak *compression method*. Setelah itu klik tombol *Extract Message*.

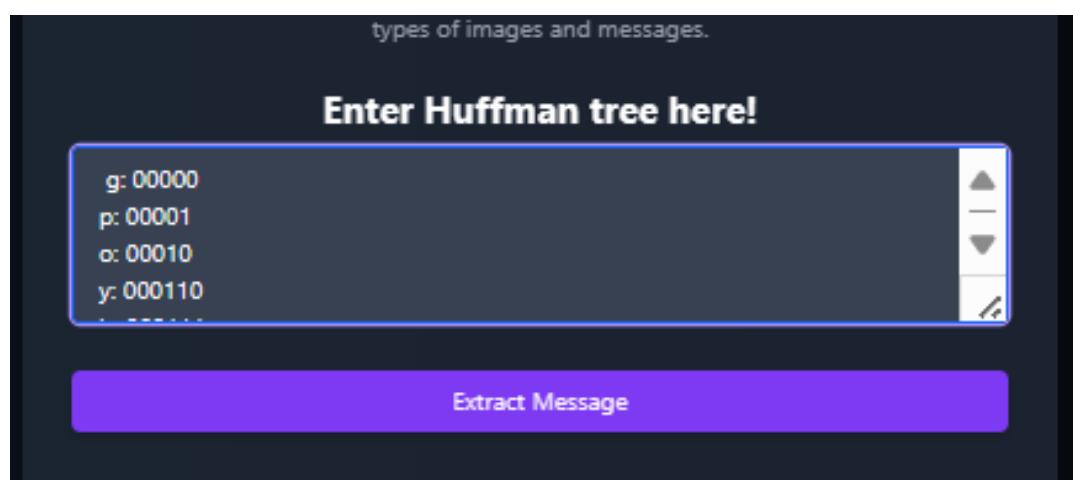


**Gambar 4.39 Pilih Metode Kompresi Huffman**

Sumber: diolah oleh peneliti

3) Tulis *Huffman tree*

Untuk memulai proses ekstraksi, metode kompresi *Huffman Coding* menggunakan kode-kode tertentu untuk menentukan sejumlah teks bit yang akan dikonversikan menjadi karakter. Ketik kode yang sudah disalin sebelumnya untuk kemudian proses ekstraksi dapat dimulai dengan mengklik tombol *Extract Message*.

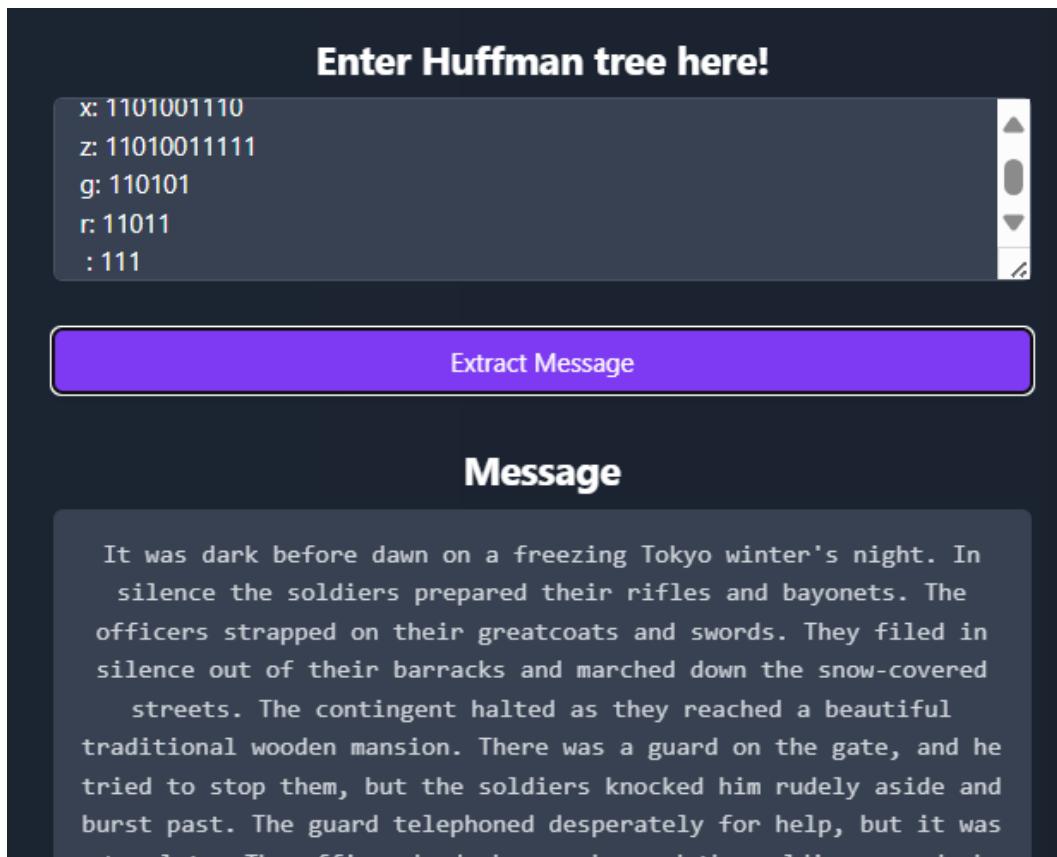


**Gambar 4.40 Tulis Pohon Huffman**

Sumber: diolah oleh peneliti

4) Membaca hasil pesan

Setelah proses ekstraksi selesai, tulisan pesan tersembunyi akan muncul pada kotak di bawah tombol *Extract Message*.



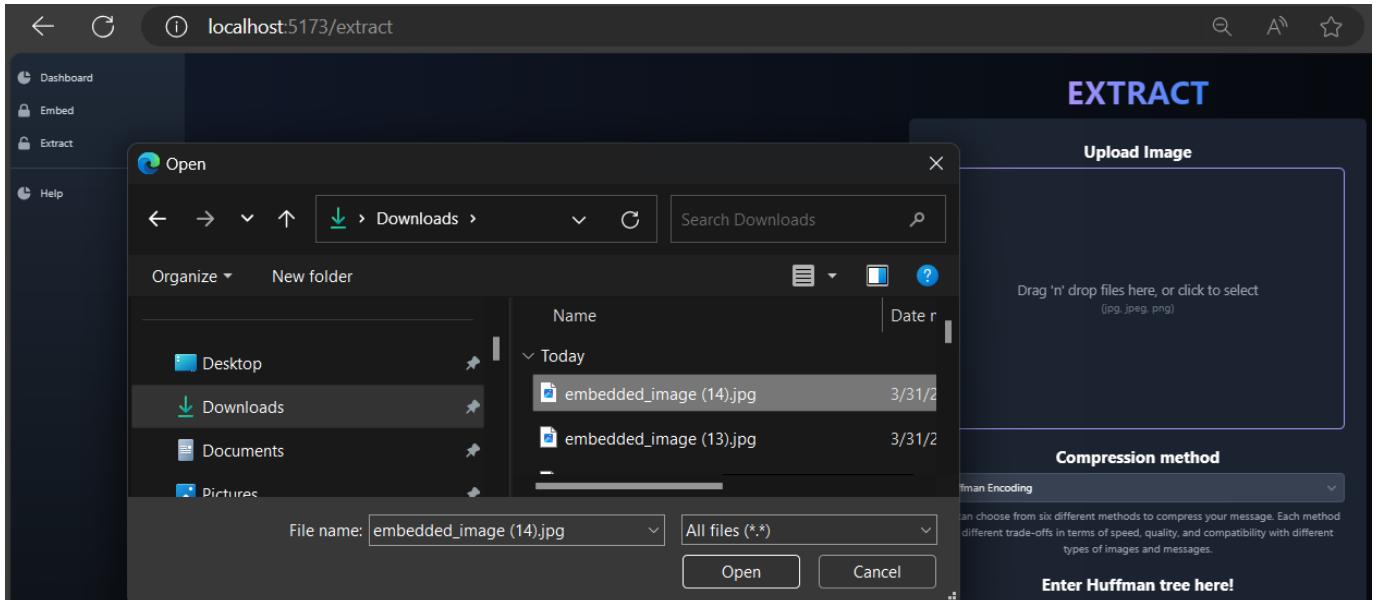
**Gambar 4.41 Membaca Hasil Pesan *Huffman***

Sumber: diolah oleh peneliti

#### 4.2.2.3 *Arithmetic Coding*

- 1) Unggah *stegoimage*

Selanjutnya untuk membaca kembali pesan yang telah disisipkan ke dalam *stegoimage* peneliti perlu mengunggah gambar yang ingin diekstrak pada laman *extract*.

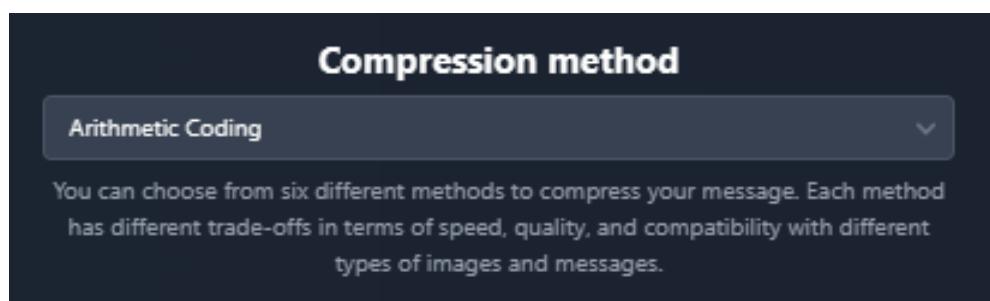


**Gambar 4.42 Unggah Stegoimage *Arithmetic***

Sumber: diolah oleh peneliti

2) Pilih metode kompresi

Pilih metode kompresi yang diinginkan. Pada bagian ini, karena peneliti akan mencoba menggunakan *Arithmetic Coding* maka peneliti pilih metode tersebut di kotak *compression method*. Setelah itu klik tombol *Extract Message*.



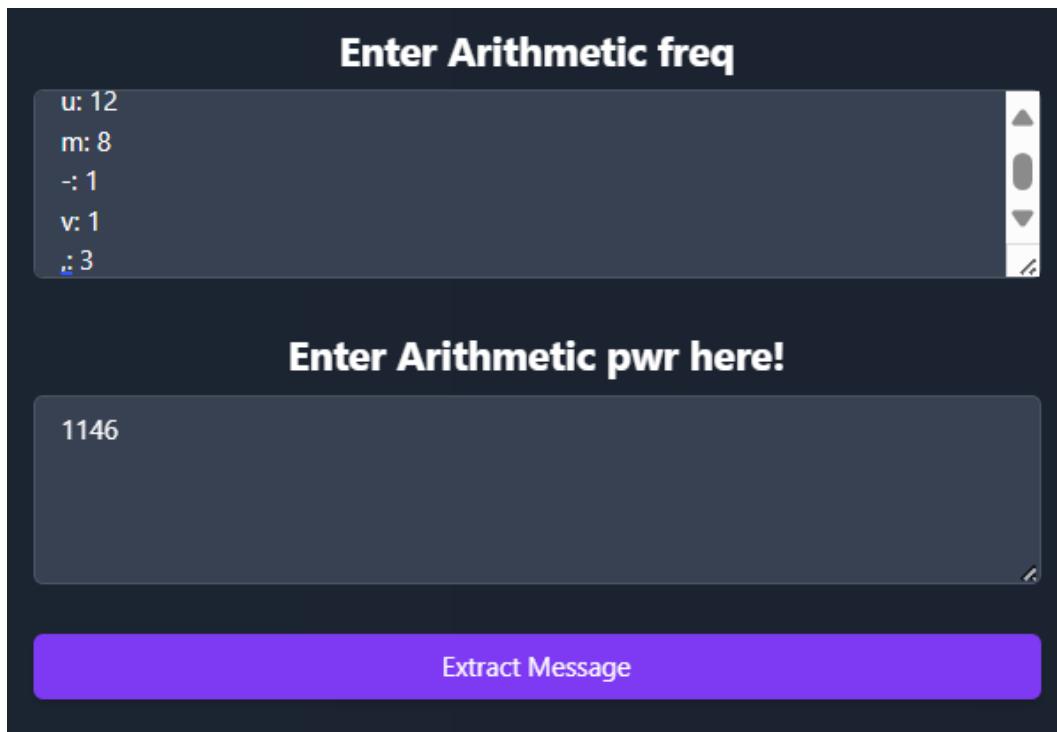
**Gambar 4.43 Pilih Metode Dekompresi *Arithmetic***

Sumber: diolah oleh peneliti

3) Tulis *Arithmetic freq* dan *Arithmetic pwr*

Untuk memulai proses ekstraksi, metode kompresi *Arithmetic Coding* menggunakan suatu kamus tertentu untuk menghitung jumlah

karakter berdasarkan frekuensi kemunculan suatu karakter serta menggunakan pangkat dari negatif sepuluh untuk mengetahui angka desimal yang dikonversikan menjadi teks semula. Ketik kode yang sudah disalin sebelumnya untuk kemudian proses ekstraksi dapat dimulai dengan mengklik tombol *Extract Message*.

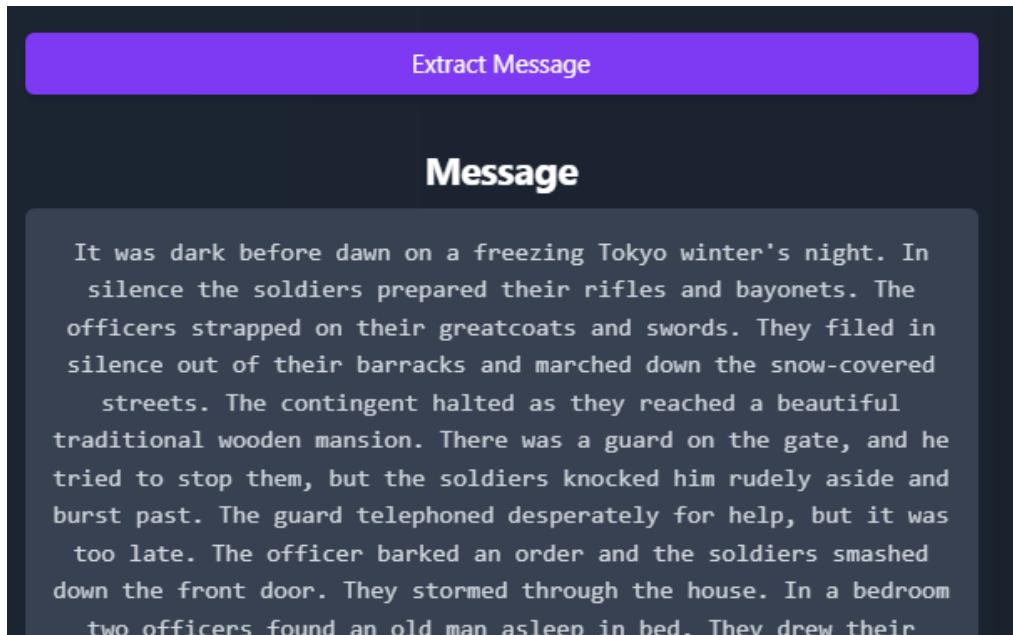


**Gambar 4.44 Tuliskan *Arithmetic Freq* dan *Pwr***

Sumber: diolah oleh peneliti

4) Membaca hasil pesan

Setelah proses ekstraksi selesai, tulisan pesan tersembunyi akan muncul pada kotak di bawah tombol *Extract Message*.



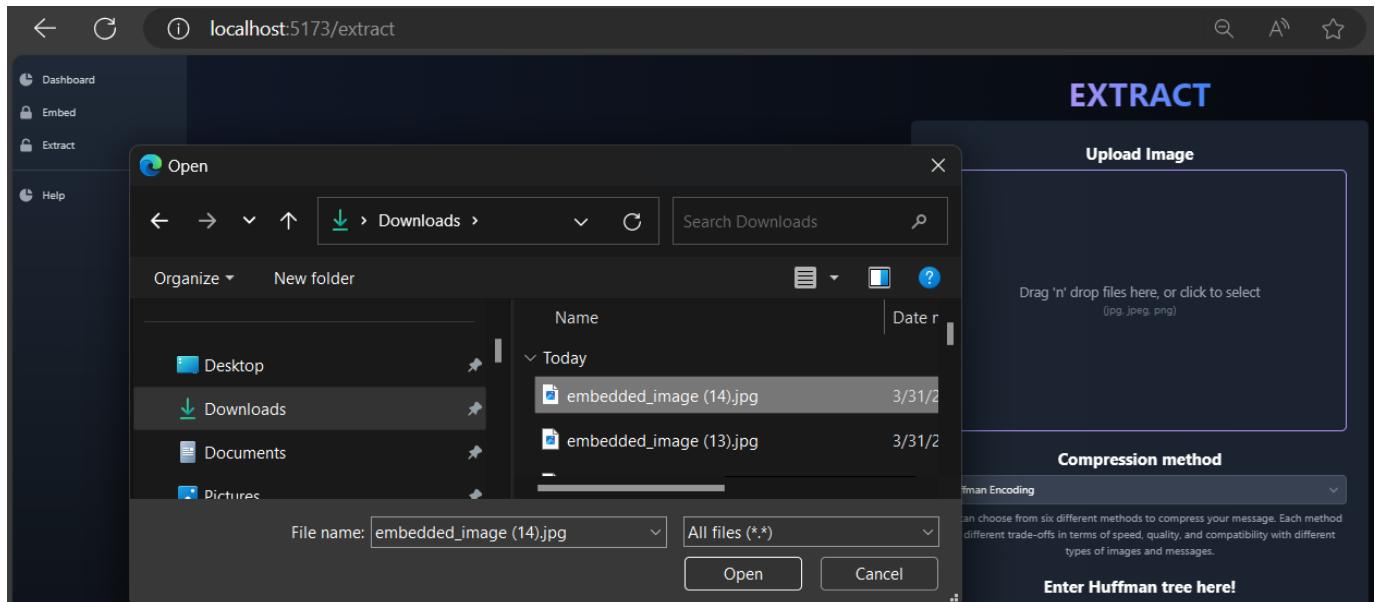
Gambar 4.45 Membaca Hasil Pesan *Arithmetic*

Sumber: diolah oleh peneliti

#### 4.2.2.4 Lempel-Ziv-Welch

- 1) Unggah *stegoimage*

Selanjutnya untuk membaca kembali pesan yang telah disisipkan ke dalam *stegoimage* peneliti perlu mengunggah gambar yang ingin diekstrak pada laman *extract*.

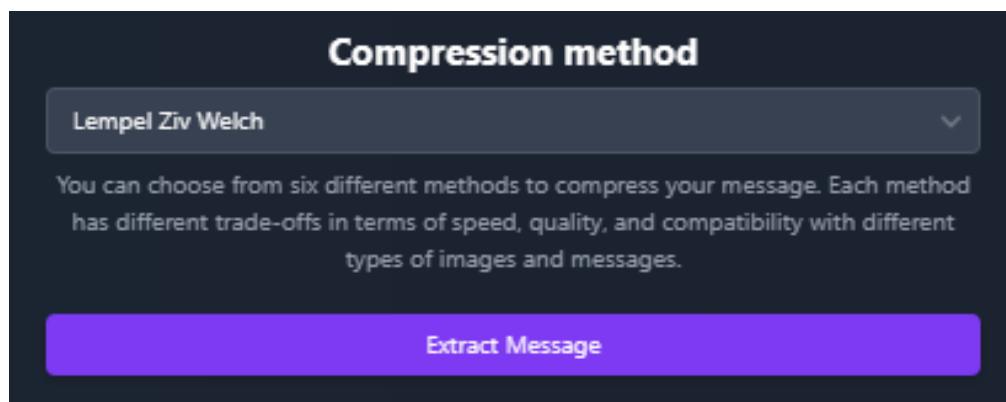


**Gambar 4.46 Unggah Stegoimage LZW**

Sumber: diolah oleh peneliti

2) Pilih metode kompresi

Pilih metode kompresi yang diinginkan. Pada bagian ini, karena peneliti akan mencoba menggunakan *Lempel-Ziv-Welch* maka peneliti pilih metode tersebut di kotak *compression method*. Setelah itu klik tombol *Extract Message*.

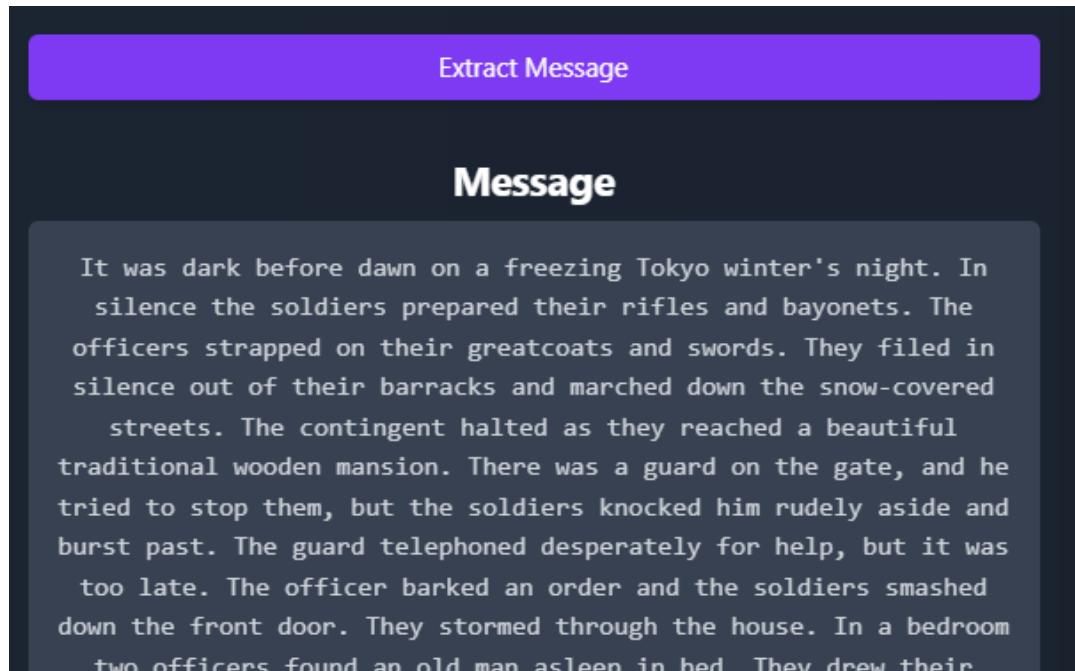


**Gambar 4.47 Pilih Metode Dekompresi LZW**

Sumber: diolah oleh peneliti

3) Membaca hasil pesan

Setelah proses ekstraksi selesai, tulisan pesan tersembunyi akan muncul pada kotak di bawah tombol *Extract Message*.



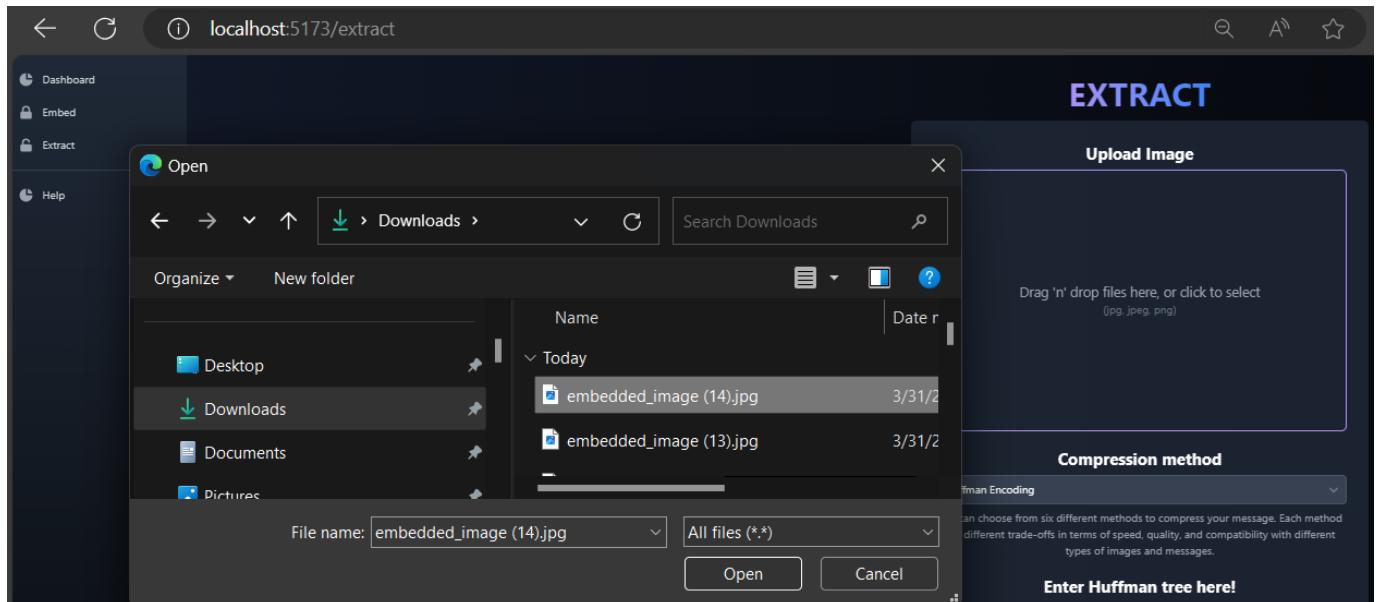
**Gambar 4.48 Membaca Hasil Pesan LZW**

Sumber: diolah oleh peneliti

#### **4.2.2.5 J-bit Encoding**

- 1) Unggah *stegoimage*

Selanjutnya untuk membaca kembali pesan yang telah disisipkan ke dalam *stegoimage* peneliti perlu mengunggah gambar yang ingin diekstrak pada laman *extract*.

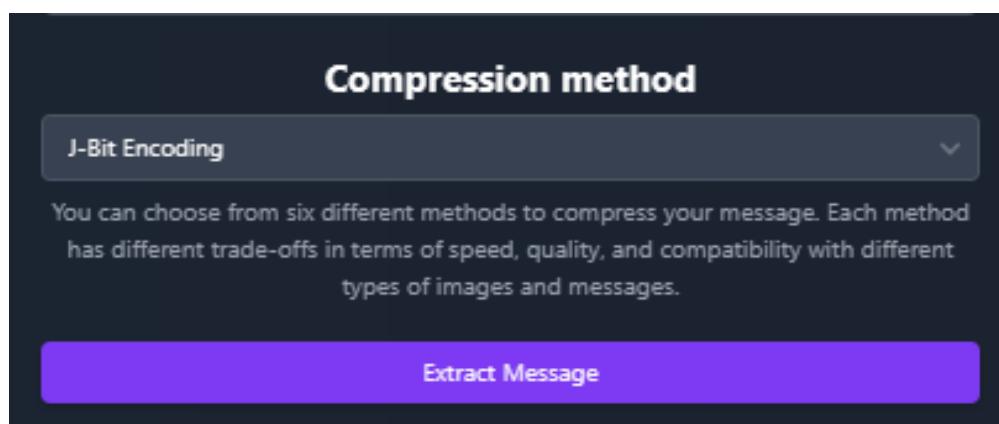


**Gambar 4.49 Unggah Stegoimage JBE**

Sumber: diolah oleh peneliti

2) Pilih metode kompresi

Pilih metode kompresi yang diinginkan. Pada bagian ini, karena peneliti akan mencoba menggunakan *J-bit Encoding* maka peneliti pilih metode tersebut di kotak *compression method*. Setelah itu klik tombol *Extract Message*.

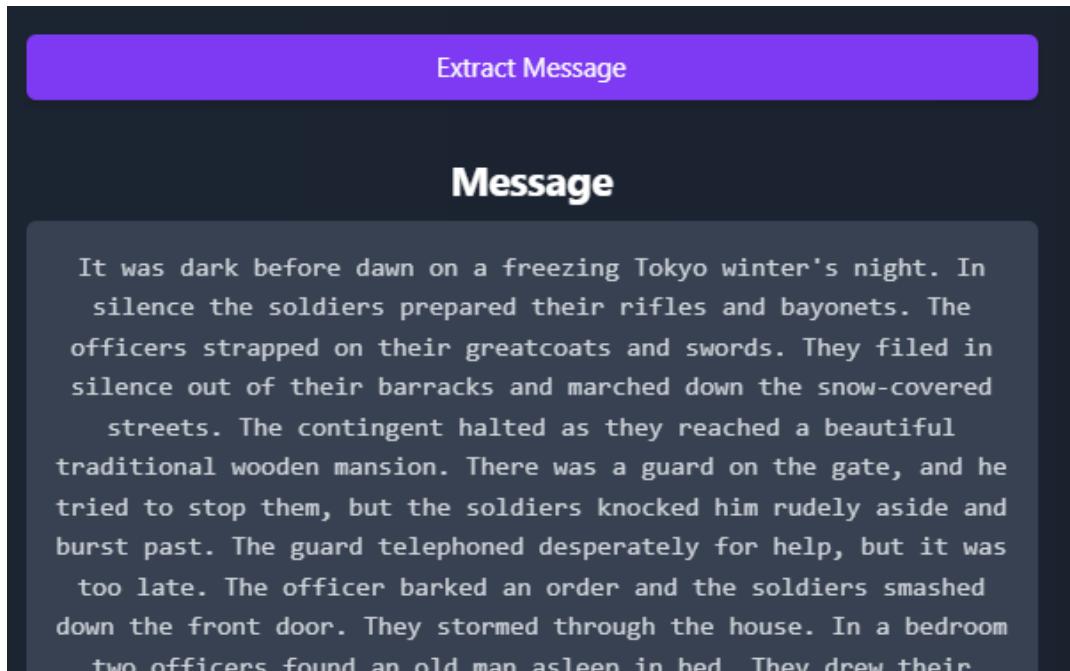


**Gambar 4.50 Pilih Metode Dekompreksi JBE**

Sumber: diolah oleh peneliti

3) Membaca hasil pesan

Setelah proses ekstraksi selesai, tulisan pesan tersembunyi akan muncul pada kotak di bawah tombol *Extract Message*.



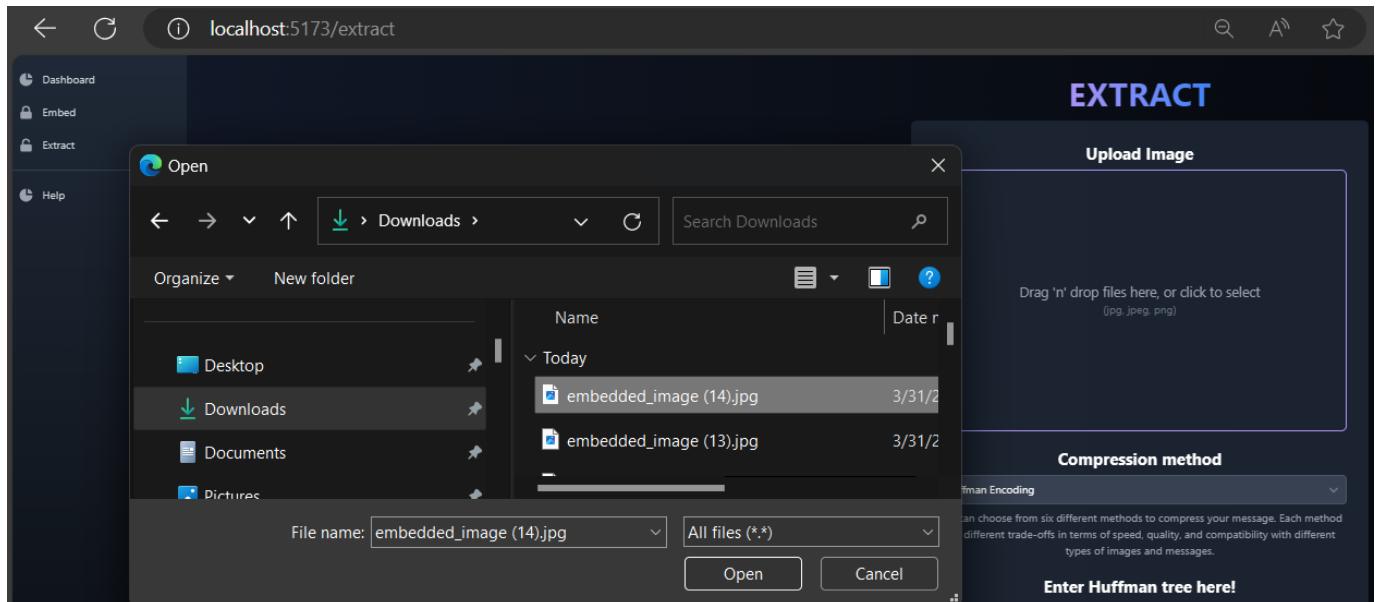
**Gambar 4.51 Membaca Hasil Pesan JBE**

Sumber: diolah oleh peneliti

#### **4.2.2.6 Deflate Compression**

- 1) Unggah *stegoimage*

Selanjutnya untuk membaca kembali pesan yang telah disisipkan ke dalam *stegoimage* peneliti perlu mengunggah gambar yang ingin diekstrak pada laman *extract*.

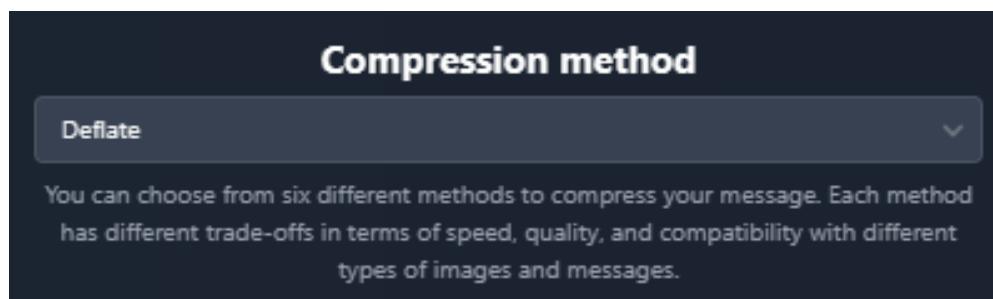


**Gambar 4.52 Unggah Stegoimage Deflate**

Sumber: diolah oleh peneliti

2) Pilih metode kompresi

Pilih metode kompresi yang diinginkan. Pada bagian ini, karena peneliti akan mencoba menggunakan *Deflate Compression* maka peneliti pilih metode tersebut di kotak *compression method*. Setelah itu klik tombol *Extract Message*.



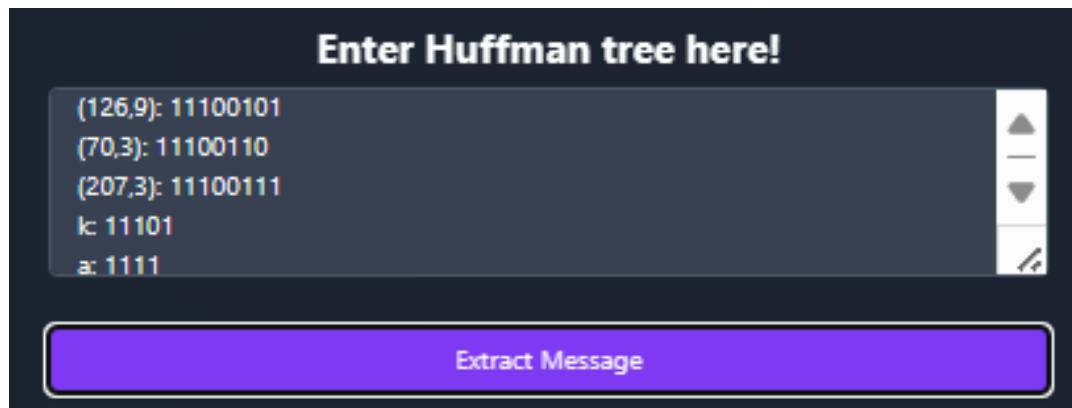
**Gambar 4.53 Pilih Metode Kompresi Deflate**

Sumber: diolah oleh peneliti

3) Tulis *Huffman tree*

Untuk memulai proses ekstraksi, metode kompresi *Deflate Compression* menggunakan kode-kode tertentu untuk menentukan

sejumlah teks bit yang akan dikonversikan menjadi karakter. Ketik kode yang sudah disalin sebelumnya untuk kemudian proses ekstraksi dapat dimulai dengan mengklik tombol *Extract Message*.

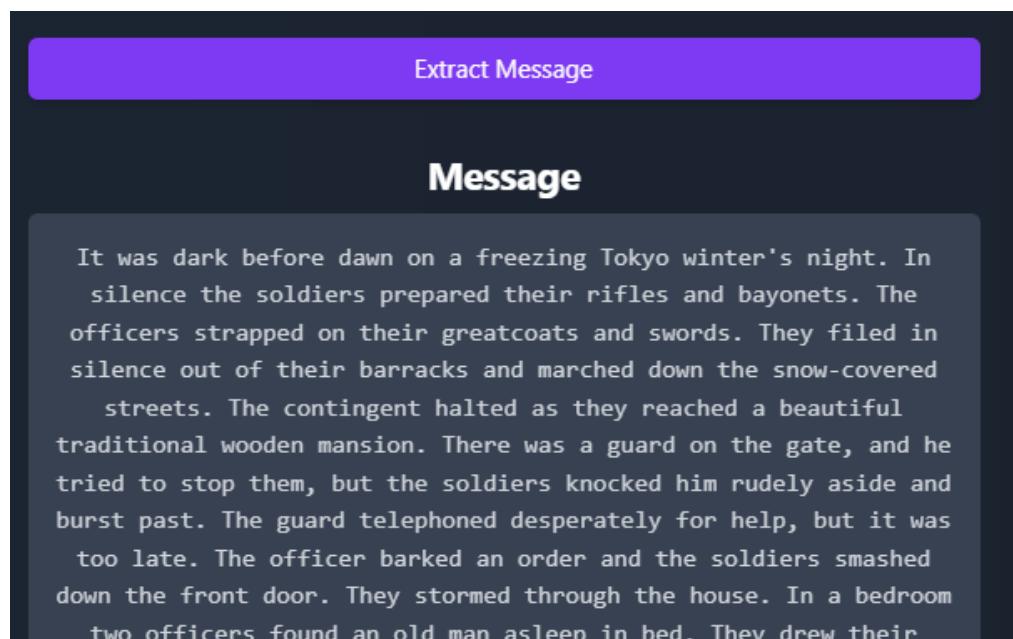


**Gambar 4.54 Tulis Pohon Huffman Deflate**

Sumber: diolah oleh peneliti

4) Membaca hasil pesan

Setelah proses ekstraksi selesai, tulisan pesan tersembunyi akan muncul pada kotak di bawah tombol *Extract Message*.



**Gambar 4. 55 Membaca Hasil Pesan**

Sumber: diolah oleh peneliti

Berdasarkan pengujian aplikasi, fungsi steganografi berhasil diterapkan dalam aplikasi web sederhana. Hal ini dapat ditandai dari pesan masukan yang sesuai dengan hasil keluaran, sehingga tidak terjadi kerusakan data pada data yang disisipkan. Hal ini menandakan fungsi steganografi telah berhasil diterapkan disertai juga dengan fungsi kompresi pada data masukannya.

### 4.3 Hasil Keluaran Data

Keluaran data merupakan nilai-nilai yang menjadi bahan ukur terhadap kualitas dari tiap-tiap algoritma. Beberapa fungsi diperlukan untuk membantu menelaah apakah hasil yang dikeluarkan oleh sistem terukur dan dapat terdokumentasikan. Variasi keluaran akan dikelompokan menjadi 3 variasi dengan masing-masing variasi diukur seberapa panjang teks keluaran dari tiap-tiap algoritma yang diuji.

#### 4.3.1 Percobaan Terhadap Variasi 1 paragraf

Pada variasi ini, algoritma akan mengerjakan kompresi terhadap teks bit yang telah dirubah dari teks biasa sepanjang 1 paragraf. Hasil dari penerapan algoritma-algoritma itu beragam yang mana secara garis besar akan dinilai apakah akan menjadi lebih kecil ukurannya atau tidak. Berikut merupakan hasil percobaan yang telah dilakukan (lihat di tabel 4.7 dan tabel 4.8).

##### A. *Run-Length Encoding*

Pengukuran pertama dilakukan pada algoritma *Run-Length Encoding*, terdapat dua keluaran yang akan dijadikan sebagai bahan evaluasi yaitu panjang bit hasil kompresi dengan lama waktu penggerjaan algoritma kompresi. Berikut merupakan pelaksanaan pengambilan hasil kinerja algoritma.

- 1) Percobaan pertama

Pada percobaan pertama didapatkan bahwa hasil bit setelah kompresi berjumlah 11296 karakter. waktu penggerjaan kompresi 0.71ms serta waktu dekompresi 0.645 ms.

2) Percobaan kedua

Kemudian peneliti melakukan percobaan kedua lalu didapatkan bahwa hasil bit setelah kompresi berjumlah 11296 karakter. waktu penggerjaan kompresi 1.419 ms serta waktu dekompresi 0.562 ms.

3) Percobaan ketiga

Terakhir pada percobaan ketiga didapatkan bahwa hasil bit setelah kompresi berjumlah 11296 karakter. waktu penggerjaan kompresi 0.367 ms serta waktu dekompresi 0.332 ms.

B. *Huffman Coding*

Kemudian dilakukan pengukuran pada algoritma *Huffman Coding*, terdapat dua keluaran yang akan dijadikan sebagai bahan evaluasi yaitu panjang bit hasil kompresi dengan lama waktu penggerjaan algoritma kompresi. Berikut merupakan pelaksanaan pengambilan hasil kinerja algoritma.

1) Percobaan pertama

Pada percobaan pertama didapatkan bahwa hasil bit setelah uji kompresi berjumlah 3022 karakter. Waktu penggerjaan kompresi 0.868 ms serta waktu dekompresi 1.459 ms.

2) Percobaan kedua

Kemudian peneliti melakukan percobaan kedua lalu didapatkan bahwa hasil bit setelah kompresi berjumlah 3022 karakter. waktu penggerjaan kompresi 0.388 ms serta waktu dekompresi 0.439 ms.

3) Percobaan ketiga

Terakhir pada percobaan ketiga didapatkan bahwa hasil bit setelah kompresi berjumlah 3022 karakter. waktu

penggerjaan kompresi 0.354 ms serta waktu dekompresi 0.904 ms.

### C. *Arithmetic Coding*

Selanjutnya ada pengukuran yang dilakukan pada algoritma *Arithmetic Coding*, terdapat dua keluaran yang akan dijadikan sebagai bahan evaluasi yaitu panjang bit hasil kompresi dengan lama waktu penggerjaan algoritma kompresi. Berikut merupakan pelaksanaan pengambilan hasil kinerja algoritma.

#### 1) Percobaan pertama

Pada percobaan pertama didapatkan bahwa hasil bit setelah kompresi berjumlah 3003 karakter. waktu penggerjaan kompresi 7.596 ms serta waktu dekompresi 28.655 ms.

#### 2) Percobaan kedua

Kemudian peneliti melakukan percobaan kedua lalu didapatkan bahwa hasil bit setelah kompresi berjumlah 3003 karakter. waktu penggerjaan kompresi 4.995 ms serta waktu dekompresi 27.173 ms.

#### 3) Percobaan ketiga

Terakhir pada percobaan ketiga didapatkan bahwa hasil bit setelah kompresi berjumlah 3003 karakter. waktu penggerjaan kompresi 6.419 ms serta waktu dekompresi 27.271 ms.

### D. *Lempel-Ziv-Welch*

Keempat dilakukan pengukuran pada algoritma *Lempel-Ziv-Welch*, terdapat dua keluaran yang akan dijadikan sebagai bahan evaluasi yaitu panjang bit hasil kompresi dengan lama waktu penggerjaan algoritma kompresi. Berikut merupakan pelaksanaan pengambilan hasil kinerja algoritma.

#### 1) Percobaan pertama

Pada percobaan pertama didapatkan bahwa hasil bit setelah kompresi berjumlah 4884 karakter. waktu penggerjaan kompresi 0.548 ms serta waktu dekompresi 0.743 ms.

2) Percobaan kedua

Kemudian peneliti melakukan percobaan kedua lalu didapatkan bahwa hasil bit setelah kompresi berjumlah 4884 karakter. waktu penggerjaan kompresi 0.531 ms serta waktu dekompresi 2.075 ms.

3) Percobaan ketiga

Terakhir pada percobaan ketiga didapatkan bahwa hasil bit setelah kompresi berjumlah 4884 karakter. waktu penggerjaan kompresi 0.363 ms serta waktu dekompresi 0.217 ms.

#### E. *J-bit Encoding*

Kelima pengukuran dilakukan pada algoritma *J-bit Encoding*, terdapat dua keluaran yang akan dijadikan sebagai bahan evaluasi yaitu panjang bit hasil kompresi dengan lama waktu penggerjaan algoritma kompresi. Berikut merupakan pelaksanaan pengambilan hasil kinerja algoritma.

1) Percobaan pertama

Pada percobaan pertama didapatkan bahwa hasil bit setelah kompresi berjumlah 6496 karakter. waktu penggerjaan kompresi 0.312 ms serta waktu dekompresi 1.035 ms.

2) Percobaan kedua

Kemudian peneliti melakukan percobaan kedua lalu didapatkan bahwa hasil bit setelah kompresi berjumlah 6496 karakter. waktu penggerjaan kompresi 0.236 ms serta waktu dekompresi 0.272 ms.

3) Percobaan ketiga

Terakhir pada percobaan ketiga didapatkan bahwa hasil bit setelah kompresi berjumlah 6496 karakter. waktu penggerjaan kompresi 1.181 ms serta waktu dekompresi 0.392 ms.

#### *F. Deflate Compression*

Terakhir dilakukan pengukuran pada algoritma *Deflate Compression*, terdapat dua keluaran yang akan dijadikan sebagai bahan evaluasi yaitu panjang bit hasil kompresi dengan lama waktu penggerjaan algoritma kompresi. Berikut merupakan pelaksanaan pengambilan hasil kinerja algoritma.

- 1) Percobaan pertama

Pada percobaan pertama didapatkan bahwa hasil bit setelah kompresi berjumlah 2295 karakter. waktu penggerjaan kompresi 7.075 ms serta waktu dekompresi 1.828 ms.

- 2) Percobaan kedua

Kemudian peneliti melakukan percobaan kedua lalu didapatkan bahwa hasil bit setelah kompresi berjumlah 2295 karakter. waktu penggerjaan kompresi 1.279 ms serta waktu dekompresi 1.282 ms.

- 3) Percobaan ketiga

Terakhir pada percobaan ketiga didapatkan bahwa hasil bit setelah kompresi berjumlah 2295 karakter. waktu penggerjaan kompresi 6.346 ms serta waktu dekompresi 0.964 ms.

#### **4.3.2 Percobaan Terhadap Variasi 5 paragraf**

Pada variasi ini, algoritma akan mengerjakan kompresi terhadap teks bit yang telah dirubah dari teks biasa sepanjang 5 paragraf. Hasil dari penerapan algoritma-algoritma itu beragam yang mana secara garis besar

akan dinilai apakah akan menjadi lebih kecil ukurannya atau tidak. Berikut merupakan hasil percobaan yang telah dilakukan.

#### A. *Run-Length Encoding*

Pengukuran pertama dilakukan pada algoritma *Run-Length Encoding*, terdapat dua keluaran yang akan dijadikan sebagai bahan evaluasi yaitu panjang bit hasil kompresi dengan lama waktu penggerjaan algoritma kompresi. Berikut merupakan pelaksanaan pengambilan hasil kinerja algoritma.

##### 1) Percobaan pertama

Pada percobaan pertama didapatkan bahwa hasil bit setelah kompresi berjumlah 64080 karakter. waktu penggerjaan kompresi 4.349 ms serta waktu dekompresi 1.635 ms.

##### 2) Percobaan kedua

Kemudian peneliti melakukan percobaan kedua lalu didapatkan bahwa hasil bit setelah kompresi berjumlah 64080 karakter. waktu penggerjaan kompresi 3.11 ms serta waktu dekompresi 2.427 ms.

##### 3) Percobaan ketiga

Terakhir pada percobaan ketiga didapatkan bahwa hasil bit setelah kompresi berjumlah 64080 karakter. waktu penggerjaan kompresi 2.591 ms serta waktu dekompresi 0.969 ms.

#### B. *Huffman Coding*

Kemudian dilakukan pengukuran pada algoritma *Huffman Coding*, terdapat dua keluaran yang akan dijadikan sebagai bahan evaluasi yaitu panjang bit hasil kompresi dengan lama waktu penggerjaan algoritma kompresi. Berikut merupakan pelaksanaan pengambilan hasil kinerja algoritma.

##### 1) Percobaan pertama

Pada percobaan pertama didapatkan bahwa hasil bit setelah kompresi berjumlah 18047 karakter. waktu penggerjaan kompresi 1.238 ms serta waktu dekompresi 3.545 ms.

2) Percobaan kedua

Kemudian peneliti melakukan percobaan kedua lalu didapatkan bahwa hasil bit setelah kompresi berjumlah 18047 karakter. waktu penggerjaan kompresi 1.328 ms serta waktu dekompresi 3.029 ms.

3) Percobaan ketiga

Terakhir pada percobaan ketiga didapatkan bahwa hasil bit setelah kompresi berjumlah 18047 karakter. waktu penggerjaan kompresi 1.735 ms serta waktu dekompresi 3.834 ms.

### C. *Arithmetic Coding*

Selanjutnya ada pengukuran yang dilakukan pada algoritma *Arithmetic Coding*, terdapat dua keluaran yang akan dijadikan sebagai bahan evaluasi yaitu panjang bit hasil kompresi dengan lama waktu penggerjaan algoritma kompresi. Berikut merupakan pelaksanaan pengambilan hasil kinerja algoritma.

1) Percobaan pertama

Pada percobaan pertama didapatkan bahwa hasil bit setelah kompresi berjumlah 17898 karakter. waktu penggerjaan kompresi 119.091 ms serta waktu dekompresi 2.711 s.

2) Percobaan kedua

Kemudian peneliti melakukan percobaan kedua lalu didapatkan bahwa hasil bit setelah kompresi berjumlah 17898 karakter. waktu penggerjaan kompresi 90.257 ms serta waktu dekompresi 1.941 s.

3) Percobaan ketiga

Terakhir pada percobaan ketiga didapatkan bahwa hasil bit setelah kompresi berjumlah 17898 karakter. waktu penggerjaan kompresi 68.277 ms serta waktu dekompresi 1.99 s.

#### D. *Lempel-Ziv-Welch*

Keempat dilakukan pengukuran pada algoritma *Lempel-Ziv-Welch*, terdapat dua keluaran yang akan dijadikan sebagai bahan evaluasi yaitu panjang bit hasil kompresi dengan lama waktu penggerjaan algoritma kompresi. Berikut merupakan pelaksanaan pengambilan hasil kinerja algoritma.

##### 1) Percobaan pertama

Pada percobaan pertama didapatkan bahwa hasil bit setelah kompresi berjumlah 21612 karakter. waktu penggerjaan kompresi 3.268 ms serta waktu dekompresi 1.086 ms.

##### 2) Percobaan kedua

Kemudian peneliti melakukan percobaan kedua lalu didapatkan bahwa hasil bit setelah kompresi berjumlah 21612 karakter. waktu penggerjaan kompresi 2.802 ms serta waktu dekompresi 1.852 ms.

##### 3) Percobaan ketiga

Terakhir pada percobaan ketiga didapatkan bahwa hasil bit setelah kompresi berjumlah 21612 karakter. waktu penggerjaan kompresi 1.748 ms serta waktu dekompresi 1.021 ms.

#### E. *J-bit Encoding*

Kelima pengukuran dilakukan pada algoritma *J-bit Encoding*, terdapat dua keluaran yang akan dijadikan sebagai bahan evaluasi yaitu panjang bit hasil kompresi dengan lama waktu penggerjaan algoritma kompresi. Berikut merupakan pelaksanaan pengambilan hasil kinerja algoritma.

- 1) Percobaan pertama

Pada percobaan pertama didapatkan bahwa hasil bit setelah kompresi berjumlah 36496 karakter. waktu penggerjaan kompresi 1.553 ms serta waktu dekompresi 2.895 ms.

- 2) Percobaan kedua

Kemudian peneliti melakukan percobaan kedua lalu didapatkan bahwa hasil bit setelah kompresi berjumlah 36496 karakter. waktu penggerjaan kompresi 0.739 ms serta waktu dekompresi 2.097 ms.

- 3) Percobaan ketiga

Terakhir pada percobaan ketiga didapatkan bahwa hasil bit setelah kompresi berjumlah 36496 karakter. waktu penggerjaan kompresi 1.151 ms serta waktu dekompresi 3.224 ms.

#### *F. Deflate Compression*

Terakhir dilakukan pengukuran pada algoritma *Deflate Compression*, terdapat dua keluaran yang akan dijadikan sebagai bahan evaluasi yaitu panjang bit hasil kompresi dengan lama waktu penggerjaan algoritma kompresi. Berikut merupakan pelaksanaan pengambilan hasil kinerja algoritma.

- 1) Percobaan pertama

Pada percobaan pertama didapatkan bahwa hasil bit setelah kompresi berjumlah 13694 karakter. waktu penggerjaan kompresi 22.185 ms serta waktu dekompresi 18.711 ms.

- 2) Percobaan kedua

Pada percobaan pertama didapatkan bahwa hasil bit setelah kompresi berjumlah 13694 karakter. waktu penggerjaan kompresi 34.696 ms serta waktu dekompresi 3.931 ms.

3) Percobaan ketiga

Pada percobaan pertama didapatkan bahwa hasil bit setelah kompresi berjumlah 13694 karakter. waktu penggerjaan kompresi 22.136 ms serta waktu dekompresi 5.528 ms.

#### **4.3.3 Percobaan Terhadap Variasi 10 paragraf**

Pada variasi ini, algoritma akan mengerjakan kompresi terhadap teks bit yang telah dirubah dari teks biasa sepanjang 10 paragraf. Hasil dari penerapan algoritma-algoritma itu beragam yang mana secara garis besar akan dinilai apakah akan menjadi lebih kecil ukurannya atau tidak. Berikut merupakan hasil percobaan yang telah dilakukan.

##### *A. Run-Length Encoding*

Pengukuran pertama dilakukan pada algoritma *Run-Length Encoding*, terdapat dua keluaran yang akan dijadikan sebagai bahan evaluasi yaitu panjang bit hasil kompresi dengan lama waktu penggerjaan algoritma kompresi. Berikut merupakan pelaksanaan pengambilan hasil kinerja algoritma.

1) Percobaan pertama

Pada percobaan pertama didapatkan bahwa hasil bit setelah kompresi berjumlah 108912 karakter. waktu penggerjaan kompresi 4.852 ms serta waktu dekompresi 2.873 ms.

2) Percobaan kedua

Kemudian peneliti melakukan percobaan kedua lalu didapatkan bahwa hasil bit setelah kompresi berjumlah 108912 karakter. waktu penggerjaan kompresi 4.651 ms serta waktu dekompresi 2.808 ms.

3) Percobaan ketiga

Terakhir pada percobaan ketiga didapatkan bahwa hasil bit setelah kompresi berjumlah 108912 karakter. waktu

penggerjaan kompresi 2.823 ms serta waktu dekompresi 1.955 ms.

#### B. *Huffman Coding*

Kemudian dilakukan pengukuran pada algoritma *Huffman Coding*, terdapat dua keluaran yang akan dijadikan sebagai bahan evaluasi yaitu panjang bit hasil kompresi dengan lama waktu penggerjaan algoritma kompresi. Berikut merupakan pelaksanaan pengambilan hasil kinerja algoritma.

##### 1) Percobaan pertama

Pada percobaan pertama didapatkan bahwa hasil bit setelah kompresi berjumlah 30709 karakter. waktu penggerjaan kompresi 3.614 ms serta waktu dekompresi 7.113 ms.

##### 2) Percobaan kedua

Pada percobaan pertama didapatkan bahwa hasil bit setelah kompresi berjumlah 30709 karakter. waktu penggerjaan kompresi 1.861ms serta waktu dekompresi 3.425 ms.

##### 3) Percobaan ketiga

Pada percobaan pertama didapatkan bahwa hasil bit setelah kompresi berjumlah 30709 karakter. waktu penggerjaan kompresi 1.412 ms serta waktu dekompresi 4.283 ms.

#### C. *Arithmetic Coding*

Selanjutnya ada pengukuran yang dilakukan pada algoritma *Arithmetic Coding*, terdapat dua keluaran yang akan dijadikan sebagai bahan evaluasi yaitu panjang bit hasil kompresi dengan lama waktu penggerjaan algoritma kompresi. Berikut merupakan pelaksanaan pengambilan hasil kinerja algoritma.

##### 1) Percobaan pertama

Pada percobaan pertama didapatkan bahwa hasil bit setelah kompresi berjumlah 30448 karakter. waktu penggerjaan kompresi 357.16 ms serta waktu dekompresi 8.316 s.

2) Percobaan kedua

Kemudian peneliti melakukan percobaan kedua lalu didapatkan bahwa hasil bit setelah kompresi berjumlah 30448 karakter. waktu penggerjaan kompresi 221.61 ms serta waktu dekompresi 7.549 s.

3) Percobaan ketiga

Terakhir pada percobaan ketiga didapatkan bahwa hasil bit setelah kompresi berjumlah 30448 karakter. waktu penggerjaan kompresi 227.169 ms serta waktu dekompresi 7.786 s.

#### D. *Lempel-Ziv-Welch*

Keempat dilakukan pengukuran pada algoritma *Lempel-Ziv-Welch*, terdapat dua keluaran yang akan dijadikan sebagai bahan evaluasi yaitu panjang bit hasil kompresi dengan lama waktu penggerjaan algoritma kompresi. Berikut merupakan pelaksanaan pengambilan hasil kinerja algoritma.

1) Percobaan pertama

Pada percobaan pertama didapatkan bahwa hasil bit setelah kompresi berjumlah 33240 karakter. waktu penggerjaan kompresi 4.177 ms serta waktu dekompresi 1.244 ms.

2) Percobaan kedua

Kemudian peneliti melakukan percobaan kedua lalu didapatkan bahwa hasil bit setelah kompresi berjumlah 33240 karakter. waktu penggerjaan kompresi 2.161 ms serta waktu dekompresi 1.771 ms.

3) Percobaan ketiga

Terakhir pada percobaan ketiga didapatkan bahwa hasil bit setelah kompresi berjumlah 33240 karakter. waktu penggerjaan kompresi 2.327 ms serta waktu dekompresi 1.592 ms.

#### E. *J-bit Encoding*

Kelima pengukuran dilakukan pada algoritma *J-bit Encoding*, terdapat dua keluaran yang akan dijadikan sebagai bahan evaluasi yaitu panjang bit hasil kompresi dengan lama waktu penggerjaan algoritma kompresi. Berikut merupakan pelaksanaan pengambilan hasil kinerja algoritma.

##### 1) Percobaan pertama

Pada percobaan pertama didapatkan bahwa hasil bit setelah kompresi berjumlah 62184 karakter. waktu penggerjaan kompresi 13.782 ms serta waktu dekompresi 2.699 ms.

##### 2) Percobaan kedua

Kemudian peneliti melakukan percobaan kedua lalu didapatkan bahwa hasil bit setelah kompresi berjumlah 62184 karakter. waktu penggerjaan kompresi 2.079 ms serta waktu dekompresi 1.939 ms.

##### 3) Percobaan ketiga

Terakhir pada percobaan ketiga didapatkan bahwa hasil bit setelah kompresi berjumlah 62184 karakter. waktu penggerjaan kompresi 1.385 ms serta waktu dekompresi 1.438 ms.

#### F. *Deflate Compression*

Terakhir dilakukan pengukuran pada algoritma *Deflate Compression*, terdapat dua keluaran yang akan dijadikan sebagai bahan evaluasi yaitu panjang bit hasil kompresi dengan lama waktu penggerjaan algoritma kompresi. Berikut merupakan pelaksanaan pengambilan hasil kinerja algoritma.

1) Percobaan pertama

Pada percobaan pertama didapatkan bahwa hasil bit setelah kompresi berjumlah 23810 karakter. waktu penggerjaan kompresi 50.431 ms serta waktu dekompresi 13.032 ms.

2) Percobaan kedua

Kemudian peneliti melakukan percobaan kedua lalu didapatkan bahwa hasil bit setelah kompresi berjumlah 23810 karakter. waktu penggerjaan kompresi 66.439 ms serta waktu dekompresi 12.153 ms.

3) Percobaan ketiga

Terakhir pada percobaan ketiga didapatkan bahwa hasil bit setelah kompresi berjumlah 23810 karakter. waktu penggerjaan kompresi 75.273 ms serta waktu dekompresi 8.966 ms.

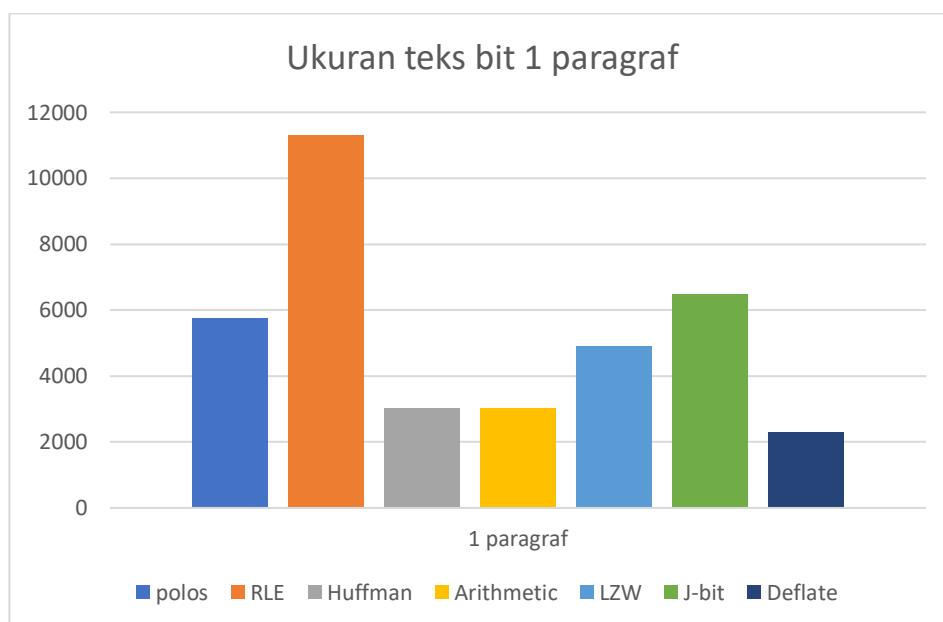
Berdasarkan hasil keluaran yang diujikan, fungsi kompresi berhasil digunakan dalam aplikasi web sederhana. Hal ini membuktikan bahwa integrasi antara fungsi kompresi terhadap data masukan dengan fungsi steganografi untuk menyisipkan pesan rahasia ke dalam citra digital berjalan sesuai dengan kerangka berpikir yang telah dirancang. Kemudian selanjutnya hasil keluaran akan dibahas pada sub bab selanjutnya yaitu Analisa Algoritma.

#### **4.4 Analisa Algoritma**

Berikut merupakan analisa dari peforma berdasarkan penggunaan metode algoritma terhadap variasi teks. Berdasarkan ukuran teks, penggunaan metode algoritma memberikan hasil yang bervariatif. Hal ini dapat dilihat dengan ukuran sebelum dan setelah dilakukannya kompresi menggunakan algoritma kompresi.

Melalui grafik 4.1 peneliti dapat melihat bahwa algoritma *Deflate Compression* memiliki konsistensi dalam memberikan hasil kompresi yang

paling baik dari setiap variasi yang diberikan. Dari gambar terlihat bahwa pada variasi 1 paragraf, tanpa menggunakan kompresi panjang teks yang tercatat adalah sepanjang 5744 bit. Setelah perlakuan kompresi *Deflate Compression* panjang teks menyusut menjadi 2295 bit. Hal ini menunjukkan bahwa terdapat kompresi sejumlah 3449 bit hasil dari penggunaan algoritma *Deflate Compression* terhadap teks. Kemudian terlihat bahwa terdapat metode kompresi terburuk yaitu *Run-Length Encoding*. Hal ini terlihat dari besarnya hasil kompresi yang mana menunjukkan meningkatnya jumlah bit menjadi 11296 bit. Hal ini menunjukkan terdapat peningkatan jumlah bit sebesar 5552 bit.

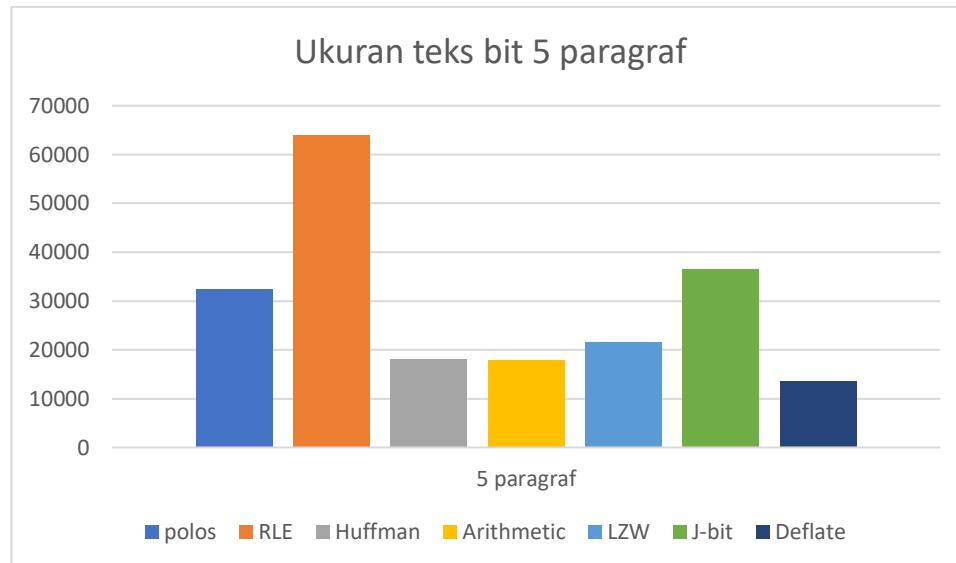


**Grafik 4.1 Ukuran Bit Teks 1 Paragraf**

Sumber: diolah oleh peneliti

Selanjutnya dari variasi 5 paragraf, tanpa menggunakan kompresi panjang teks yang tercatat adalah sepanjang 32408 bit. Setelah perlakuan kompresi *Deflate Compression* panjang teks menyusut menjadi 13694 bit. Hal ini menunjukkan bahwa terdapat kompresi sejumlah 18714 bit hasil dari penggunaan algoritma *Deflate Compression* terhadap teks. Kemudian terlihat bahwa terdapat metode kompresi terburuk yaitu *Run-Length Encoding*. Hal ini terlihat dari besarnya hasil kompresi yang mana

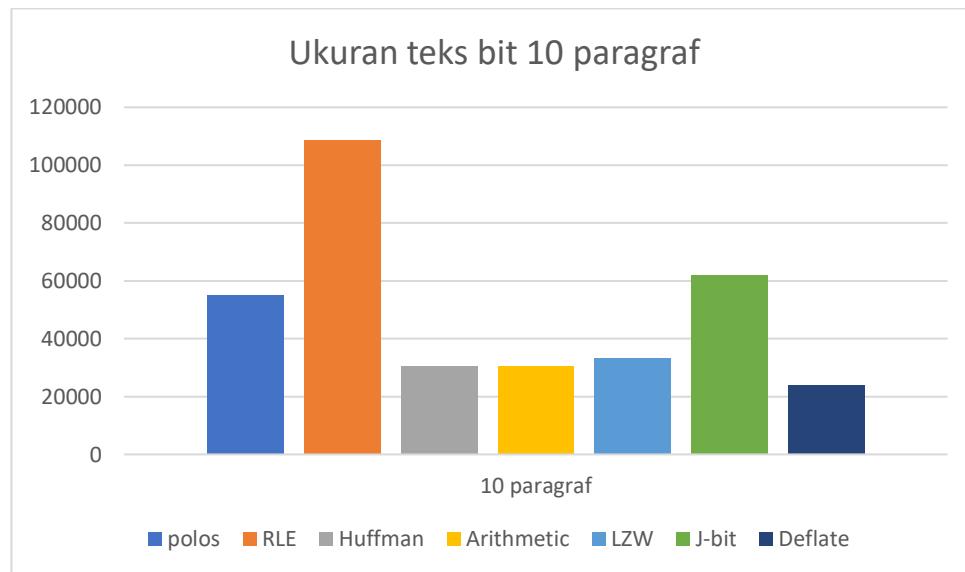
menunjukkan meningkatnya jumlah bit menjadi 64080 bit. Hal ini menunjukkan terdapat peningkatan jumlah bit sebesar 31672 bit.



**Grafik 4.2 Ukuran Teks Bit 5 Paragraf**

Sumber: diolah oleh peneliti

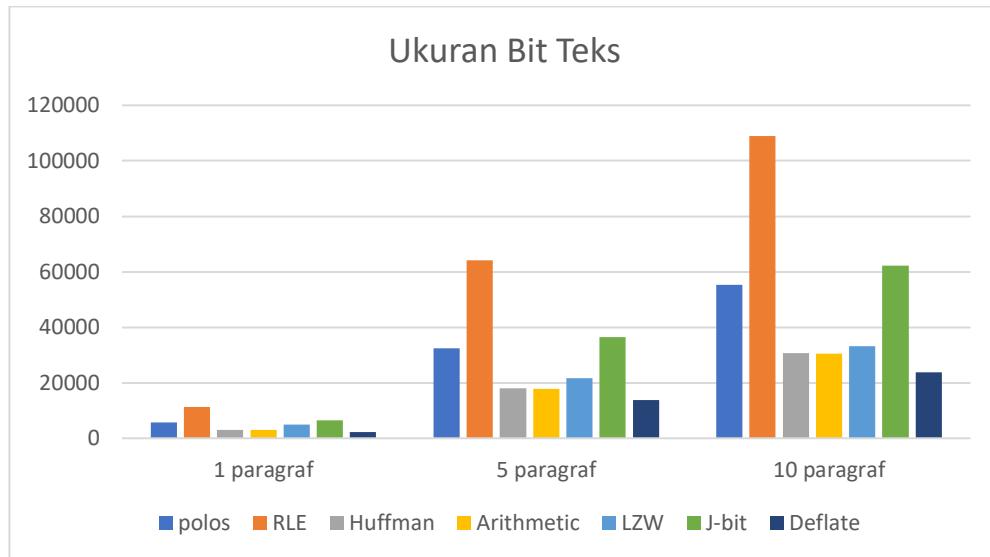
Terakhir ada variasi 10 paragraf, tanpa menggunakan kompresi panjang teks yang tercatat adalah sepanjang 55240 bit. Setelah perlakuan kompresi *Deflate Compression* panjang teks menyusut menjadi 23810 bit. Hal ini menunjukkan bahwa terdapat kompresi sejumlah 31430 bit hasil dari penggunaan algoritma *Deflate Compression* terhadap teks. Kemudian terlihat bahwa terdapat metode kompresi terburuk yaitu *Run-Length Encoding*. Hal ini terlihat dari besarnya hasil kompresi yang mana menunjukkan meningkatnya jumlah bit menjadi 108912 bit. Hal ini menunjukkan terdapat peningkatan jumlah bit sebesar 53672 bit.



**Grafik 4.3 Ukuran Teks Bit 10 Paragraf**

Sumber: diolah oleh peneliti

Secara garis besar, *Deflate Compression* memberikan hasil keluaran yang konsisten dengan memberikan keluaran dengan panjang teks bit terpendek. Sebaliknya *Run-Length Encoding* memberikan keluaran teks terpanjang dari ketiga variasi teks yang digunakan. Berikut merupakan penampakan secara umum dari hasil kompresi yang dilakukan dari setiap algoritma.



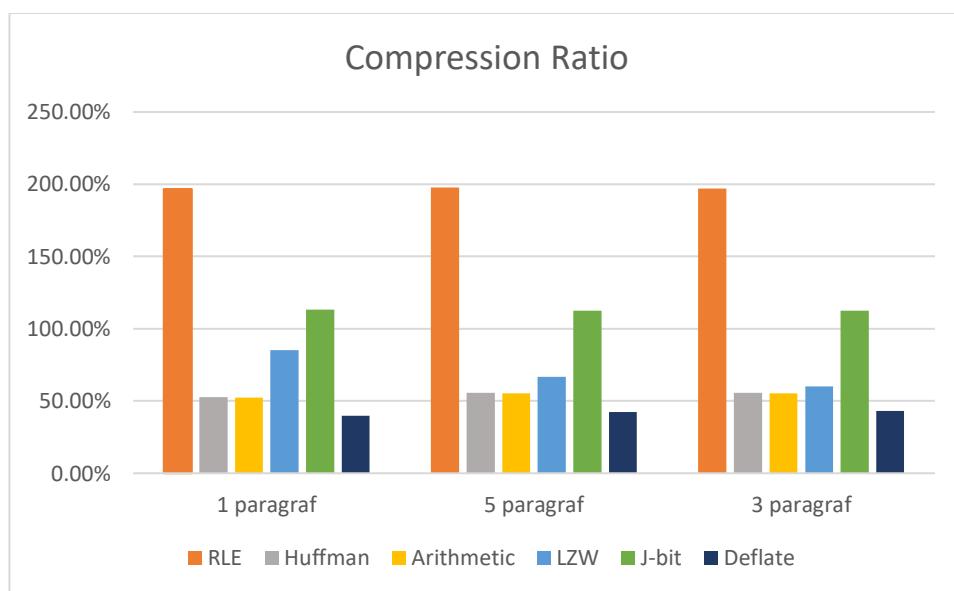
**Grafik 4.4 Ukuran Bit Teks**

Sumber: diolah oleh peneliti

Berdasarkan persentase hasil kompresi terhadap teks dasar dapat dilihat dari grafik 4.5 berikut. Dari grafik 4.5 terlihat bahwa pada variasi 1 paragraf, Setelah perlakuan kompresi *Deflate Compression* panjang teks menyusut menjadi 39.95% dari teks semula. Hal ini menunjukkan bahwa kompresi menghemat penyimpanan sebesar 60.05% dari teks awal. Kemudian terlihat bahwa terdapat metode kompresi terburuk yaitu *Run-Length Encoding*. Hal ini terlihat dari besarnya hasil kompresi yang mana menunjukkan meningkatnya jumlah bit menjadi 196.66% dari teks awal. Hal ini menunjukkan terdapat penghematan ruang penyimpanan sebesar -96.66% dari teks dasar.

Selanjutnya dari variasi 5 paragraf, Setelah perlakuan kompresi *Deflate Compression* panjang teks menyusut menjadi 42.25% dari teks semula. Hal ini menunjukkan bahwa kompresi menghemat penyimpanan sebesar 57.75% dari teks awal. Kemudian terlihat bahwa terdapat metode kompresi terburuk yaitu *Run-Length Encoding*. Hal ini terlihat dari besarnya hasil kompresi yang mana menunjukkan meningkatnya jumlah bit menjadi 197.73% dari teks awal. Hal ini menunjukkan terdapat penghematan ruang penyimpanan sebesar -97.73% dari teks dasar.

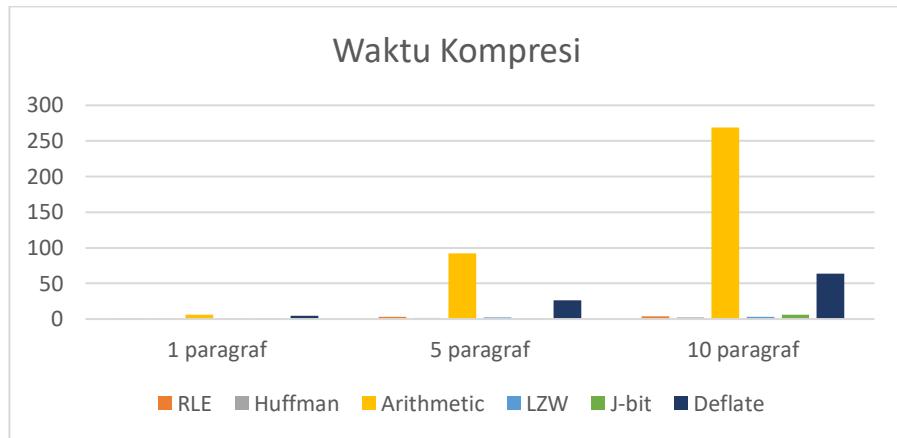
Terakhir ada variasi 10 paragraf, Setelah perlakuan kompresi *Deflate Compression* panjang teks menyusut menjadi 43.10% dari teks semula. Hal ini menunjukan bahwa kompresi menghemat penyimpanan sebesar 57.90% dari teks awal. Kemudian terlihat bahwa terdapat metode kompresi terburuk yaitu *Run-Length Encoding*. Hal ini terlihat dari besarnya hasil kompresi yang mana menunjukan meningkatnya jumlah bit menjadi 197.16% dari teks awal. Hal ini menunjukkan terdapat penghematan ruang penyimpanan sebesar -97.16% dari teks dasar.



**Grafik 4.5 Compression Ratio**

Sumber: diolah oleh peneliti

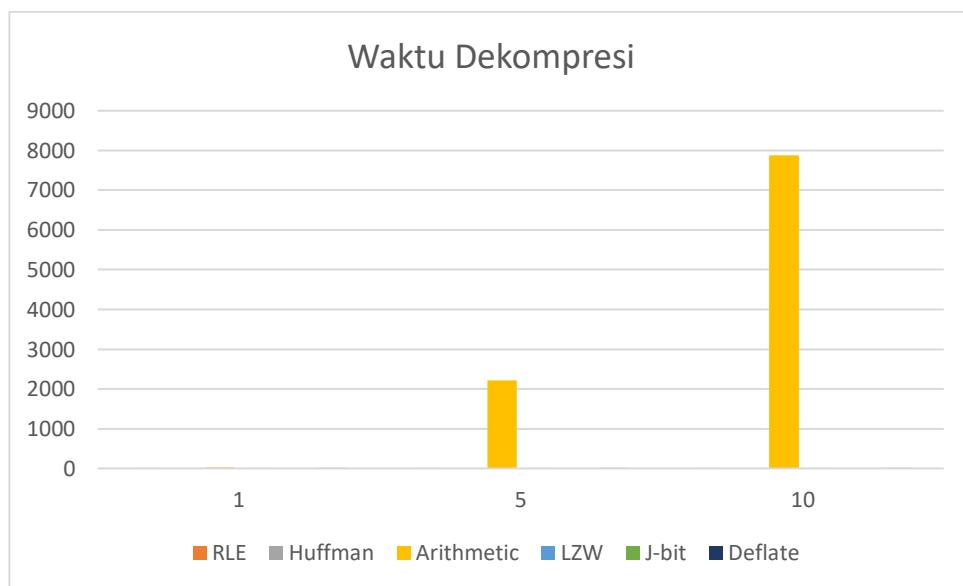
Kemudian dapat dilihat waktu penggerjaan dari tiap-tiap algoritma dalam grafik 4.6 berikut. Dari gambar dapat dilihat bahwa terdapat peningkatan waktu kompresi secara eksponensial dari metode *Arithmetic Coding* yang berbanding lurus dengan besarnya teks. Ada juga yang memiliki waktu yang relatif lambat tetapi tidak mengalami kenaikan yang eksponensial yaitu *Deflate Compression* yang dapat dilihat dengan menggunakan variasi 1 paragraf dan 5 paragraf metode ini merupakan metode paling lambat tetapi pada variasi 10 paragraf metode ini tidak lagi merupakan metode yang paling lambat.



**Grafik 4.6 Waktu Kompresi**

Sumber: diolah oleh peneliti

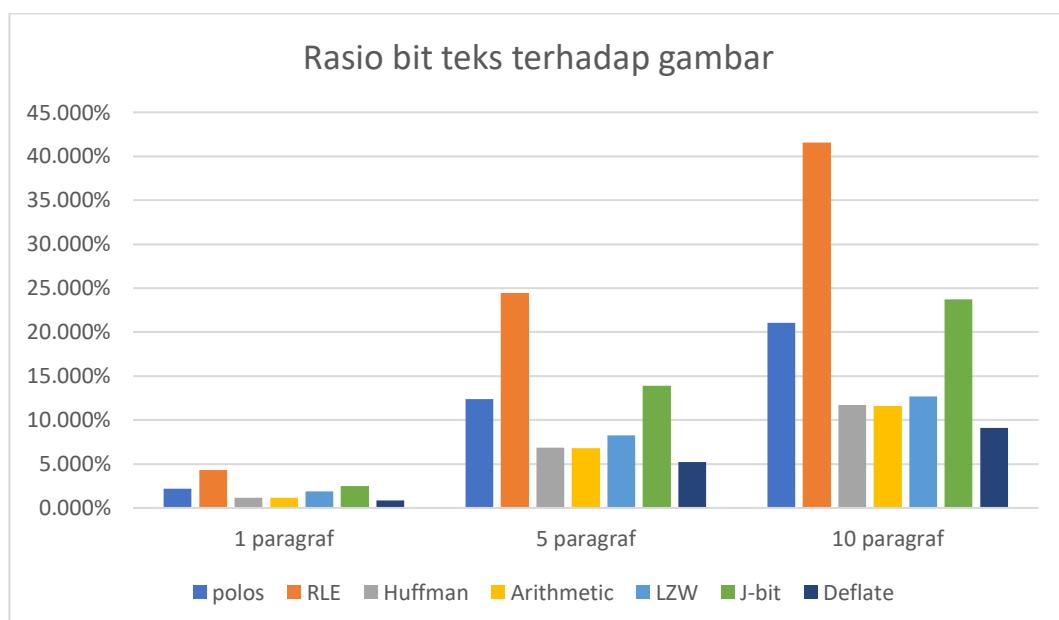
Selanjutnya dari grafik 4.7 dapat dilihat bahwa terdapat peningkatan rata-rata waktu dekompresi secara eksponensial dari metode *Arithmetic Coding* yang berbanding lurus dengan besarnya teks. Kemudian ada pun perbedaan rata-rata waktu pengerjaan dari tiap metode lain relatif mirip yaitu berkisar antara 0.5 ms hingga 4.9 ms.



**Grafik 4.7 Waktu Dekompressi**

Sumber: diolah oleh peneliti

Berdasarkan spesifikasi *coverimage* yang digunakan, dapat digambarkan rasio dari teks yang telah dikompresi terhadap gambar yang dapat dilihat dari grafik 4.8 berikut. Dapat dilihat pada hasil rasio mendapatkan nilai yang konsisten untuk metode Run-Length Encoding memberikan rasio perbandingan terhadap gambar berdimensi 512x512 yang paling buruk baik pada variasi 1 paragraf, 5 paragraf, maupun 10 paragraf yaitu secara berturut-turut mendapatkan nilai 4.309%, 24.445%, dan 41.547%. Serta metode yang memiliki rasio perbandingan terhadap *coverimage* yang paling baik terhadap ketiga variasi adalah metode *deflate compression* yang mendapatkan nilai berturut-turut yaitu: 0.875%, 5.224%, 9.083%.



**Grafik 4.8 Rasio Bit Teks Terhadap Gambar**

Sumber: diolah oleh peneliti

## 4.5 Penerapan Metode Perbandingan Eksponensial

### A. 1 Paragraf

Berdasarkan hasil yang sudah dipaparkan di bagian sebelumnya, berikut merupakan tabel yang memberikan ringkasan secara umum atas performa pada algoritma di variasi teks satu paragraf.

Tabel 4.1 Performa Algoritma Variasi Teks 1 Paragraf

Alternatif	Sebelum Kompresi	Sesudah Kompresi
RLE	5744	11296
Huffman	5744	3022
Arithmetic	5744	3003
LZW	5744	4884
J-bit	5744	6496
Deflate	5744	2295

Sumber: diolah oleh peneliti

Berdasarkan data di atas dapat dihitung parameter kinerja kompresinya yang digunakan seperti di bawah ini:

1) *Ratio Of Compression*

$$\text{Run-Length Encoding} = \frac{5744}{11296} = 0.508498584$$

$$\text{Huffman Coding} = \frac{5744}{3022} = 1.900727995$$

$$\text{Arithmetic Coding} = \frac{5744}{3003} = 1.912753913$$

$$\text{Lempel-Ziv-Welch} = \frac{5744}{4884} = 1.176085176$$

$$\text{J-bit Encoding} = \frac{5744}{6496} = 0.884236453$$

$$\text{Deflate Compression} = \frac{5744}{2295} = 2.502832244$$

2) *Compression Ratio*

$$\text{Run-Length Encoding} = \frac{11296}{5744} * 100\% = 197\%$$

$$\text{Huffman Coding} = \frac{3022}{5744} * 100\% = 53\%$$

$$\text{Arithmetic Coding} = \frac{3003}{5744} * 100\% = 52\%$$

$$\text{Lempel-Ziv-Welch} = \frac{4884}{5744} * 100\% = 85\%$$

$$\text{J-bit Encoding} = \frac{6496}{5744} * 100\% = 113\%$$

$$\text{Deflate Compression} = \frac{2295}{5744} * 100\% = 40\%$$

3) *Space Savings*

*Run-Length Encoding* =  $100\% - 197\% = -97\%$

*Huffman Coding* =  $100\% - 53\% = 47\%$

*Arithmetic Coding* =  $100\% - 52\% = 48\%$

*Lempel-Ziv-Welch* =  $100\% - 85\% = 15\%$

*J-bit Encoding* =  $100\% - 113\% = -13\%$

*Deflate Compression* =  $100\% - 40 = 60\%$

#### 4) Redundancy

*Run-Length Encoding* =  $100\% - 197\% = -97\%$

*Huffman Coding* =  $100\% - 53\% = 47\%$

*Arithmetic Coding* =  $100\% - 52\% = 48\%$

*Lempel-Ziv-Welch* =  $100\% - 85\% = 15\%$

*J-bit Encoding* =  $100\% - 113\% = -13\%$

*Deflate Compression* =  $100\% - 40 = 60\%$

Adapun penerapan dari metode perbandingan yaitu metode eksponensial dapat dilihat sebagai berikut:

$$TNi = \sum_{j=1}^m (Vij) \times Bj$$

Total Nilai *Run-Length Encoding* =  $\sum_{j=1}^m (Vij) \times Bj$

$$= ((V1 * Bcr) + (V2 * Bcr) + (V3 * Bcr) + (V4 * Bcr))$$

$$= ((0.508498584 * 0.508498584) + ((197\%) * 0.508498584))$$

$$+ ((-97\%) * 0.508498584)$$

$$+ ((-97\%) * 0.508498584))$$

$$= 0.275567977$$

Total Nilai *Huffman Coding* =  $\sum_{j=1}^m (Vij) \times Bj$

$$= ((V1 * Bcr) + (V2 * Bcr) + (V3 * Bcr) + (V4 * Bcr))$$

$$= ((1.900727995 * 1.900727995) + ((53\%) * 1.900727995))$$

$$+ ((47\%) * 1.900727995) + (47\%) * 1.900727995))$$

$$= 6.414222899$$

Total Nilai *Arithmetic Coding* =  $\sum_{j=1}^m (Vij) \times Bj$

$$= ((V1 * Bcr) + (V2 * Bcr) + (V3 * Bcr) + (V4 * Bcr))$$

$$\begin{aligned}
&= ((1.912753913 * 1.912753913) + ((52\%) * 1.912753913) \\
&\quad + ((48\%) * 1.912753913) + (48\%) * 1.912753913)) \\
&= 6.484135356
\end{aligned}$$

$$\begin{aligned}
\text{Total Nilai } Lempel-Ziv-Welch &= \sum_{j=1}^m (V_{ij}) \times B_j \\
&= ((V1 * Bcr) + (V2 * Bcr) + (V3 * Bcr) + (V4 * Bcr)) \\
&= ((1.176085176 * 1.176085176) + ((85\%) * 1.176085176) \\
&\quad + ((15\%) * 1.176085176) + ((15\%) * 1.176085176)) \\
&= 2.735346694
\end{aligned}$$

$$\begin{aligned}
\text{Total Nilai } J\text{-bit Encoding} &= \sum_{j=1}^m (V_{ij}) \times B_j \\
&= ((V1 * Bcr) + (V2 * Bcr) + (V3 * Bcr) + (V4 * Bcr)) \\
&= ((0.884236453 * 0.884236453) + ((113\%) * 0.884236453) \\
&\quad + ((-13\%) * 0.884236453) \\
&\quad + ((-13\%) * 0.884236453)) \\
&= 1.550347012
\end{aligned}$$

$$\begin{aligned}
\text{Total Nilai } Deflate \text{ Compression} &= \sum_{j=1}^m (V_{ij}) \times B_j \\
&= ((V1 * Bcr) + (V2 * Bcr) + (V3 * Bcr) + (V4 * Bcr)) \\
&= ((2.502832244 * 2.502832244) + ((40\%) * 2.502832244) \\
&\quad + ((60\%) * 2.502832244) + ((60\%) * 2.502832244)) \\
&= 10.26983373
\end{aligned}$$

Setelah diperoleh nilai akhir atau total nilai dari masing-masing alternatif, maka tahapan selanjutnya adalah menentukan prioritas keputusan berdasarkan nilai dari masing–masing alternatif. Hasil prioritas keputusan dapat dilihat pada tabel dibawah ini:

Tabel 4.2 Nilai Eksponensial 1 Paragraf

Alternatif	Ratio Of Compression	Compression Ratio	Space Savings	Redundancy	Exponential	Rank
RLE	0.508498584	197%	-97%	-97%	0.275567977	6
Huffman	1.900727995	53%	47%	47%	6.414222899	3
Arithmetic	1.912753913	52%	48%	48%	6.484135356	2
LZW	1.176085176	85%	15%	15%	2.735346694	4
J-bit	0.884236453	113%	-13%	-13%	1.550347012	5
Deflate	2.502832244	40%	60%	60%	10.26983373	1

Sumber: diolah oleh peneliti

Setelah dilihat dari tabel di atas, maka dapat disimpulkan bahwa algoritma yang lebih baik untuk mengompresi teks satu paragraf adalah algoritma *Deflate Compression*. Hasil kompresi menggunakan algoritma *Deflate Compression* relatif lebih besar dibandingkan hasil kompresi menggunakan lima algoritma lainnya.

#### B. 5 Paragraf

Berdasarkan hasil yang sudah dipaparkan di bagian sebelumnya, berikut merupakan tabel yang memberikan ringkasan secara umum atas performa pada algoritma di variasi teks lima paragraf.

Tabel 4.3 Performa Algoritma Teks 5 Paragraf

Alternatif	Sebelum Kompresi	Sesudah Kompresi
RLE	32408	64080
Huffman	32408	18047
Arithmetic	32408	17898
LZW	32408	21612
J-bit	32408	36496
Deflate	32408	13694

Sumber: diolah oleh peneliti

Berdasarkan data di atas dapat dihitung parameter kinerja kompresinya yang digunakan seperti di bawah ini:

1) *Ratio Of Compression*

$$\text{Run-Length Encoding} = \frac{32408}{64080} = 0.505742821$$

$$\text{Huffman Coding} = \frac{32408}{18047} = 1.795755527$$

$$\text{Arithmetic Coding} = \frac{32408}{17898} = 1.810705107$$

$$\text{Lempel-Ziv-Welch} = \frac{32408}{21612} = 1.499537294$$

$$\text{J-bit Encoding} = \frac{32408}{36496} = 0.887987725$$

$$\text{Deflate Compression} = \frac{32408}{13694} = 2.366583905$$

2) *Compression Ratio*

$$\text{Run-Length Encoding} = \frac{64080}{32408} * 100\% = 198\%$$

$$\text{Huffman Coding} = \frac{18047}{32408} * 100\% = 56\%$$

$$\text{Arithmetic Coding} = \frac{17898}{32408} * 100\% = 55\%$$

$$\text{Lempel-Ziv-Welch} = \frac{21612}{32408} * 100\% = 67\%$$

$$\text{J-bit Encoding} = \frac{36496}{32408} * 100\% = 113\%$$

$$\text{Deflate Compression} = \frac{13694}{32408} * 100\% = 42\%$$

3) *Space Savings*

*Run-Length Encoding* =  $100\% - 198\% = -98\%$

*Huffman Coding* =  $100\% - 56\% = 44\%$

*Arithmetic Coding* =  $100\% - 55\% = 45\%$

*Lempel-Ziv-Welch* =  $100\% - 67\% = 33\%$

*J-bit Encoding* =  $100\% - 113 = -13\%$

*Deflate Compression* =  $100\% - 42\% = 58\%$

4) *Redundancy*

*Run-Length Encoding* =  $100\% - 198\% = -98\%$

*Huffman Coding* =  $100\% - 56\% = 44\%$

*Arithmetic Coding* =  $100\% - 55\% = 45\%$

*Lempel-Ziv-Welch* =  $100\% - 67\% = 33\%$

*J-bit Encoding* =  $100\% - 113 = -13\%$

*Deflate Compression* =  $100\% - 42\% = 58\%$

Adapun penerapan dari metode perbandingan yaitu metode eksponensial dapat dilihat sebagai berikut:

$$TNi = \sum_{j=1}^m (Vij) \times Bj$$

Total Nilai *Run-Length Encoding* =  $\sum_{j=1}^m (Vij) \times Bj$

$$= ((V1 * Bcr) + (V2 * Bcr) + (V3 * Bcr) + (V4 * Bcr))$$

$$= ((0.505742821 * 0.505742821) + ((198\%) * 0.505742821)$$

$$+ ((-98\%) * 0.505742821)$$

$$+ ((-98\%) * 0.505742821))$$

$$= 0.267261444$$

Total Nilai *Huffman Coding* =  $\sum_{j=1}^m (Vij) \times Bj$

$$= ((V1 * Bcr) + (V2 * Bcr) + (V3 * Bcr) + (V4 * Bcr))$$

$$= ((1.795755527 * 1.795755527) + ((56\%) * 1.795755527)$$

$$+ ((44\%) * 1.795755527) + (44\%) * 1.795755527))$$

$$= 5.816248968$$

Total Nilai *Arithmetic Coding* =  $\sum_{j=1}^m (Vij) \times Bj$

$$\begin{aligned}
&= ((V1 * Bcr) + (V2 * Bcr) + (V3 * Bcr) + (V4 * Bcr)) \\
&= ((1.810705107 * 1.810705107) + ((55\%) * 1.810705107) \\
&\quad + ((45\%) * 1.810705107) + (45\%) * 1.810705107)) \\
&= 5.900063197
\end{aligned}$$

$$\begin{aligned}
\text{Total Nilai } Lempel-Ziv-Welch &= \sum_{j=1}^m (Vij) \times Bj \\
&= ((V1 * Bcr) + (V2 * Bcr) + (V3 * Bcr) + (V4 * Bcr)) \\
&= ((1.499537294 * 1.499537294) + ((67\%) * 1.499537294) \\
&\quad + ((33\%) * 1.499537294) + ((33\%) * 1.499537294)) \\
&= 4.247686685
\end{aligned}$$

$$\begin{aligned}
\text{Total Nilai } J\text{-bit Encoding} &= \sum_{j=1}^m (Vij) \times Bj \\
&= ((V1 * Bcr) + (V2 * Bcr) + (V3 * Bcr) + (V4 * Bcr)) \\
&= ((0.887987725 * 0.887987725) + ((113\%) * 0.887987725) \\
&\quad + ((-13\%) * 0.887987725) \\
&\quad + ((-13\%) * 0.887987725)) \\
&= 1.564497649
\end{aligned}$$

$$\begin{aligned}
\text{Total Nilai } Deflate \text{ Compression} &= \sum_{j=1}^m (Vij) \times Bj \\
&= ((V1 * Bcr) + (V2 * Bcr) + (V3 * Bcr) + (V4 * Bcr)) \\
&= ((2.366583905 * 2.366583905) + ((42\%) * 2.366583905) \\
&\quad + ((58\%) * 2.366583905) + ((58\%) * 2.366583905)) \\
&= 9.333887192
\end{aligned}$$

Setelah diperoleh nilai akhir atau total nilai dari masing – masing alternatif, maka tahapan selanjutnya adalah menentukan prioritas keputusan berdasarkan nilai dari masing – masing alternatif. Hasil prioritas keputusan dapat dilihat pada tabel dibawah ini:

Tabel 4.4 Nilai Eksponensial 5 Paragraf

Alternatif	Ratio Of Compression	Compression Ratio	Space Savings	Redundancy	Exponential	Rank
RLE	0.505742821	198%	-98%	-98%	0.267261	6
Huffman	1.795755527	56%	44%	44%	5.816249	3
Arithmetic	1.810705107	55%	45%	45%	5.900063	2
LZW	1.499537294	67%	33%	33%	4.247687	4
J-bit	0.887987725	113%	-13%	-13%	1.564498	5
Deflate	2.366583905	42%	58%	58%	9.333887	1

Sumber: diolah oleh peneliti

Setelah dilihat dari tabel diatas, maka dapat disimpulkan bahwa algoritma yang lebih baik untuk mengkompresi teks satu paragraf adalah algoritma *Deflate Compression*. Hasil kompresi menggunakan algoritma *Deflate Compression* relatif lebih besar dibandingkan hasil kompresi menggunakan lima algoritma lainnya.

### C. 10 Paragraf

Berdasarkan hasil yang sudah dipaparkan di bagian sebelumnya, berikut merupakan tabel yang memberikan ringkasan secara umum atas performa pada algoritma di variasi teks lima paragraf.

Tabel 4.5 Performa Algoritma Teks 10 Paragraf

Alternatif	Sebelum Kompresi	Sesudah Kompresi
RLE	55240	108912
Huffman	55240	30709
Arithmetic	55240	30448
LZW	55240	33240
J-bit	55240	62184
Deflate	55240	23810

Sumber: diolah oleh peneliti

Berdasarkan data di atas dapat dihitung parameter kinerja kompresinya yang digunakan seperti di bawah ini:

1) *Ratio Of Compression*

$$\text{Run-Length Encoding} = \frac{55240}{108912} = 0.507198472$$

$$\text{Huffman Coding} = \frac{55240}{30709} = 1.798821192$$

$$\text{Arithmetic Coding} = \frac{55240}{30448} = 1.814240673$$

$$\text{Lempel-Ziv-Welch} = \frac{55240}{33240} = 1.661853189$$

$$\text{J-bit Encoding} = \frac{55240}{62184} = 0.888331404$$

$$\text{Deflate Compression} = \frac{55240}{23810} = 2.320033599$$

2) *Compression Ratio*

$$\text{Run-Length Encoding} = \frac{108912}{55240} * 100\% = 197\%$$

$$\text{Huffman Coding} = \frac{30709}{55240} * 100\% = 56\%$$

$$\text{Arithmetic Coding} = \frac{30448}{55240} * 100\% = 55\%$$

$$\text{Lempel-Ziv-Welch} = \frac{33240}{55240} * 100\% = 60\%$$

$$\text{J-bit Encoding} = \frac{62184}{55240} * 100\% = 113\%$$

$$\text{Deflate Compression} = \frac{23810}{55240} * 100\% = 43\%$$

3) *Space Savings*

*Run-Length Encoding* =  $100\% - 197\% = -97\%$

*Huffman Coding* =  $100\% - 56\% = 44\%$

*Arithmetic Coding* =  $100\% - 55\% = 45\%$

*Lempel-Ziv-Welch* =  $100\% - 60\% = 40\%$

*J-bit Encoding* =  $100\% - 113 = -13\%$

*Deflate Compression* =  $100\% - 43\% = 57\%$

4) *Redundancy*

*Run-Length Encoding* =  $100\% - 197\% = -97\%$

*Huffman Coding* =  $100\% - 56\% = 44\%$

*Arithmetic Coding* =  $100\% - 55\% = 45\%$

*Lempel-Ziv-Welch* =  $100\% - 60\% = 40\%$

*J-bit Encoding* =  $100\% - 113 = -13\%$

*Deflate Compression* =  $100\% - 43\% = 57\%$

Adapun penerapan dari metode perbandingan yaitu metode eksponensial dapat dilihat sebagai berikut:

$$TNi = \sum_{j=1}^m (Vij) \times Bj$$

Total Nilai *Run-Length Encoding* =  $\sum_{j=1}^m (Vij) \times Bj$

$$= ((V1 * Bcr) + (V2 * Bcr) + (V3 * Bcr) + (V4 * Bcr))$$

$$= ((0.507198472 * 0.507198472) + ((197\%) * 0.507198472)$$

$$+ ((-97\%) * 0.507198472)$$

$$+ ((-97\%) * 0.507198472))$$

$$= 0.271647234$$

Total Nilai *Huffman Coding* =  $\sum_{j=1}^m (Vij) \times Bj$

$$= ((V1 * Bcr) + (V2 * Bcr) + (V3 * Bcr) + (V4 * Bcr))$$

$$= ((1.798821192 * 1.798821192) + ((56\%) * 1.798821192)$$

$$+ ((44\%) * 1.798821192) + (44\%) * 1.798821192))$$

$$= 5.833400067$$

Total Nilai *Arithmetic Coding* =  $\sum_{j=1}^m (Vij) \times Bj$

$$\begin{aligned}
&= ((V1 * Bcr) + (V2 * Bcr) + (V3 * Bcr) + (V4 * Bcr)) \\
&= ((1.814240673 * 1.814240673) + ((55\%) * 1.814240673) \\
&\quad + ((45\%) * 1.814240673) + (45\%) * 1.814240673)) \\
&= 6.4246990087183775
\end{aligned}$$

$$\begin{aligned}
\text{Total Nilai } Lempel-Ziv-Welch &= \sum_{j=1}^m (Vij) \times Bj \\
&= ((V1 * Bcr) + (V2 * Bcr) + (V3 * Bcr) + (V4 * Bcr)) \\
&= ((1.661853189 * 1.661853189) + ((60\%) * 1.661853189) \\
&\quad + ((40\%) * 1.661853189) + ((40\%) * 1.661853189)) \\
&= 3.5848932033818244
\end{aligned}$$

$$\begin{aligned}
\text{Total Nilai } J\text{-bit Encoding} &= \sum_{j=1}^m (Vij) \times Bj \\
&= ((V1 * Bcr) + (V2 * Bcr) + (V3 * Bcr) + (V4 * Bcr)) \\
&= ((0.888331404 * 0.888331404) + ((113\%) * 0.888331404) \\
&\quad + ((-13\%) * 0.888331404) \\
&\quad + ((-13\%) * 0.888331404)) \\
&= 1.5604731790054343
\end{aligned}$$

$$\begin{aligned}
\text{Total Nilai } Deflate \text{ Compression} &= \sum_{j=1}^m (Vij) \times Bj \\
&= ((V1 * Bcr) + (V2 * Bcr) + (V3 * Bcr) + (V4 * Bcr)) \\
&= ((2.320033599 * 2.320033599) + ((43\%) * 2.320033599) \\
&\quad + ((57\%) * 2.320033599) + ((57\%) * 2.320033599)) \\
&= 9.022623101
\end{aligned}$$

Setelah diperoleh nilai akhir atau total nilai dari masing – masing alternatif, maka tahapan selanjutnya adalah menentukan prioritas keputusan berdasarkan nilai dari masing – masing alternatif. Hasil prioritas keputusan dapat dilihat pada tabel dibawah ini:

Tabel 4.6 Nilai Eksponensial 10 Paragraf

Alternatif	Ratio Of Compression	Compression Ratio	Space Savings	Redundancy	Exponential	Rank
RLE	0.507198472	197%	-97%	-97%	0.271647234	6
Huffman	1.798821192	56%	44%	44%	5.833400067	3
Arithmetic	1.814240673	55%	45%	45%	6.4246990087183775	2
LZW	1.53256705	60%	40%	40%	3.5848932033818244	4
J-bit	0.888331404	113%	-13%	-13%	1.5604731790054343	5
Deflate	2.320033599	43%	57%	57%	9.022623101	1

Sumber: diolah oleh peneliti

Setelah dilihat dari tabel diatas, maka dapat disimpulkan bahwa algoritma yang lebih baik untuk mengkompresi teks satu paragraf adalah algoritma *Deflate Compression*. Hasil kompresi menggunakan algoritma *Deflate Compression* relatif lebih besar dibandingkan hasil kompresi menggunakan lima algoritma lainnya.

Berdasarkan tiga variasi teks yang sudah diujikan, dapat diambil hasil bahwa algoritma yang memiliki hasil kompresi terbaik dari enam algoritma yang sudah diujikan adalah algoritma *Deflate Compression*. Baik dinilai dari variasi pertama, kedua atau pun ketiga semuanya menunjukkan hasil terbaik. Sebaliknya algoritma dengan hasil kompresi terburuk adalah *Run-Length Encoding*. Hal ini berbeda dengan yang Hardi et al., (2019) temukan bahwa Run-Length Encoding mendapatkan *compression ratio* sebesar 30.645%. Hal ini mungkin terjadi karena tidak banyaknya karakter sama yang berderet pada teks sehingga algoritma ini mendapat performa terburuk.

## 4.6 Hasil Analisis Waktu

### A. 1 Paragraf

Berdasarkan hasil yang sudah dipaparkan di bagian sebelumnya, berikut merupakan tabel yang memberikan ringkasan secara umum atas performa pada algoritma di variasi teks satu paragraf atas proses *encoding*. Setelah dilihat dari tabel 4.7, maka dapat dilihat bahwa algoritma yang melakukan pekerjaan paling cepat untuk mengkompresi teks satu paragraf adalah algoritma *Lempel-Ziv-Welch*. Serta yang memiliki hasil kompresi paling lambat adalah algoritma *Arithmetic Coding* relatif lebih besar dibandingkan hasil kompresi menggunakan lima algoritma lainnya.

Tabel 4.7 Performa Waktu *Encoding* 1 Paragraf

Alternatif	Percobaan pertama(ms)	Percobaan kedua(ms)	Percobaan ketiga(ms)	Rata-rata(ms)
RLE	0.71	1.419	0.367	0.832
Huffman	0.868	0.388	0.354	0.536667
Arithmetic	7.596	4.995	6.419	6.336667
LZW	0.548	0.531	0.363	0.480667
J-bit	0.312	0.236	1.181	0.576333
Deflate	7.075	1.279	6.346	4.9

Sumber: diolah oleh peneliti

Kemudian berikut merupakan tabel yang memberikan ringkasan secara umum atas performa pada algoritma di variasi teks satu paragraf atas proses *decoding*. Setelah dilihat dari tabel 4.8, maka dapat dilihat bahwa algoritma yang melakukan pekerjaan paling cepat untuk mengkompresi teks satu paragraf adalah algoritma *Run-Length Encoding*. Serta yang memiliki hasil kompresi paling lambat adalah algoritma *Arithmetic Coding* relatif lebih besar dibandingkan hasil kompresi menggunakan lima algoritma lainnya.

Tabel 4. 8 Performa Waktu *Decoding* 1 Paragraf

Alternatif	Percobaan pertama(ms)	Percobaan kedua(ms)	Percobaan ketiga(ms)	Rata-rata(ms)
RLE	0.645	0.562	0.332	0.513
Huffman	1.459	0.439	0.904	0.934
Arithmetic	28.655	27.173	27.271	27.69967
LZW	0.743	2.075	0.217	1.011667
J-bit	1.035	0.272	0.392	0.566333
Deflate	1.828	1.282	0.964	1.358

Sumber: diolah oleh peneliti

### B. 5 Paragraf

Berdasarkan hasil yang sudah dipaparkan di bagian sebelumnya, berikut merupakan tabel yang memberikan ringkasan secara umum atas performa pada algoritma di variasi teks satu paragraf atas proses *encoding*. Setelah dilihat dari tabel 4.9, maka dapat dilihat bahwa algoritma yang melakukan pekerjaan paling cepat untuk mengkompresi teks satu paragraf adalah algoritma *J-bit Encoding*. Serta yang memiliki hasil kompresi paling lambat adalah algoritma *Arithmetic Coding* relatif lebih besar dibandingkan hasil kompresi menggunakan lima algoritma lainnya.

Tabel 4.9 Performa Waktu *Encoding* 5 Paragraf

Alternatif	Percobaan pertama(ms)	Percobaan kedua(ms)	Percobaan ketiga(ms)	Rata-rata(ms)
RLE	4.349	3.11	2.591	3.35
Huffman	1.238	1.328	1.735	1.433667
Arithmetic	119.091	90.257	68.277	92.54167
LZW	3.268	2.802	1.748	2.606
J-bit	1.553	0.739	1.151	1.147667
Deflate	22.185	34.696	22.136	26.339

Sumber: diolah oleh peneliti

Kemudian berikut merupakan tabel yang memberikan ringkasan secara umum atas performa pada algoritma di variasi teks satu paragraf atas proses *decoding*. Setelah dilihat dari tabel 4.10, maka dapat dilihat bahwa algoritma yang melakukan pekerjaan paling cepat untuk mengkompresi teks satu paragraf adalah algoritma *Lempel-Ziv-Welch*. Serta yang memiliki hasil kompresi paling lambat adalah algoritma *Arithmetic Coding* relatif lebih besar dibandingkan hasil kompresi menggunakan lima algoritma lainnya.

Tabel 4.10 Performa Waktu *Decoding* 5 Paragraf

Alternatif	Percobaan pertama(ms)	Percobaan kedua(ms)	Percobaan ketiga(ms)	Rata-rata(ms)
RLE	1.635	2.427	0.969	1.677
Huffman	3.545	3.029	3.834	3.469333
Arithmetic	2711	1941	1990	2214
LZW	1.086	1.852	1.021	1.319667
J-bit	2.895	2.097	3.224	2.738667
Deflate	18.711	3.931	5.528	9.39

Sumber: diolah oleh peneliti

### C. 10 Paragraf

Berdasarkan hasil yang sudah dipaparkan di bagian sebelumnya, berikut merupakan tabel yang memberikan ringkasan secara umum atas performa pada algoritma di variasi teks satu paragraf atas proses *encoding*. Setelah dilihat dari tabel ini, maka dapat dilihat bahwa algoritma yang melakukan pekerjaan paling cepat untuk mengkompresi teks satu paragraf adalah algoritma *J-bit Encoding*. Serta yang memiliki hasil kompresi paling lambat adalah algoritma *Arithmetic Coding* relatif lebih besar dibandingkan hasil kompresi menggunakan lima algoritma lainnya.

Tabel 4.11 Performa Waktu *Encoding* 10 Paragraf

Alternatif	Percobaan pertama(ms)	Percobaan kedua(ms)	Percobaan ketiga(ms)	Rata-rata(ms)
RLE	4.852	4.651	2.823	4.108667
Huffman	3.614	1.861	1.412	2.295667
Arithmetic	357.16	221.61	227.169	268.6463
LZW	4.177	2.161	2.327	2.888333
J-bit	13.782	2.079	1.385	5.748667
Deflate	50.431	66.4439	75.273	64.0493

Sumber: diolah oleh peneliti

Kemudian berikut merupakan tabel yang memberikan ringkasan secara umum atas performa pada algoritma di variasi teks satu paragraf atas proses *decoding*. Setelah dilihat dari tabel 4.12, maka dapat dilihat bahwa algoritma yang melakukan pekerjaan paling cepat untuk mengkompresi teks satu paragraf adalah algoritma *Lempel-Ziv-Welch*. Serta yang memiliki hasil kompresi paling lambat adalah algoritma *Arithmetic Coding* relatif lebih besar dibandingkan hasil kompresi menggunakan lima algoritma lainnya.

Tabel 4.12 Performa Waktu *Decoding* 10 Paragraf

Alternatif	Percobaan pertama(ms)	Percobaan kedua(ms)	Percobaan ketiga(ms)	Rata-rata(ms)
RLE	2.873	2.808	1.955	2.545333
Huffman	7.113	3.425	4.283	4.940333
Arithmetic	8316	7549	7786	7883.667
LZW	1.244	1.771	1.592	1.535667
J-bit	2.699	1.939	1.438	2.025333
Deflate	13.032	12.153	8.966	11.38367

Sumber: diolah oleh peneliti

Setelah didapatkan semua hasil simulasi tiap skenario yang sudah dijalankan dan didapatkan rata-rata hasil *runtime* bahwa hasil pekerjaan dari setiap algoritma adalah relatif sama. Hal ini sesuai dengan nilai kompleksitas waktu yang dimiliki oleh tiap-tiap algoritma yaitu  $O(n)$ . Namun, terdapat anomali nilai pada algoritma *Arithmetic* yang mana hal ini sangat menyimpang dari nilai kompleksitas yang dimilikinya. Secara teori seharusnya algoritma ini hanya mengalami peningkatan waktu secara linear berbanding lurus terhadap panjang teks masukan. Hal yang mungkin terjadi karena di dalam setiap iterasi yang dimiliki oleh algortima ini memanfaatkan pangkat dalam perhitungan prosesnya. Sehingga meskipun fungsi di dalam algoritma yang terjadi memang bernilai linear, hasil yang dihasilkan menunjukkan jumlah peningkatan waktu penggeraan secara eksponensial.

## **BAB V**

### **HASIL PENELITIAN DAN PEMBAHASAN**

#### **5.1. Kesimpulan**

Pada penelitian ini, didapatkan kesimpulan bahwa pengembangan aplikasi steganografi berbasis web yang di dalamnya disertai dengan kompresi bit dapat meningkatkan keamanan distribusi informasi terutama terkait dengan informasi yang bersifat rahasia. Pemanfaatan kompresi bit dalam implementasi steganografi menggunakan metode Least Significant Bit, membuat pemanfaatan transparansi persepsi, robustness, tetap terjaga dikarenakan memang kelebihan dari metode steganografi ini serta dapat meningkatkan kapasitas penyamaran data masukan hingga 2.5 kali sehingga dapat menghemat ruang hingga 60% pada medium dengan menggunakan algoritma *Deflate Coding* dengan menambah waktu yang relatif singkat pada proses steganografi.

Keberhasilan fungsi terhadap penggunaan teknik steganografi yang dikembangkan dapat dilihat dari *stegoimage* yang dapat diunduh dalam aplikasi web sederhana. Hal ini menunjukkan bahwa penambahan data teks ke dalam medium citra digital tidak merusak persepsi berkas tersebut. Dengan demikian, citra digital dapat tetap digunakan sebagai medium dalam distribusi informasi rahasia.

Keberhasilan integrasi antara fungsi kompresi bit dan fungsi steganografi dalam sebuah aplikasi sederhana dapat dilihat dari proses memasukan data dilanjutkan dengan memilih metode kompresi untuk disisipkan ke dalam medium citra digital dalam aplikasi web sederhana. Setelah citra *stegoimage* yang berhasil diunduh dan diunggah kembali untuk diekstrak, pesan rahasia kembali menjadi pesan awal yang dimasukan. Hal ini menunjukkan bahwa kompresi data teks ke dalam medium citra digital tidak merusak data yang disisipkan ke dalam berkas digital serta tetap mempertahankan persepsi berkas tersebut. Dengan demikian, citra digital dapat tetap digunakan sebagai medium dalam

distribusi informasi rahasia dengan tambahan jumlah kapasitas penyimpanan dan peningkatan keamanan terhadap data yang disisipkan.

### **5.2. Saran**

Berdasarkan kesimpulan dari penelitian yang sudah dijabarkan, terdapat beberapa saran untuk penelitian selanjutnya, yaitu saran teoritis dan saran implementatif. Saran teoritis yang dapat diberikan adalah penggunaan algoritma yang menggunakan metode kunci, pencarian pola, atau pohon kode. Metode-metode tersebut sangat cocok untuk kompresi teks berdasarkan hasil penelitian ini. Disarankan untuk mengkombinasikan beberapa teknik kompresi berbeda, seperti Deflate Coding, karena terbukti memberikan hasil yang sangat baik dalam kompresi bit teks serta menggunakan waktu kompresi yang relatif rendah. Saran implementatif terhadap penelitian ini adalah bahwa implementasi kompresi bit dalam aplikasi web dapat membantu meningkatkan protokol komunikasi yang mengandung informasi rahasia, sehingga dapat memperkaya dan memperkuat strategi persebaran informasi, khususnya yang berkaitan dengan distribusi teks rahasia.

## DAFTAR PUSTAKA

- Agrawal, H., Andrades, D., & Ringe, S. (2015). Approaches to Optimize Bit Compression Algorithm. *International Journal of Engineering Research And*, V4(03), 903–906.  
<https://doi.org/10.17577/ijertv4is030800>
- Al Maki, W. F., Muktyas, I. B., Arifin, S., Suwarno, & Aziz, M. K. B. M. (2023). Implementation of a Logistic Map to Calculate the Bits Required for Digital Image Steganography Using the Least Significant Bit (LSB) Method. *Journal of Computer Science*, 19(6), 686–693.  
<https://doi.org/10.3844/jcssp.2023.686.693>
- Albayati, H. A. W. J., & Ali, S. A. (2021). A Comparative Study of Image Steganography Based on Edge Detection A Comparative Study of Image Steganography Based on Edge Detection. *Journal of Physics: Conference Series*. <https://doi.org/10.1088/1742-6596/1818/1/012032>
- Aruna, M. T. N., Nandika, L. R., Sneha, C. I., Xavier, imothy J., & George, T. M. (2023). Text, Image and Audio Steganography. *International Journal for Research in Applied Science and Engineering Technology*, 11(4), 4435–4439.  
<https://doi.org/10.22214/ijraset.2023.51091>
- Baldha, N. (2023). A Web-based Least Significant Bit (LSB) Image Steganographic Technique. *Science Journal of Circuits, Systems and Signal Processing*, 11(1), 12–18.  
<https://doi.org/10.11648/j.cssp.20231101.12>
- Bille, P., Gørtz, I. L., Puglisi, S. J., & Tarnow, S. R. (2023). Hierarchical Relative Lempel-Ziv Compression. *Leibniz International Proceedings in Informatics, LIPIcs*, 265.  
<https://doi.org/10.4230/LIPIcs.SEA.2023.18>
- Cheddad, A., Condell, J., Curran, K., & Mc Kevitt, P. (2008). Enhancing steganography in digital images. *Proceedings of the 5th Canadian Conference on Computer and Robot Vision, CRV 2008*, 326–332.

- <https://doi.org/10.1109/CRV.2008.54>
- Cyber defense Definition & Meaning - Merriam-Webster.* (n.d.), from <https://www.merriam-webster.com/dictionary/cyber defense> (diakses pada 18 Januari 2024)
- Denning, D. E., & Strawser, B. J. (2012). Active Cyber Defense - Cyber Analogies. *CSS Cyberdefense Trend Analyses* 1.
- Dey, S. (2022). *An Efficient Data Compression Algorithm*.  
<https://doi.org/https://doi.org/10.21203/rs.3.rs-2082171/v1>
- Djebbar, F., Ayad, B., Meraim, K. A., & Hamam, H. (2012). Comparative study of digital audio steganography techniques. *Eurasip Journal on Audio, Speech, and Music Processing*, 2012(1), 1–16.  
<https://doi.org/10.1186/1687-4722-2012-25>
- Dutta, P. S., & Chakraborty, S. (2020). Image based Steganography in Cryptography implementing different Encryption-Decryption Algorithm. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 6(3), 745–748.  
<https://doi.org/10.32628/cseit2063191>
- Feng, S., Cao, J., Luo, Y., Dai, Z., Zhang, Y., & Wang, Y. (2020). Compression for Text Detection and Recognition Based on Low Bit-Width Quantization. *2020 IEEE 5th International Conference on Signal and Image Processing, ICSIP 2020*, 313–317.  
<https://doi.org/10.1109/ICSIP49896.2020.9339320>
- Galinec, D. (2023). Cyber Security and Cyber Defense: Challenges and Building of Cyber Resilience Conceptual Model. *International Journal of Applied Sciences & Development*, 1, 83–88.  
<https://doi.org/10.37394/232029.2022.1.10>
- Gupta, A., Bansal, A., & Khanduja, V. (2017). *Modern Lossless Compression Techniques: Review, Comparison and Analysis*.
- Hardi, S. M., Angga, B., Lydia, M. S., Jaya, I., & Tarigan, J. T. (2019a). Comparative Analysis Run-Length Encoding Algorithm and Fibonacci Code Algorithm on Image Compression. *Journal of Physics:*

- Conference Series*, 1235(1). <https://doi.org/10.1088/1742-6596/1235/1/012107>
- Hardi, S. M., Angga, B., Lydia, M. S., Jaya, I., & Tarigan, J. T. (2019b). Comparative Analysis Run-Length Encoding Algorithm and Fibonacci Code Algorithm on Image Compression. *Journal of Physics: Conference Series*, 1235(1). <https://doi.org/10.1088/1742-6596/1235/1/012107>
- Hasibuan, N. A. (2022). *Analisa Perbandingan Algoritma Huffman Dengan Rice Code Dalam Kompresi File Video*. 6(November), 159–166. <https://doi.org/10.30865/komik.v6i1.5751>
- Hassan, M. U., Rehmani, M. H., & Chen, J. (2023). *Huff-DP: Huffman Coding based Differential Privacy Mechanism for Real-Time Data*. <http://arxiv.org/abs/2301.10395>
- Imanda, R., Nasution, H., Fauzi, A., & Khair, H. (2023). Hybrid Cryptosystem Algorithm Vigenere Cipher and Base64 for Text Message Security Utilizing Least Significant Bit ( LSB ) Steganography as Insert into Image. *Journal of Artificial Intelligence and Engineering Applications*, 2(3), 89–98.
- Jenifer, J. M. (2021). An Analysis on Video Steganography Techniques. *Journal of Emerging Technologies and Innovative Research (JETIR)*, 8(11), 79–84.
- Kehar, N., & Kaur, J. (2011). A Smart Technique : Stegnography. *International Journal of Computer Science Trends and Technology*, 4333(1), 211–215.
- Kose, J., Chia, O. B., & Baboolal, V. (2020). *Review and test of steganography techniques*. <https://arxiv.org/abs/2012.08460>
- Lee, D. (2021). *Steganography of Complex Networks*. <http://arxiv.org/abs/2110.10418>
- Li, G., Li, S., Li, M., Zhang, X., & Qian, Z. (2023). Steganography of Steganographic Networks. *Proceedings of the 37th AAAI Conference on Artificial Intelligence, AAAI 2023*, 37, 5178–5186.

- <https://doi.org/10.1609/aaai.v37i4.25647>
- Mados, B., Bilanová, Z., & Hurtuk, J. (2021). RLE: Lossless data compression algorithm using delta transformation and optimized bit-level run-length encoding. *Journal of Information and Organizational Sciences*, 45(1), 329–349.
- Milosav, P., Milosavljević, M., & Banjac, Z. (2023). Steganographic Method in Selected Areas of the Stego-Carrier in the Spatial Domain. *Symmetry*, 15(5). <https://doi.org/10.3390/sym15051015>
- Modupe, A. O., Adedoyin, A. E., Titilayo, A. O., & Deborah, F. O. (2021). A Comparative Analysis of LSB , MSB and PVD Based Image Steganography. *International Journal of Research and Review*, 8(9), 373–377.
- Mohey, G., Zekry, A., & Zakaria, H. (2021). Fpga implementation of lempel-ziv data compression. *International Journal of Reconfigurable and Embedded Systems*, 10(2), 99–108.  
<https://doi.org/10.11591/ijres.v10.i2.pp99-108>
- Naibaho, F. E. (2020). Implementasi Algoritma J-Bit Encoding Pada Kompresi File PDF. *Jurnal Sistem Komputer Dan Informatika (JSON)*, 1(3), 278–283. <https://doi.org/10.30865/json.v1i3.2153>
- Parkhomenko-Kutsevil, O. I. (2022). Problems of Information Security During Military Operations and Hostilities. *Public Management and Administration in Ukraine*, 28, 177–181.  
<https://doi.org/10.32843/pma2663-5240-2022.28.35>
- Reyzin, L., & Russell, S. (2004). Simple Stateless Steganography. *Proceedings of the 10th International Conference on Theory and Application of Cryptology and Information Security (ASIACRYPT 2004)*, 3329, 355–372. <http://dblp.uni-trier.de/db/journals/iacr/iacr2003.html#ReyzinR03>
- Shanmugasundaram, S., & Lourdusamy, R. (2011). Text Compression Algorithms - a Comparative Study. *ICTACT Journal on Communication Technology*, 02(04), 444–451.

- <https://doi.org/10.21917/ijct.2011.0062>
- Sihotang, D. (2017). Perancangan Aplikasi Keamanan Data Text Dengan Metode Idea Dan Kompresi Menggunakan Algoritma Huffman. *Majalah Ilmiah INTI*, 12(1), 14–19.
- Silitonga, P. D. P., & Morina, I. S. (2019). Compression and Decompression of Audio Files Using the Arithmetic Coding Method. *Scientific Journal of Informatics*, 6(1), 73–81.  
<https://doi.org/10.15294/sji.v6i1.17839>
- Tian, J., Rivera, C., Di, S., Chen, J., Liang, X., Tao, D., & Cappello, F. (2021). Revisiting huffman coding: Toward extreme performance on modern GPU architectures. *Proceedings - 2021 IEEE 35th International Parallel and Distributed Processing Symposium, IPDPS 2021, May*, 881–891.  
<https://doi.org/10.1109/IPDPS49936.2021.00097>
- Utama, F. (2023). BSSN Ungkap Ada 376 Dugaan Kebocoran Data Vital Selama 2022-2023 : Okezone Nasional. *Okezone*.  
<https://nasional.okezone.com/read/2023/08/22/337/2869210/bssn-ungkap-ada-376-dugaan-kebocoran-data-vital-selama-2022-2023> (diakses pada 5 Januari 2024)
- Yanti, F., & Budayawan, K. (2023). Implementasi Steganografi Menggunakan Metode Least Significant Bit (LSB) dalam Pengamanan Informasi pada Citra Digital. *Voteteknika (Vocational Teknik Elektronika Dan Informatika)*, 11(1), 63.  
<https://doi.org/10.24036/voteteknika.v11i1.121968>

## LAMPIRAN

### LAMPIRAN 1

#### BUKTI BIMBINGAN DOSEN PEMBIMBING 1

BIMBINGAN KONSULTASI  
USULAN PENELITIAN SKRIPSI  
PEMBIMBING 1

No.	Tanggal	Konsultasi (Saran / Perbaikan)	Ttd
1	4/1/2023	<ul style="list-style-type: none"><li>- Cari 10 referensi jurnal yang relevan dengan topik</li><li>- Cari 3 algoritma selain 5-bit dan Lzw</li><li>- mencoba 2 Code Huffman dan normal</li><li>- Coba cari referensi dari phd 1TB Rinal munir</li></ul>	
2	14/1/2023	<ul style="list-style-type: none"><li>- Parameter yang digunakan, Timerspasial bit</li><li>- gunakan referensi Rinal munir</li><li>- Cari dan buat Lzw, 5-bit, nanti boleh pakai Chat Gpt dengan catatan ada referensi sebagai penjelasan</li><li>- Fokuskan algoritma yang digunakan sebagai landasan dalam pembuatan judul</li><li>- judul disertakan dengan rujukan</li></ul>	
3	18/1/2023	<ul style="list-style-type: none"><li>- Algoritma bisa berubah sesuai komentar pengaji</li><li>- ketika dijalankan baru bisa diketahui hasilnya atau tidak</li><li>- parameter bit untuk mengetahui block bit yang digunakan (dalam grafik) misal space presentase dan bit yang digunakan</li><li>- Penelitian sebelumnya bisa berbeda algoritma tetapi media sama atau media berbeda tetapi algoritmanya</li></ul>	

No.	Tanggal	Konsultasi (Saran / Perbaikan)
4	18-1-2024	<ul style="list-style-type: none"> <li>- Dikaribut (lansasatu)</li> <li>- isi PPT berdasarkan sub judul dan hadir</li> </ul>
5	18-1-2024	<ul style="list-style-type: none"> <li>- bagian yang diperbaiki sebelumnya secara konten hanya diperbaiki atas komentar sebelumnya</li> <li>- class diagram diperbaiki</li> <li>- tulang kasi protode (Indris) Anom</li> </ul>
6	29-1-2024	<ul style="list-style-type: none"> <li>- pembuatan bagian yang sudah di tandai</li> </ul>
7	6-2-2024	<ul style="list-style-type: none"> <li>- tidak ada apa segera anom</li> </ul>
8	28-3-2024	<ul style="list-style-type: none"> <li>- tulis spesifikasi gambar (rgb, source, resolusi)</li> <li>- tahapan Steganografi seharusnya algoritma</li> <li>- pada bagian hasil diperbaiki</li> </ul>
9	30-6-2024	<ul style="list-style-type: none"> <li>- setiap gambar diberi pengelaran</li> <li>- tambah data masuk apakah juga</li> <li>- format atau senin nanti contoh revisinya akan diberikan</li> </ul>

No.	Tanggal	Konsultasi (Saran / Perbaikan)	Ttd
10	9-7-2014	<ul style="list-style-type: none"> <li>- Make sure dimasai algoritma</li> <li>- di PPT -nya tampil flowchart</li> <li>- flowchart di tampil di lembar 3</li> <li>- <sup>→</sup> 6 metode</li> <li>- <sup>Banyak</sup> "Tujuan"</li> <li>- Tujuan sederhana saja</li> <li>" bisa dipake di satuan"</li> <li>" memenuhi kebutuhan persandian disajikan"</li> <li>" Ciphertext + dikripsi/decripsi</li> <li>Ciphertext + desklripsi/decripsi</li> <li>d'introduction → tampil <sup>→</sup></li> <li>luvih kriptografi dan steganografi</li> <li>6 atau 7 terminologi</li> <li>Persandian digital → komprehensif</li> <li>- ilustrasi tulisan paku sembada</li> <li>- gambar dengan judul 1 con informasi</li> <li>Logo Informatica</li> <li>- Siap Salat [alasan]</li> <li>- attribute -nya diperbaiki,</li> <li>- Komprehensif</li> <li>- gambar dan dihapus</li> <li>- input 1 kata "Budi" pusatkan</li> <li>flowchart algoritma</li> <li>- Kuasai 6 algoritma</li> <li>- Penekanan terhadap siapa</li> </ul>	

**LAMPIRAN 2**  
**BUKTI BIMBINGAN DOSEN PEMBIMBING 2**



**BIMBINGAN KONSULTASI  
USULAN PENELITIAN SKRIPSI  
PEMBIMBING 2**

No.	Tanggal	Konsultasi (Saran / Perbaikan)
1	17-1-2024	<ul style="list-style-type: none"> <li>- latar belakang dikencangkan dan dimulai dari masalah dan kegiatan lalu berdampak penelitian, tidak fokus dan grafis sbgnya, siswa;</li> <li>- Sebaiknya identifikasi masalah point-point dari latar belakang</li> <li>- referensi dari identifikasi masalah ke latar belakang</li> <li>- gambar kerja berpikir minisaja</li> <li>- Use case buat UML, buat sesuai Standar</li> <li>- tambahkan public dan private di Class diagram</li> <li>- penulisannya rumus diperbaiki</li> <li>- faktor pemakaian perbaiki</li> </ul>
2	19-1-2024	<ul style="list-style-type: none"> <li>- tetap tambahkan SLOC di bagian pertama</li> <li>- Tambahkan bagian bukti dan tulisan</li> <li>- hapuskam bagian bab 3</li> </ul>

Buku Bimbingan dan Log Book Penelitian Skripsi FSTP Unhan RI

No.	Tanggal	Konsultasi (Saran / Perbaikan)	Ttd
3	30-1-2024	<ul style="list-style-type: none"> <li>- tolony didiskusikan terkait dengan metode komporasi dengan payiji</li> <li>- Tanyakan selanjutnya tentang konsentrasi dari payiji</li> </ul>	Ahmad
4	5-2-2024	<ul style="list-style-type: none"> <li>- Sebaiknya perbaikan ini melibatkan gambaran atas algoritma yang dibandingkan bukan mencari yang terbaik</li> <li>- Untuk metode kereta api yang punya bisa diperbaiki - perbaikan</li> </ul>	Ahmad
5	20-3-2024	<ul style="list-style-type: none"> <li>- Maka bisa sebelum Mei sudah siap aplikasinya</li> <li>- revisi 2 pada bulan Mei sudah selesai</li> </ul>	Ahmad
	16-4-2024	<ul style="list-style-type: none"> <li>- Cek draft isi [- indonesi] <ul style="list-style-type: none"> <li>- di (draf belakang) berikan keterangan</li> <li>- Pendekar nafas dan perhatian lagi</li> <li>- ke simpulan (draf belakang) perbaiki lagi</li> <li>- hitung dua (pendekar)</li> <li>- tambahkam kutipan sebelum arahsi</li> <li>- adiklat pembuktian proses PTK</li> <li>- bab 1 senggut kutipan</li> <li>- diperbaiki jugaan pendekar tersebut dilihat belakang</li> <li>- identifikasi kesalahan per pem - KAR</li> </ul> </li> </ul>	Ahmad

No.	Tanya	Jawaban	Ttd
6	26-6-2014	<p>pergunaan kata "kami" Coba cari log dan ganti menjadi <u>peneliti / penulis</u></p> <ul style="list-style-type: none"> <li>- tidak terlihat → tersandarkan</li> <li>- 3.7 Jadihan 1 sub bab saja.</li> <li>- pembahasan per proyek</li> <li>- Spesifikasi komputer dimis sebutnya pada bab 3</li> <li>- Coba tiaf emm</li> <li>- beri spas di bagian bawah</li> <li>- gambar dari bagian paragraf</li> <li>- di bagian penjelasan aplikasi kaw hananya menyajikan menggunakan aplikasi dan menjelaskan bentuk aplikasi yang akan terjadi</li> <li>- "fungsinya adalah untuk..."</li> <li>- dibuat point-point</li> <li>- gambar</li> <li>- point-point</li> <li>- bab 4 nanti bisa khusus dengan paku atau telur</li> </ul>	Ahmad

No.	Tanggal	Konsultasi (Saran / Perbaikan)
7	7-7-2024	<ul style="list-style-type: none"> <li>- Sidang 19/7/24</li> <li>- Judul Saya yg 1 terbaik</li> <li>- Proposisi - Skripsi</li> <li>- Headline → daftar isi</li> <li>- Pertukaran - Pertukaran tifromy;</li> <li>↓ Keadaan ini harus:</li> <li>adalah pertukaran siber → menyatakan</li> <li>Bisa → Terdapat</li> <li>dan merupakan yang membaca</li> <li>untuk melaporkan</li> <li>U.I.I kerinduan → jadi kerinduan</li> <li>Maluunt pengalaman (selanjutnya)</li> <li>→ Tahap selanjutnya ...</li> <li>Dapat dilihat lagi → garis</li> <li>↳ suplemen buku</li> <li>futuristik</li> <li>→ U.I.I implementasi</li> <li>→ U.I.II masukan data</li> <li>Bilah X → Jika ✓</li> <li>Pertukaran X</li> </ul>

Buku Bimbingan dan Log Book Penelitian Skripsi FSTP Unhan RI |

No.	Tanggal	Konsultasi (Saran / Perbaikan)	Ttd
		<p>ini → aduan algoritma          4.1.3</p> <p>1. Sni di dapatis → didapati</p> <p>Kompres; → OJT Kompres;</p> <p>Modifikasi 4.1.3 → diperbaiki</p> <p>algoritma ini → algoritma apa?</p> <p>dimensioni → usi coba</p> <p><del>data → web → Hall</del></p> <p>Data → <sup>algoritma Kompresi</sup>          Penguruan nilai n dipisah          dengan web</p> <p>dipisah antara web          dan halaman</p>	

No.	Tanggal		
8	10-2-2024	<ul style="list-style-type: none"> <li>- Beta min 2 kali sebelum print</li> <li>- Rapikan indensasi</li> <li>- label rapikan</li> </ul>	file

Buku Bimbingan dan *Log Book* Penelitian Skripsi FSTP Unhan R