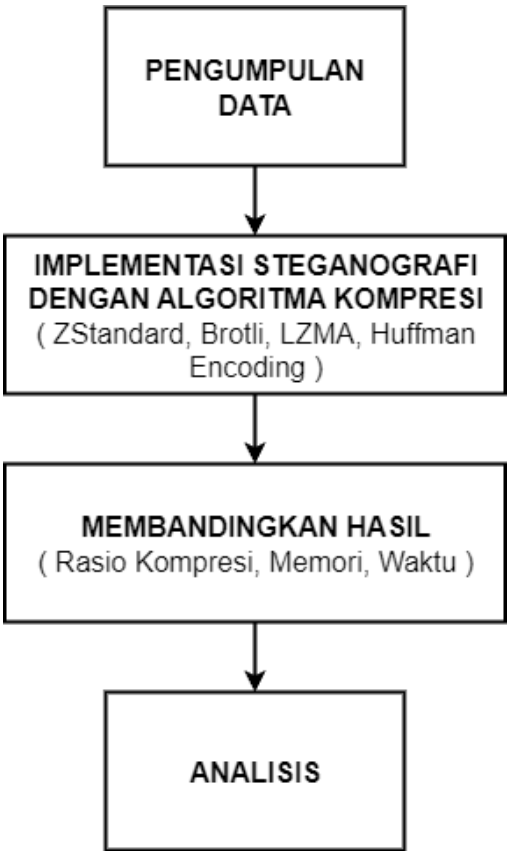
BAB III

METODOLOGI PENELITIAN

3.1. Metodologi Penelitian



Gambar 3. 1 Tahapan Penelitan

Gambar 3.1 menunjukkan alur tahapan penelitian yang terdiri dari empat tahap utama, yaitu pengumpulan data, pengimplementasian steganografi EOF dengan empat algoritma kompresi (*ZStandard*, *Brotli*, LZMA, dan *Huffman Encoding*), membandingkan hasil steganografi, serta analisis. Penelitian dimulai dengan menyiapkan dataset berupa *cover image* dan *file* rahasia berbagai format, kemudian diterapkan metode EOF yang dikombinasikan dengan algoritma kompresi. Selanjutnya dilakukan pengujian terhadap rasio kompresi, durasi pemrosesan,

penggunaan memori, dan tingkat keberhasilan proses, yang hasilnya dianalisis untuk menentukan algoritma paling optimal dalam mengoptimalkan metode EOF pada steganografi gambar.

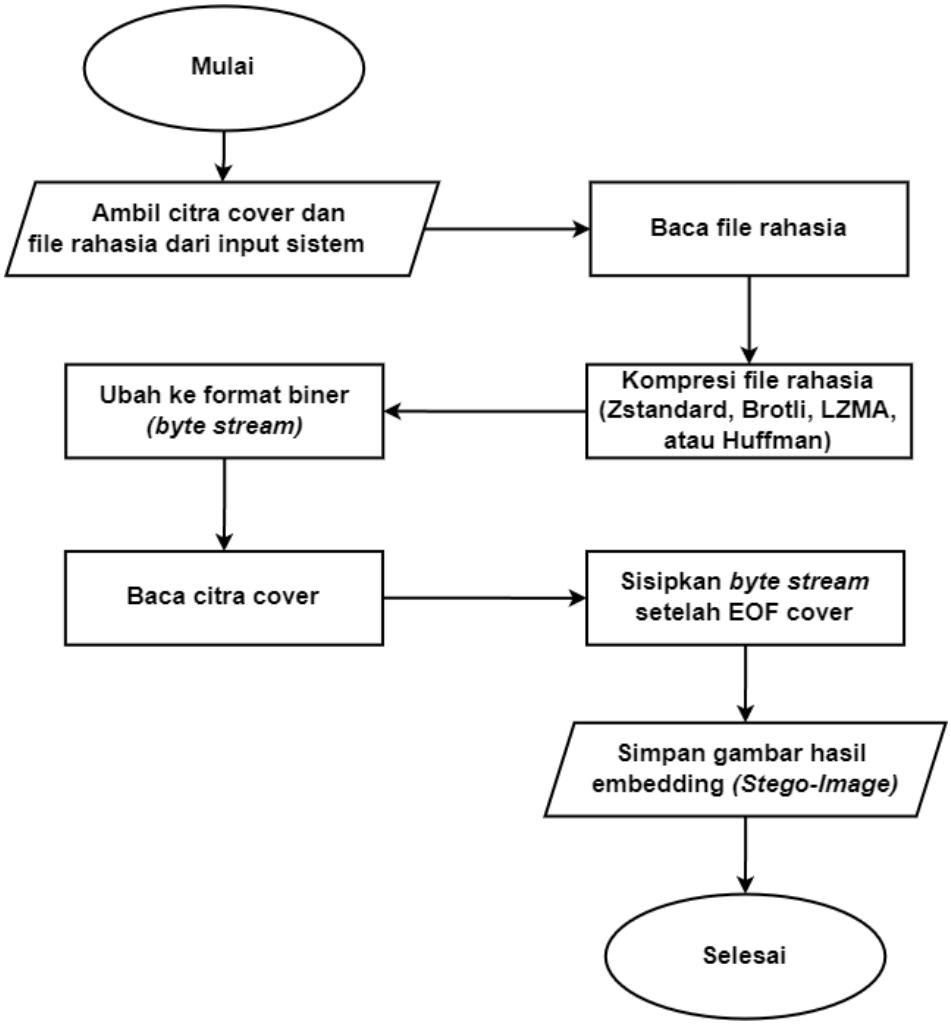
3.1.1 Pengumpulan Data

Pada tahap ini, seluruh data yang diperlukan dalam penelitian disiapkan secara sistematis. Dataset yang digunakan terdiri atas citra digital berformat PNG yang berperan sebagai media penampung (*cover*), serta pesan rahasia yang berbentuk *file* JSON, WAV, TXT, CSV dan BMP. Pemilihan format *file* tertentu sebagai data uji didasarkan pada temuan bahwa format-format ini memiliki tingkat redundansi yang tinggi dan belum terkompresi. Hal ini sejalan dengan penelitian oleh (Yang dkk., 2023), yang menunjukkan bahwa format *file* yang tidak terkompresi seperti BMP dan TXT menawarkan rasio kompresi tertinggi, sehingga ideal untuk mengukur kinerja maksimal dari sebuah algoritma kompresi. Data *file* rahasia diperoleh dari situs penyedia *file* contoh seperti file-examples.net, examplefile.com, dan json-format.com.

3.1.2 Implementasi Steganografi dengan Algoritma Kompresi

Implementasi dilakukan dengan menggabungkan metode steganografi berbasis *End of File* (EOF) dengan algoritma kompresi data, yaitu *ZStandard*, *Brotli*, LZMA, dan *Huffman Encoding*. Proses ini dilakukan melalui dua tahap utama yaitu *embedding* (penyisipan) dan *extracting* (ekstraksi).

1. *Flowchart* Penyisipan *(Embedding)*

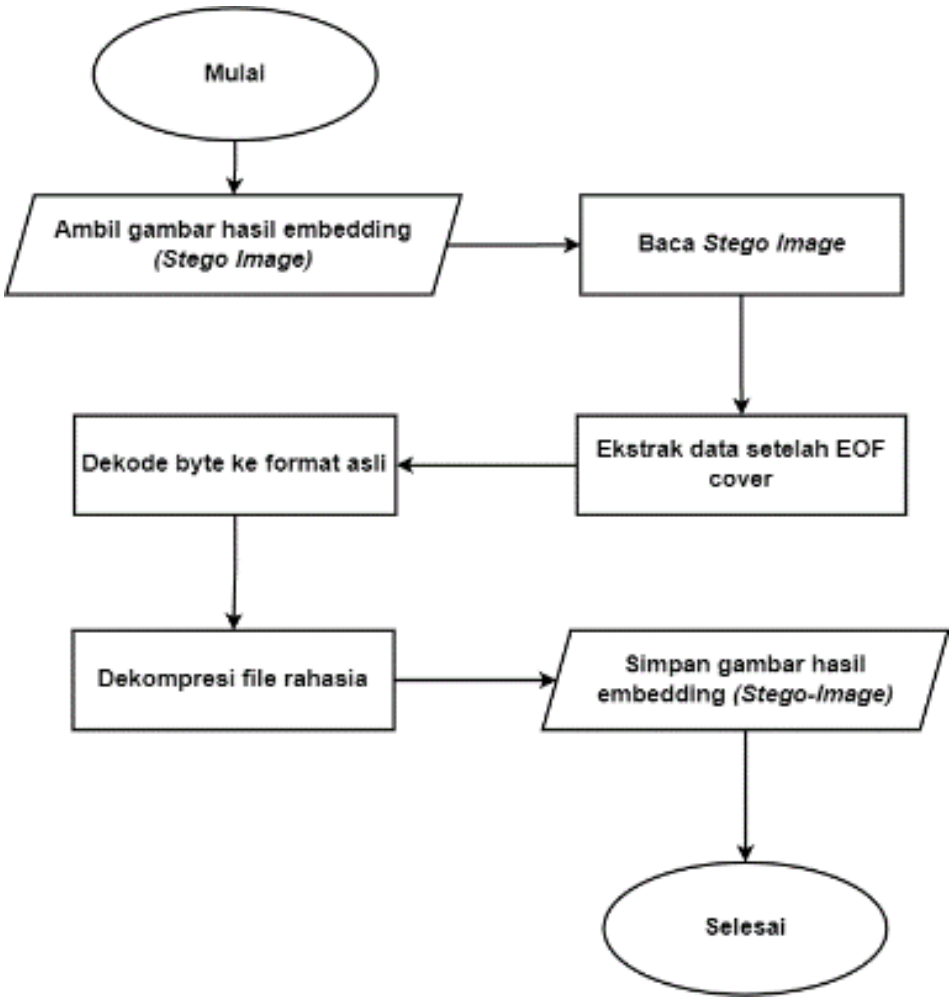


Gambar 3. 2 *Flowchart* Penyisipan *(Embedding)*

Gambar 3.2 menjelaskan proses penyisipan data pada penelitian ini dilakukan menggunakan metode *End of File* (EOF) yang dikombinasikan dengan algoritma kompresi. Tahapannya dimulai dengan memilih citra *cover* dan *file* rahasia sebagai masukan sistem. *File* rahasia kemudian dibaca dan dikompresi menggunakan salah satu algoritma yang ditentukan (*Zstandard*, *Brotli*, LZMA, atau *Huffman Encoding*), lalu diubah menjadi *byte stream*. Secara paralel, sistem membaca citra *cover* dalam bentuk *byte* untuk menentukan posisi *End of File* (EOF). Data rahasia yang telah dikompresi kemudian ditambahkan tepat setelah *marker* EOF pada citra

cover tanpa memodifikasi piksel aslinya. Hasil *embedding* disimpan sebagai *Stego Image,* yang berukuran lebih besar dibanding citra awal. Setelah *stego image* berhasil disimpan, proses penyisipan dinyatakan selesai.

2. *Flowchart* Ekstraksi *(Extracting)*



Gambar 3. 3 *Flowchart* Ekstraksi *(Extracting)*

Gambar 3.3 menjelaskan proses ekstraksi. Proses ekstraksi dimulai dengan membaca seluruh isi berkas *stego* dalam bentuk *byte stream*. Sistem kemudian mengidentifikasi batas akhir data asli citra (EOF) untuk menentukan posisi awal pesan yang disisipkan. *Byte* yang berada setelah posisi EOF dianggap sebagai data pesan yang disembunyikan. Tahap berikutnya adalah penyusunan ulang *byte* hasil

ekstraksi menjadi berkas terkompresi yang valid sesuai algoritma yang digunakan saat penyisipan (misalnya *Zstandard*, *Brotli*, LZMA, atau *Huffman Encoding*). Setelah itu, dilakukan dekompresi terhadap berkas tersebut untuk memperoleh kembali pesan asli dalam format awalnya. Proses ini memastikan integritas data, karena metode EOF tidak mengubah isi *byte* selama penyisipan. Oleh karena itu, *byte* hasil ekstraksi identik dengan *byte* yang dimasukkan pada tahap *embedding*, sehingga pesan dapat dipulihkan secara utuh tanpa kehilangan informasi.

3.1.3 Membandingkan Hasil Steganografi

Setelah tahap implementasi selesai, dilakukan perbandingan hasil steganografi untuk mengukur performa masing-masing algoritma kompresi. Parameter yang diukur meliputi tingkat keberhasilan proses penyisipan dan ekstraksi, rasio kompresi, durasi pemrosesan ketika penyisipan, penggunaan memori ketika penyisipan, durasi pemrosesan ketika ekstraksi dan penggunaan memori ketika ekstraksi.

1. Rasio kompresi

Parameter ini mengukur efektivitas algoritma dalam memperkecil ukuran *file* rahasia. Rasio kompresi dihitung menggunakan persamaan (3.1).

$$\boldsymbol{Compression\ Rate}\ (\%) = ((\boldsymbol{x} - \boldsymbol{y}) / \boldsymbol{x}) * \mathbf{100}\% \qquad (3.1)$$

Rumus rasio kompresi dalam persentase menunjukkan pengurangan ukuran data dengan mengurangi ukuran terkompresi (y) dari ukuran asli (x), membagi hasilnya dengan ukuran asli, dan kemudian mengalikannya dengan 100 untuk menyatakannya sebagai persentase.

2. Penggunaan memori (penyisipan)

Parameter ini mengukur jumlah memori (RAM) maksimum yang digunakan oleh sistem selama proses penyisipan data. Penggunaan memori yang lebih rendah menunjukkan algoritma yang lebih hemat sumber daya. Metode pengukuran memori menggunakan pendekatan akumulasi memori puncak untuk mengatasi perilaku dinamis dari *Garbage Collector Python*. Proses pengukuran dilakukan dengan langkah-langkah berikut:

1. Fungsi `gc.collect()` dijalankan untuk membersihkan sisa memori dari proses sebelumnya dan menetapkan titik awal pencatatan yang bersih.
2. Instrumen pengukuran diimplementasikan menggunakan library `psutil`. Sistem melakukan pelacakan puncak secara berkala selama proses berjalan. Jika penggunaan memori saat ini melebihi nilai tertinggi yang pernah tercatat, maka nilai puncak tersebut akan diperbarui.
3. Setelah sub-proses (kompresi atau penyisipan) selesai, sistem melakukan pembersihan pasca-proses dengan menghapus variabel data besar dari memori guna mencegah bias pada pengukuran tahapan selanjutnya.
4. Nilai memori puncak dari tahap Kompresi dan tahap Penyisipan dijumlahkan untuk merepresentasikan total kebutuhan memori sistem secara keseluruhan.

Total penggunaan memori dihitung menggunakan persamaan (3.2).

$$\boldsymbol{M_{total\_penyisipan} = M_{kompresi} + M_{sisip}} \quad (3.2)$$

Keterangan :

$\boldsymbol{M_{total\_penyisipan}}$ = Total memori yang digunakan dalam satu pengujian (dalam byte atau KB).

$\boldsymbol{M_{kompresi}}$ = Memori yang digunakan untuk mengompresi *file* yang disisipkan menggunakan algoritma kompresi (dalam *byte* atau KB).

$\boldsymbol{M_{sisip}}$ = Memori yang digunakan untuk menyisipkan data terkompresi ke gambar berformat .png menggunakan metode EOF (dalam *byte* atau KB).

3. Durasi penyisipan

Parameter ini mengukur total waktu yang dibutuhkan untuk menyelesaikan keseluruhan proses penyisipan data, mulai dari kompresi hingga menghasilkan *stego-image*. Instrumen pengukuran durasi pemrosesan diimplementasikan menggunakan modul standar `time`. Pengukuran dilakukan dengan mekanisme berikut:

1. Sistem mencatat waktu saat ini (`t_{start}`) dalam satuan nanodetik tepat sebelum sebuah sub-proses (seperti kompresi atau penyisipan) dijalankan. Penggunaan satuan nanodetik bertujuan untuk menangkap presisi waktu setinggi mungkin
2. Segera setelah proses algoritma berakhir, sistem kembali mencatat waktu aktual saat itu sebagai waktu akhir (`t_{end}`)
3. Durasi eksekusi diperoleh dengan menghitung selisih antara waktu akhir dan waktu awal, kemudian dikonversi dari nanodetik menjadi satuan milidetik (ms) atau detik sesuai kebutuhan.

Total durasi pemrosesan dihitung menggunakan persamaan (3.3).

$$\boldsymbol{T_{total\_penyisipan} = T_{kompresi} + T_{penyisipan}} \quad (3.3)$$

Keterangan :

$\boldsymbol{T_{total\_penyisipan}}$ = Waktu total untuk satu pengujian (dalam detik).

$\boldsymbol{T_{kompresi}}$ = Waktu untuk mengompresi data yang disisipkan (dalam detik).

$\boldsymbol{T_{penyisipan}}$ = Waktu untuk menyisipkan data terkompresi ke gambar berformat .png (dalam detik).

4. Penggunaan memori (ekstraksi)

Parameter ini mengukur jumlah memori (RAM) maksimum yang digunakan oleh sistem selama proses pengekstraksian data. Penggunaan memori yang lebih rendah menunjukkan algoritma yang lebih hemat sumber daya. Proses pengukuran Memori Ketika ekstraksi sama dengan proses pengukuran memori Ketika penyisipan. Mulai dari membersihkan memori sebelum ekstraksi dimulai, melacak memori tertinggi selama proses ekstraksi dan dekompresi berlangsung. Setelah sub-proses selesai, dilakukan pembersihan memori untuk mencegah bias data.

Total penggunaan memori dihitung menggunakan persamaan (3.4).

$$\boldsymbol{M_{total\_ekstraksi} = M_{ekstraksi} + M_{dekompresi}} \quad (3.4)$$

Keterangan :

$\boldsymbol{M_{total\_ekstraksi}}$ = Total memori yang digunakan dalam satu pengujian (dalam byte atau KB).

$\boldsymbol{M_{ekstraksi}}$ = Memori yang digunakan untuk mengesktraksi *file* yang disisipkan (dalam *byte* atau KB).

$\boldsymbol{M_{dekompresi}}$ = Memori yang digunakan untuk mendekompresi data yang terkompresi menjadi data asli (dalam *byte* atau KB).

5. Durasi pemrosesan (ekstraksi)

Parameter ini mengukur total waktu yang dibutuhkan untuk menyelesaikan keseluruhan proses pengekstraksian data. Pengukuran dilakukan menggunakan modul standar `time` dengan mencatat waktu awal (`t_{start}`) sebelum proses ekstraksi dan dekompresi dimulai, serta waktu akhir (`t_{end}`) setelah proses selesai. Selisih waktu dalam nanodetik kemudian dikonversi untuk mendapatkan durasi eksekusi yang presisi. Nilai puncak dari tahap Ekstraksi dan Dekompresi kemudian dijumlahkan.

Total durasi pemrosesan dihitung menggunakan persamaan (3.5).

$$\boldsymbol{T_{total\_ekstraksi} = T_{ekstraksi} + T_{dekompresi}} \quad (3.5)$$

Keterangan :

$\boldsymbol{T_{total\_ekstraksi}}$ = Waktu total untuk satu pengujian (dalam detik).

$\boldsymbol{T_{ekstraksi}}$ = Waktu untuk mengesktraksi *file* yang disisipkan (dalam detik).

$\boldsymbol{T_{dekompresi}}$ = Waktu untuk mendekompresi data yang terkompresi menjadi data asli (dalam detik).

3.1.4 Analisis

Analisis dilakukan dengan menganalisis beberapa parameter utama, yaitu rasio kompresi, durasi pemrosesan, dan penggunaan memori baik pada proses penyisipan maupun ekstraksi.