**PROJECT IMPLEMENTASI STEGANOGRAFI**



**Disusun Oleh :**

Nama : Gede Krisna Surya Artajaya

NIM : 2008561074

**Dosen Pengampu :**

I Komang Ari Mogi, S.Kom., M.Kom.

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS UDAYANA**

**2023**

# Penjelasan Program

Program yang dibuat merupakan implementasi dari steganografi menggunakan teknik LSB, di mana program ini dapat melakukan penyisipan pesan serta mengambil pesan yang sudah disisipkan pada gambar. Program ini dibuat menggunakan bahasa pemrograman python dengan library tkinker untuk membuat GUI dan library pillow untuk pemrosesan gambar.

Impelentasi dari teknik LSB terdapat pada class Backend, di mana terdapat dua fungsi yakni fungsi embed\_message untuk melakukan penyisipan pesan dan fungsi extract\_message untuk mengambil pesan yang sudah disisipkan pada gambar.



Gambar 1. Fungsi embed\_message

Pada fungsi embed\_message dilakukan beberapa langkah sebagai berikut:

1. **Membuka Gambar**: Pertama-tama, gambar yang akan digunakan untuk menyisipkan pesan dibuka menggunakan modul PIL dengan menggunakan fungsi **Image.open(self.image\_path)**.
2. **Konversi Pesan ke Biner**: Pesan yang ingin disisipkan dikonversi ke dalam representasi biner menggunakan metode **text\_to\_binary**. Representasi biner dari setiap karakter pesan ditambahkan ke dalam variabel **binary\_message**.
3. **Penanda Akhir Pesan**: Sebuah penanda akhir pesan ditentukan (dalam hal ini, '\*\*\*\*\*\*\*\*') untuk menandai akhir pesan yang akan disisipkan. Representasi biner dari penanda ini juga ditambahkan ke dalam **binary\_message**.
4. **Iterasi Melalui Piksel Gambar**: Setiap piksel dalam gambar diakses menggunakan loop **for pixel in pixels**. Setiap piksel direpresentasikan oleh nilai-nilai RGB.
5. **Modifikasi Piksel**: Setiap nilai RGB dari piksel diubah dengan mengganti bit paling rendahnya (LSB) dengan bit pesan yang akan disisipkan. Ini dilakukan dengan menggunakan operasi bitwise (dan, atau) pada **new\_pixel[j]**.
6. **Pergerakan ke Bit Berikutnya**: Indeks **data\_index** digunakan untuk melacak posisi saat ini dalam **binary\_message**. Setiap kali satu bit pesan telah disisipkan, **data\_index** ditingkatkan untuk memindahkan posisi ke bit berikutnya.
7. **Pembentukan Gambar Baru**: Piksel-piksel yang sudah dimodifikasi kemudian ditambahkan ke dalam list **new\_pixels**.
8. **Simpan Hasil**: Jika **output\_path** diberikan (tidak **None**), gambar yang sudah dimodifikasi dapat disimpan menggunakan **new\_image.save(output\_path)**.
9. **Kembalikan Gambar Baru**: Fungsi mengembalikan objek gambar baru yang telah dimodifikasi.



Gambar 2. Fungsi extract\_message

Pada fungsi extract\_message dilakukan beberapa langkah sebagai berikut:a

1. **Membuka Gambar**: Gambar yang berisi pesan tersembunyi dibuka menggunakan modul PIL dengan menggunakan fungsi **Image.open(self.image\_path)**.
2. **Mengambil Nilai Piksel**: Nilai piksel dari gambar diambil dan disimpan dalam bentuk list menggunakan **pixels = list(image.getdata())**.
3. **Ekstraksi Bit Pesan**: Dalam loop pertama (**for pixel in pixels**), setiap nilai RGB dari setiap piksel diiterasi. Kemudian, untuk setiap nilai tersebut, bit paling rendah (Least Significant Bit, LSB) diambil menggunakan operasi bitwise **value & 1**. Bit tersebut kemudian ditambahkan ke dalam variabel **binary\_message** sebagai representasi biner dari pesan tersembunyi.
4. **Penanda Akhir Pesan**: Sebuah penanda akhir pesan ('\*\*\*\*\*\*\*\*') ditentukan. Representasi biner dari penanda ini juga dibuat menggunakan metode **format(ord(char), '08b')** dan disimpan dalam variabel **end\_marker\_binary**.
5. **Mencari Akhir Pesan**: Menggunakan **binary\_message.find(end\_marker\_binary)**, dilakukan pencarian dalam **binary\_message** untuk menemukan indeks dari penanda akhir pesan. Jika ditemukan, indeks tersebut disimpan dalam **end\_marker\_index**.
6. **Pengecekan Akhir Pesan**: Jika **end\_marker\_index** tidak sama dengan -1, artinya penanda akhir pesan ditemukan dalam **binary\_message**. Maka, langkah-langkah berikut dilakukan:

a. **Menghapus Penanda Akhir Pesan**: Bagian pesan setelah penanda akhir dihapus dari **binary\_message** dengan menggunakan slicing (**binary\_message[:end\_marker\_index]**).

b. **Konversi Biner ke Teks**: Representasi biner dari pesan yang tersisa diubah menjadi teks menggunakan metode **binary\_to\_text**. Hasilnya disimpan dalam variabel **text\_message**.

c. **Simpan Gambar Hasil (Opsional)**: Jika **output\_path** diberikan (tidak **None**), gambar yang berisi pesan tersembunyi dapat disimpan sebagai gambar baru. Setiap nilai biner diubah kembali menjadi nilai piksel dan gambar baru tersebut disimpan menggunakan **new\_image.save(output\_path)**.

d. **Kembalikan Pesan**: Fungsi mengembalikan teks pesan yang berhasil diekstrak dari gambar.

1. **Pesan Tidak Ditemukan**: Jika penanda akhir pesan tidak ditemukan dalam **binary\_message**, fungsi mengembalikan string "Tidak ada pesan di gambar ini."