**PROJECT IMPLEMENTASI STEGANOGRAFI**

****

**Disusun Oleh:**

Nama : I Nyoman Budhiarta Suputra

NIM : 2008561110

**Dosen Pengampu :**

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

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS UDAYANA**

**2023**

**Penjelasan Program**

Dalam tugas project ini, program yang dibuat menggunakan teknik Least Significant Bit (LSB). LSB dapat diartikan sebagai konsep untuk menyembunyikan bit terakhir dari setiap bit atau sampel. Perubahan dilakukan di bit paling kanan karena bit paling kanan merupakan bit dengan nilai terendah dan tidak secara mutlak merubah nilai dari RGB sebuat gambar. Program implementasi ini dibuat menggunakan bahasa pemrograman python dengan library tkinker yang digunakan untuk membuat GUI dab library pillow yang berguna dalam pemrosesan gambar.

Masuk kedalam program, implementasi dari LSB ini terletak pada class Program, yang didalamnya sudah terdapat dua fungsi yaitu *embed\_message* dan *extract\_message*. Fungsi *embed\_message* digunakan untuk melakukan penyisipan pesan pada gambar yang dipilih. Sedangkan fungsi *extract\_message* adalah untuk menampilkan pesan yang sudah disisipkan ke dalam sebuah gambar.

**Fungsi Penyisipan Pesan Kedalam Gambar**

Didalam implementasi yang dibuat, fungsi yang digunakan untuk menyisipkan pesan kedalam gambar adalah fungsi *embed\_message* yang terdapat dalam *class* Program. Potongan *code* dapat dilihat pada gambar dibawah ini.

A computer screen shot of a program code

Description automatically generated

*Gambar 1. Fungsi Menyisipkan Pesan*

Pada fungsi *embed\_message* dilakukan dengan beberapa langkah yaitu:

1. Memilih gambar yang ingin disisipkan pesan. Dalam langkah ini menggunakan modul PIL dengan menggunakan fungsi Image.open(self.image\_path).
2. Mengkonversi pesan yang ingin diarsipkan kedalam bentuk biner dengan menggunakan fungsi text\_to\_biner yang terdapat didalam program. Setelah dikonversi lalu representasi biner dari setiap karakter akan disimpan ke dalam variable binary\_message.
3. Menandai akhir pesan yang akan diarsipkan, dan menyimpan representasi binernya kedalam variabel binary\_message.
4. Mempresentasikan nilai RGB setiap piksel menggunakan perulangan *for pixel in pixel.*
5. 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. 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. Piksel-piksel yang sudah dimodifikasi kemudian ditambahkan ke dalam list **new\_pixels**.
8. Langkah terakhir adalah memilih tempat untuk meletakkan gambar baru yang sudah disispkan pesan kedalam folder local.

**Fungsi Menampilkan Pesan Yang Disispkan**

Didalam implementasi yang dibuat, fungsi yang digunakan untuk menyisipkan pesan kedalam gambar adalah fungsi *extract\_message* yang terdapat dalam *class* Program. Potongan *code* dapat dilihat pada gambar dibawah ini.

A screen shot of a computer code

Description automatically generated

*Gambar 2. Fungsi Menampilkan Pesan Yang Disisipkan*

Pada fungsi *extract\_message* dilakukan dengan beberapa langkah yaitu:

1. MembukaGambar yang berisi pesan tersembunyi dibuka menggunakan modul PIL dengan menggunakan fungsi Image.open(self.image\_path).
2. Mengambil 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:
7. **Menghapus Penanda Akhir Pesan**: Bagian pesan setelah penanda akhir dihapus dari **binary\_message** dengan menggunakan slicing (**binary\_message[:end\_marker\_index]**).
8. **Konversi Biner ke Teks**: Representasi biner dari pesan yang tersisa diubah menjadi teks menggunakan metode **binary\_to\_text**. Hasilnya disimpan dalam variabel **text\_message**.
9. **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)**.
10. **Kembalikan Pesan**: Fungsi mengembalikan teks pesan yang berhasil diekstrak dari gambar.
11. Terdapat sebuah kondisi ketika gambar tidak memiliki pesan maka fungsi akan menampilkan “Tidak ada pesan di gambar ini”. Pengecekan ini dilakukan dengan cara melihat penanda akhir pesan, jika tidak ditemukan maka gambar dianggap tidak memiliki pesan