Nama : Algof Kristian Zega
NRP : 5025231235
Kode paper : 3

Analisis paper "Enhanced Spatial Domain Image Steganography for Improved IoT Security and Privacy Applications"

Abstrak:
Di era saat ini, yang mana setiap orang bisa terhubung dengan mudah satu sama lain, memastikan keamanan dan privasi yang kuat bagi sistem berbasis IoT sangat penting, karena adanya kerentanan yang muncul akibat penyebaran IoT yang cepat.

Media paling umum yang digunakan untuk pertukaran data dengan IoT adalah gambar digital. Sehingga banyak praktisi lebih memilih mengamankan informasi dengan teknik steganografi. Steganografi adalah teknik menyembunyikan informasi di dalam media lain (seperti gambar, audio, video, dan teks) tanpa terlihat oleh orang yang tidak berhak.

Mungkin kita pernah mendengar juga tentang kriptografi dan mengira itu sama dengan steganografi, namun kenyataannya bukan. Kriptografi mengacak pesan agar tidak dapat dibaca (terlihat namun tidak dapat dibaca), sedangkan steganografi menyembunyikan keberadaan pesan itu sendiri.

Paper ini akan memperkenalkan algoritma steganografi baru yang dirancang untuk meminimalkan distorsi gambar sambil mengoptimalkan penggunaan piksel yang dapat disisipkan dalam gambar penutup (*cover image*).

Pendahuluan:
Steganografi berasal dari kata Yunani "steganos" yang berarti "tulisan tersembunyi." Teknik ini telah digunakan oleh banyak praktisi untuk menyembunyikan informasi rahasia dalam objek digital, memperkuat ketidakterdeteksian informasi oleh pihak ketiga.

Steganografi dalam gambar dapat diklasifikasikan berdasarkan metode penyembunyiannya, yaitu: domain spasial, transformasi (frekuensi), pembuatan penutup (cover creation), dan metode numerik. Paper ini akan berfokus pada metode domain spasial.

Steganografi sering kali menghadapi kompromi antara jumlah data rahasia yang dapat disisipkan dengan kualitas gambar steganografi yang dihasilkan. Jika memaksakan memasukan data rahasia terlalu banyak, kualitas gambar akan menurun dan membuat kecurigaan di pihak ketiga (steganalysis).

Untuk mengatasi isu penurunan kualitas gambar, berbagai penelitian dalam steganografi gambar digital telah berfokus pada metode penyembunyian data secara adaptif dan spasial. Dalam konteks IoT, hal ini sangat penting karena perangkat IoT sering kali bergantung pada data gambar untuk berbagai aplikasi.

Selain menyeimbangkan antara kapasitas muatan dan kualitas gambar, skema steganografi dalam IoT juga harus tahan terhadap serangan dari pihak yang menggunakan steganalysis.

Metode yang diusulkan:

A. Proses Penyisipan Data

1. **Langkah 1**: Ambil citra pembawa (*cover image*) yang akan digunakan sebagai media penutup dan bit-bit rahasia *(s)* yang akan disembunyikan dalam citra tersebut.
2. **Langkah 2**: Susun ulang array 2D dari citra penutup menjadi array 1D. Setelah penyusunan ulang, kelompokkan piksel menjadi pasangan-pasangan, kemudian hitung perbedaan antar piksel did_idi tanpa tumpang tindih menggunakan persamaan (1):

$$d_i = p(n) - p(n+1) \tag{1}$$

3. **Langkah 3**: Iterasi dalam nilai-nilai d_i. Jika nilai d_i berada dalam rentang 0 hingga 3, maka sisipkan bit rahasia s ke elemen pertama dari pasangan piksel untuk menghasilkan piksel stego menggunakan persamaan (2). Jika nilai *stego_pixel* melebihi 255, maka nilai tersebut disamakan dengan *cover_pixel_pair(1)* untuk menghindari overflow dan atur nilai kunci ke 1.

$$stego\_pixel = cover\_pixel\_pair(1) + d_i + s \tag{2}$$

4. **Langkah 4**: Jika nilai d_i tidak berada dalam rentang 0–3, ulangi iterasi dengan memeriksa nilai did_idi dalam rentang 4–5. Jika sesuai, dapatkan perbedaan baru d_i' menggunakan persamaan (3), dan terapkan operasi matematis pada persamaan (4) untuk menghasilkan piksel stego baru. Kombinasikan d_i' dengan data rahasia s, dan atur nilai kunci ke 2. Jika perbedaan berada di luar rentang 0–5, piksel stego akan tetap sama dengan piksel citra penutup asli, dan nilai kunci diatur ke 0.

$$d_i' = \left\lfloor \sqrt{d_i} \right\rfloor \tag{3}$$

$$stego\_pixel = cover\_pixel\_pair(1) + d_i' + s \tag{4}$$

5. **Langkah 5**: Bangun citra stego baru dan tabel kunci (*key table*), yang berfungsi sebagai referensi untuk melacak piksel citra stego yang mengandung bit rahasia berdasarkan nilai-nilai kunci yang diperoleh dari langkah sebelumnya.

B. Proses Ekstraksi Data

1. **Langkah 1**: Ambil citra yang membawa pesan dan tabel kunci.
2. **Langkah 2**: Sama seperti Langkah 2 pada proses penyisipan data, susun piksel citra stego menjadi pasangan berdasarkan kedekatannya, dan hitung perbedaan d_i''di antara pasangan tersebut tanpa tumpang tindih menggunakan persamaan (5). Lakukan iterasi berdasarkan nilai-nilai pada tabel kunci. Jika nilai kunci sama dengan 1 atau 2, ekstrak bit rahasia s menggunakan persamaan (6). Jika nilai kunci adalah 2, bangun kembali citra penutup menggunakan persamaan (7), dan jika nilai kunci adalah 1, setelah ekstraksi bit menggunakan operasi LSB, hitung kembali citra penutup menggunakan persamaan (8). Perbedaan antara (7) dan (8) adalah pada penambahan nilai 1 yang dieliminasi di persamaan (8).

$$d_i'' = p(n) - p(n+1) \tag{5}$$

$$s = d_i'' \mod 2 \tag{6}$$

$$cover\_pixel = pixels\_stego(1) - \left\lfloor \sqrt{d_i} \right\rfloor + 1 \tag{7}$$

$$cover\_pixel = pixels\_stego(1) - \left\lfloor \sqrt{d_i} \right\rfloor \tag{8}$$

3. **Langkah 3**: Selain nilai kunci 1 dan 2, yang berarti nilai kunci adalah 0, maka piksel dalam citra stego tidak mengalami perubahan. Menggunakan piksel dari citra hasil rekonstruksi yang baru diperoleh, bangun kembali citra penutup agar memiliki tampilan visual yang identik dengan aslinya, meskipun data telah diekstrak.

Implementasi dalam kode python:

```
import numpy as np
from PIL import Image

def embed_data(cover_img, secret_bits):
    # Ubah ke grayscale sehingga mudah dilakukan operasi tidak nilai {R, G, B}
    img = np.array(cover_img.convert('L'))
    # Merubah gambar 2 dimensi menjadi 1 dimensi linear
    flat = img.flatten()
    # Menyimpan di variabel lain
    stego_img = flat.copy()
    key_table = []
    # Variabel untuk tracking index image
```

```
    idx = 0
    # Variabel untuk tracking index secret bits
    bit_idx = 0

    while idx < len(flat) - 1 and bit_idx < len(secret_bits):
        p1 = int(flat[idx])
        p2 = int(flat[idx + 1])
        d = p1 - p2
        s = int(secret_bits[bit_idx])

        if 0 <= d <= 3:
            new_pixel = p1 + d + s
            if new_pixel > 255:
                stego_img[idx] = p1
                key_table.append(0)
            else:
                stego_img[idx] = new_pixel
                key_table.append(1)
                bit_idx += 1
        elif 4 <= d <= 5:
            d_prime = int(np.floor(np.sqrt(abs(d))))
            new_pixel = p1 + d_prime + s
            if new_pixel > 255:
                stego_img[idx] = p1
                key_table.append(0)
            else:
                stego_img[idx] = new_pixel
                key_table.append(2)
                bit_idx += 1
        else:
            stego_img[idx] = p1
            key_table.append(0)

        stego_img[idx + 1] = p2  # biarkan piksel kedua tidak berubah
        idx += 2 # index image maju 2

    stego_img = np.reshape(stego_img, img.shape) # Mengembalikan dari 1
dimensi linear ke 2 dimensi semula
    return Image.fromarray(np.uint8(stego_img)), key_table

def extract_data(stego_img, key_table):
    # Ubah ke grayscale sehingga mudah dilakukan operasi tidak nilai {R, G,
B}, ubah juga ke 1 dimensi linear
    img = np.array(stego_img.convert('L')).flatten()
    # Buat simpen shape
    dummy = np.array(stego_img.convert('L'))
    # Variabel untuk tracking index image
    idx = 0
```

```
    # Penyimpanan pesan rahasia
    bits = []
    # Penyimpanan gambar asli
    new_img = img.copy()

    for key in key_table:
        p1 = int(img[idx])
        p2 = int(img[idx + 1])
        d = p1 - p2
        d_prime = int(np.floor(np.sqrt(abs(d))))

        if key == 1:
            new_img[idx] = p1 - d_prime
            s = d % 2
            bits.append(str(s))
        elif key == 2:
            new_img[idx] = p1 - d_prime + 1
            s = d % 2
            bits.append(str(s))
        # key == 0 → skip

        new_img[idx + 1] = p2
        idx += 2

    new_img = np.reshape(new_img, dummy.shape)
    return Image.fromarray(np.uint8(new_img)), ''.join(bits)

# Contoh penggunaan:
if __name__ == "__main__":
    # Buka citra grayscale
    img = Image.open("test.png")  # ganti dengan path citramu
    secret = "1101010101011100"     # data rahasia

    stego, key = embed_data(img, secret)
    stego.save("stego.png")

    # Ekstraksi kembali
    stego_loaded = Image.open("stego.png")
    new_image, extracted = extract_data(stego_loaded, key)
    new_image.save("after_extract.png")
    print("Data yang diekstrak:", extracted)
```