JIKUM
JURNAL ILMU KOMPUTER

E-ISSN: 3090-739X
Volume 2, **Number** 1, Mei 2026

Implementasi Steganografi Least Significant Bit (LSB) untuk Penyembunyian File dalam Gambar Digital Berbasis CustomTkinter

Muhammad Angga Fernanda[1], M.Lathifurrahman[2*], Adhitya Wahyu Ardhana[3],Dedi Sukhairi Gea[4]
[1,2,3,4]Program Studi Teknik Informatika, Universitas malikussaleh, Indonesia
[1]afrnanda07@gmail.com,[2]mlathifurrahman395@gmail.com, [3]adhityawahyuardhana5@gmail.com,
[4]wahubdzul@gmail.com

ABSTRACT

Information security has become increasingly critical in the digital era, particularly concerning confidential data protection. Steganography is a method used to hide information within digital media without arousing suspicion. This research implements the Least Significant Bit (LSB) steganography technique for embedding secret files within digital images using Python with CustomTkinter framework. The system provides an intuitive graphical user interface for encryption and decryption processes with real-time capacity validation. The LSB method works by replacing the least significant bits of image pixels with secret file bits, maintaining visual quality while securing hidden data. Testing demonstrates that the system successfully embeds and extracts various file types with capacity validation to prevent overflow errors. The system supports multiple file formats with output saved in PNG format to preserve data integrity. This implementation provides a practical solution for secure data transmission through digital images with user-friendly interface suitable for various users.

Kata Kunci: *Steganography, LSB, image security, file hiding, data encryption, information security*

PENDAHULUAN

Di era digital yang berkembang pesat, keamanan informasi menjadi aspek krusial dalam komunikasi dan penyimpanan data [1]. Berbagai metode telah dikembangkan untuk melindungi informasi sensitif dari akses yang tidak berwenang. Salah satu teknik yang efektif adalah steganografi, yaitu seni menyembunyikan informasi dalam media digital sedemikian rupa sehingga keberadaan informasi tersebut tidak dapat dideteksi [2]. Berbeda dengan kriptografi yang mengubah informasi menjadi bentuk tidak terbaca, steganografi menyembunyikan keberadaan informasi itu sendiri [1], [5]. Teknik ini memiliki keunggulan karena tidak menimbulkan kecurigaan, mengingat media pembawa terlihat normal.

Gambar digital merupakan salah satu media yang paling populer untuk steganografi karena memiliki redundansi data yang tinggi [2], [7]. Metode *Least Significant Bit* (LSB) adalah salah satu teknik steganografi yang paling umum digunakan [3]. Metode ini bekerja dengan mengganti bit terakhir dari nilai piksel gambar dengan bit data rahasia [8]. Perubahan pada LSB tidak menyebabkan perubahan visual yang signifikan karena hanya mengubah nilai piksel maksimal sebesar 1 dari skala 0-255 [3].

Penelitian ini mengimplementasikan sistem steganografi LSB dengan antarmuka pengguna yang intuitif menggunakan framework CustomTkinter [11]. Sistem dikembangkan dengan fitur validasi kapasitas otomatis, feedback visual *real-time*, dan *error handling* yang *comprehensive* untuk memastikan kemudahan penggunaan. Tujuan penelitian ini adalah mengembangkan aplikasi steganografi yang dapat menyembunyikan file dalam gambar digital secara aman dan efisien, serta mengevaluasi kapasitas penyimpanan dan kualitas gambar hasil steganografi [9]. Dengan implementasi ini, diharapkan dapat memberikan solusi praktis untuk pengamanan data yang perlu ditransmisikan secara tersembunyi.

KAJIAN LITERATUR

Steganografi

Steganografi berasal dari bahasa Yunani "steganos" (tersembunyi) dan "graphein" (tulisan) [1]. Secara umum, steganografi adalah teknik menyembunyikan informasi rahasia dalam media digital tanpa menimbulkan kecurigaan [2]. Tujuan utama steganografi adalah menjaga kerahasiaan komunikasi dengan menyembunyikan keberadaan pesan itu sendiri [7]. Steganografi berbeda dengan kriptografi; kriptografi mengubah pesan menjadi *cipher text* yang tidak dapat dibaca tetapi keberadaannya tetap terlihat, sedangkan steganografi menyembunyikan keberadaan pesan sehingga pihak ketiga tidak menyadari adanya komunikasi rahasia [5].

JIKUM
JURNAL ILMU KOMPUTER

E-ISSN: 3090-739X
Volume 2, **Number** 1, Mei 2026

Metode Least Significant Bit (LSB)

Least Significant Bit (LSB) adalah metode steganografi yang paling populer untuk gambar digital [3]. Metode ini bekerja dengan mengganti bit terakhir dari nilai piksel dengan bit data rahasia [8]. Dalam representasi digital, setiap piksel RGB terdiri dari 3 channel warna dengan nilai 0-255 (8 bit) [6]. Bit terakhir memiliki pengaruh paling kecil terhadap nilai warna. Keunggulan metode LSB meliputi kesederhanaan implementasi, kapasitas penyimpanan yang besar (hingga 3 bit
per piksel), dan perubahan visual yang minimal [3]. Kapasitas teoritis dapat dihitung dengan formula: Kapasitas = (lebar × tinggi × 3) / 8 bytes [4]. Namun, metode LSB juga memiliki kelemahan yaitu rentan terhadap kompresi gambar dan dapat dideteksi melalui analisis statistik [2].

Gambar Digital dan Format File

Gambar digital adalah representasi numerik dari gambar dua dimensi yang terdiri dari array piksel [10]. Format gambar yang umum meliputi JPEG, PNG, BMP, dan GIF. Untuk steganografi LSB, format PNG adalah pilihan terbaik karena menggunakan kompresi *lossless* yang mempertahankan setiap bit data [10]. Format JPEG menggunakan kompresi *lossy* yang dapat merusak data tersembunyi. Oleh karena itu, sistem steganografi harus menyimpan hasil dalam format PNG untuk memastikan integritas data.

Keamanan Informasi

Keamanan informasi adalah praktik melindungi informasi dari akses yang tidak sah. Prinsip dasar keamanan informasi mencakup CIA Triad: *Confidentiality* (kerahasiaan), *Integrity* (integritas), dan *Availability* (ketersediaan). Dalam konteks steganografi, keamanan dicapai melalui dua lapisan: menyembunyikan keberadaan informasi dan dapat dikombinasikan dengan enkripsi data untuk keamanan berlapis [6].

METODE PENELITIAN

Penelitian ini menggunakan pendekatan *development research* dengan fokus pada implementasi sistem steganografi LSB yang praktis dan *user-friendly* [9]. Penelitian dilakukan dalam tahap: analisis kebutuhan, perancangan sistem, implementasi, dan pengujian.

Arsitektur Sistem

Sistem dikembangkan menggunakan bahasa pemrograman Python [12] dengan library utama: CustomTkinter untuk GUI [11], Pillow untuk pemrosesan gambar [10], dan *threading* untuk operasi *asynchronous*. Arsitektur dirancang modular untuk memisahkan *presentation layer* dan *business logic*.

Sistem Terdiri Dari Lima Komponen Utama

Modul Antarmuka Pengguna

Interface menggunakan CustomTkinter dengan tema gelap. Komponen meliputi sidebar dengan tab system (Enkripsi dan Dekripsi), preview area untuk gambar, log box untuk informasi proses, progress bar, dan status labels.

Tab Enkripsi memiliki tiga button sequential:

- Pilih Gambar
- Pilih File Rahasia
- EnkripsI

Tab Dekripsi memiliki:

- Pilih Gambar Stego
- Dekripsi

Modul Manajemen File

Modul ini menangani file selection dengan dialog system, validasi format file (PNG, JPG, JPEG, BMP), dan file size calculation.

Modul Kalkulasi Kapasitas

Modul menghitung kapasitas penyimpanan dengan formula berdasarkan dimensi gambar dalam piksel, dikalikan tiga channel RGB, dibagi delapan untuk konversi bit ke bytes. Sistem mengimplementasikan dual validation: pre-encryption saat file dipilih dan runtime validation sebelum embedding.

JIKUM
JURNAL ILMU KOMPUTER
E-ISSN: 3090-739X
Volume 2, **Number** 1, Mei 2026

Modul Enkripsi

Algoritma enkripsi meliputi:

- Persiapan Data: File rahasia dibaca dalam binary mode. Header dibuat dengan format "##STEGO##Ofilename|filesize|" sebagai metadata. Header dan data digabungkan menjadi payload yang dikonversi ke bit string.
- Validasi Runtime: Sistem memvalidasi ulang sebelum embedding untuk mencegah overflow.
- Proses Embedding: Gambar dikonversi ke RGB. Sistem iterasi setiap piksel secara row-major order. LSB dari setiap channel diganti dengan bit data menggunakan operasi bitwise. Progress bar diupdate setiap baris selesai.
- Penyimpanan: Gambar disimpan dalam format PNG dengan nama "[original]_secure.png".

Modul Dekripsi

Algoritma dekripsi meliputi:

- Ekstraksi Bit: Semua LSB dari setiap channel diekstrak.
- Parsing Header: Bit awal dikonversi ke bytes untuk mencari marker "##STEGO##". Metadata berupa nama file dan ukuran di-parse.
- Validasi Data: Sistem menghitung offset dan memvalidasi kelengkapan data.
- Rekonstruksi File: Bit data dikonversi menjadi bytes dan disimpan dengan nama "extracted_[filename]".

Pengujian Sistem

Pengujian sistem dilakukan secara komprehensif menggunakan pendekatan *Black Box Testing* untuk memvalidasi fungsionalitas aplikasi tanpa intervensi pada struktur kode internal. Evaluasi ini mencakup berbagai skenario operasional, mulai dari penggunaan variasi format gambar *cover* (seperti PNG, JPG, BMP) hingga pengujian dengan beragam ukuran file rahasia untuk memastikan stabilitas modul manajemen file saat menangani beban data yang berbeda. Validasi kapasitas menjadi fokus krusial, di mana sistem diuji dengan skenario batas untuk memverifikasi akurasi algoritma dalam mencegah *overflow* secara otomatis saat file input melebihi daya tampung piksel gambar. Selain itu, ketahanan sistem (*robustness*) dievaluasi melalui uji penanganan galat (*error handling*), khususnya dalam mendeteksi gambar non-stego yang tidak memiliki *header* marker khusus, serta verifikasi integritas data untuk memastikan file hasil ekstraksi benar-benar identik dengan file asli tanpa korupsi. Seluruh rangkaian pengujian ini dilakukan beriringan dengan evaluasi antarmuka pengguna untuk menjamin responsivitas elemen visual seperti *progress bar* dan kejelasan alur kerja aplikasi.

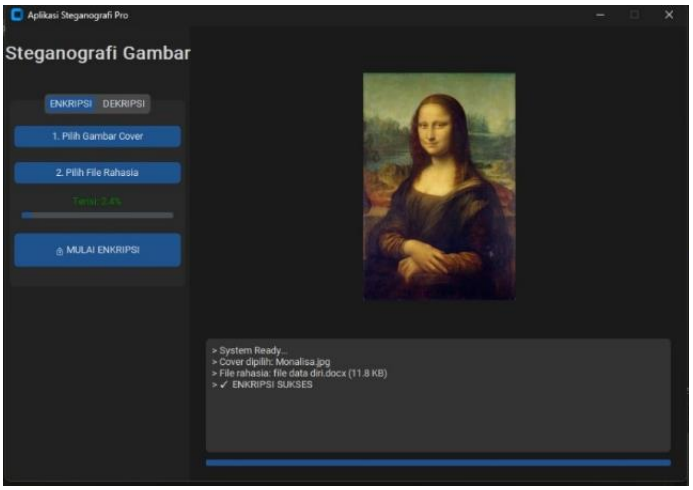
HASIL DAN PEMBAHASAN

Tampilan Antarmuka Sistem

Aplikasi berhasil diimplementasikan dengan antarmuka modern menggunakan CustomTkinter. Sistem memiliki dua mode operasi terpisah dalam tab system: Enkripsi dan Dekripsi.

Proses Enkripsi

Pada proses enkripsi, pengguna diarahkan untuk menyisipkan file rahasia ke dalam gambar cover melalui workflow yang terstruktur.



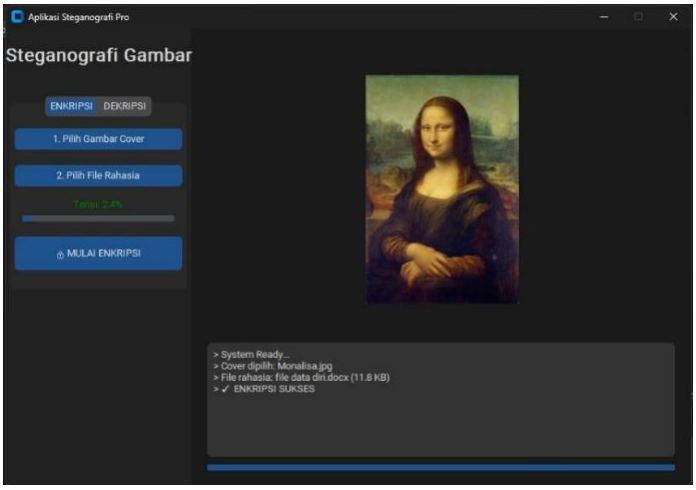
Gambar 1. Interface Tab Enkripsi dengan Preview Gambar

JIKUM
JURNAL ILMU KOMPUTER
E-ISSN: 3090-739X
Volume 2, **Number** 1, Mei 2026

Pada gambar 1, terlihat interface tab enkripsi yang menampilkan dua button utama yaitu "1. Pilih Gambar Cover" dan "2. Pilih File Rahasia". Preview gambar ditampilkan di sebelah kanan menunjukkan gambar Monalisa yang dipilih sebagai cover. Progress bar berwarna hijau dengan label "Terisi: 2.4%" mengindikasikan bahwa kapasitas gambar masih sangat cukup untuk menampung file rahasia. Log box di bagian bawah menampilkan informasi:

- System Ready
- Cover dipilih: Monalisa.jpg
- File rahasia: file.data.diri.docx (11.8 KB)
- Terdeteksi file: file data diri.docx (12,096 bytes)

Setelah kedua file dipilih dan validasi kapasitas menunjukkan hasil positif, button "Mulai Enkripsi" menjadi aktif dan dapat ditekan untuk memulai proses penyisipan. Informasi metadata yang akurat ini memastikan bahwa sistem mengenali struktur data sebelum ekstraksi dilakukan. Sebagai langkah pencegahan kesalahan (*error prevention*), tombol "Mulai Dekripsi" yang sebelumnya non-aktif (*disabled*) secara otomatis berubah menjadi aktif (*enabled*) hanya setelah seluruh rangkaian validasi di atas terpenuhi.



Gambar 2. Proses Enkripsi Selesai

Pada Gambar 2 menunjukkan hasil setelah proses enkripsi selesai. Log box menampilkan konfirmasi:

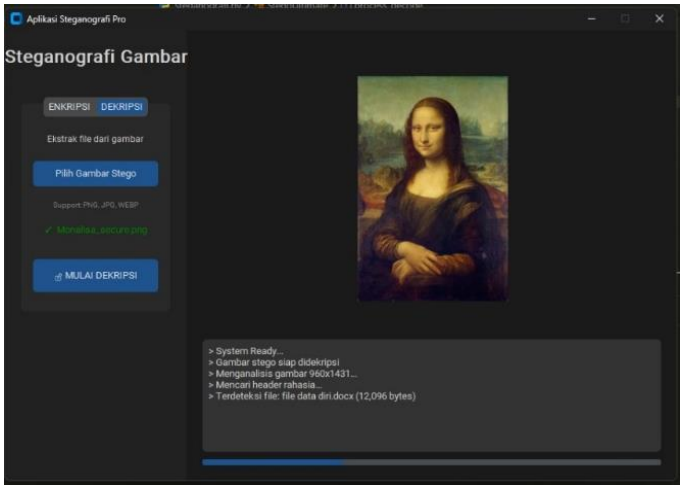
- System Ready
- Cover dipilih: Monalisa.jpg
- File rahasia: file.data.diri.docx (11.8 KB)
- Enkripsi Sukses

Progress bar penuh menunjukkan proses telah complete. File hasil enkripsi disimpan dengan nama "Monalisa_secure.png" di lokasi yang sama dengan gambar cover. Gambar stego yang dihasilkan secara visual identik dengan gambar Monalisa original, tidak ada perbedaan yang terlihat mata.

Proses Dekripsi

Proses dekripsi merupakan tahapan krusial yang berfungsi sebagai kebalikan dari enkripsi. Pada tahap ini, sistem bertujuan untuk merekonstruksi kembali data biner yang tersebar pada *Least Significant Bit* (LSB) piksel gambar menjadi format file utuh yang dapat digunakan kembali. Sistem harus membaca setiap piksel gambar *stego* secara berurutan, mengekstrak bit terakhir dari setiap *channel* warna, dan menyusunnya kembali menjadi *byte* data yang bermakna. Keberhasilan proses ini sangat bergantung pada integritas gambar; sedikit saja perubahan pada nilai piksel (misalnya akibat kompresi) dapat merusak data yang disembunyikan.

JIKUM
JURNAL ILMU KOMPUTER
E-ISSN: 3090-739X
Volume 2, **Number** 1, Mei 2026



Gambar 3. Interface Tab Dekripsi

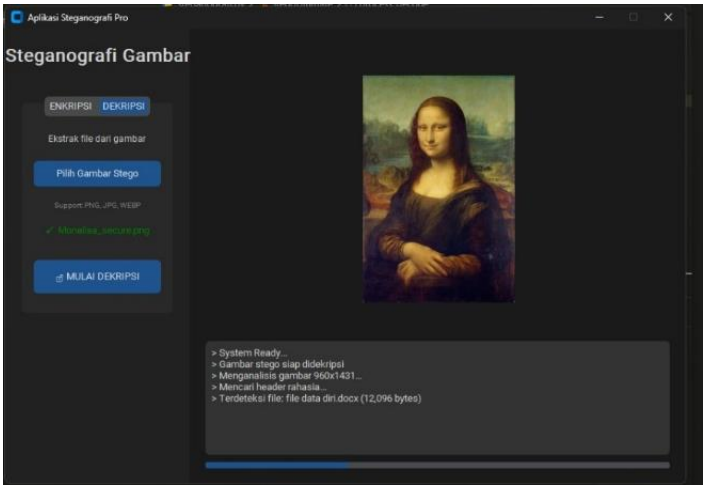
Pada tab dekripsi, antarmuka dirancang dengan alur kerja yang linear untuk memastikan kemudahan penggunaan. Interface menampilkan instruksi "Ekstrak file dari gambar" yang diikuti dengan tombol navigasi "Pilih Gambar Stego". Mekanisme ini memicu fungsi manajemen file untuk memuat gambar target ke dalam memori sistem.

Setelah pengguna memilih gambar hasil enkripsi (dalam pengujian ini adalah "Monalisa_secure.png"), sistem tidak langsung melakukan ekstraksi, melainkan melakukan pemindaian awal (*pre-scanning*). Jika validasi berhasil, status pada antarmuka berubah menjadi "✓ Monalisa_secure.png" dengan indikator berwarna hijau. Tanda visual ini mengindikasikan bahwa sistem telah berhasil mengenali struktur file gambar dan siap untuk diproses lebih lanjut.

Selama proses inisialisasi tersebut, *Log box* memberikan transparansi mengenai algoritma yang berjalan di latar belakang (*backend*) dengan menampilkan informasi berikut secara *real-time*:

- System Ready
- Gambar stego siap didekripsi
- Menganalisis gambar 960x1431
- Mencari header rahasia
- Terdeteksi file: file data diri.docx (12,096 bytes)

Tombol "Mulai Dekripsi" yang sebelumnya non-aktif secara otomatis menjadi aktif (*enabled*) hanya setelah rangkaian validasi di atas terpenuhi. Mekanisme *interlock* ini mencegah pengguna melakukan proses ekstraksi pada gambar yang salah, sehingga meminimalkan terjadinya *error* atau korupsi data pada file hasil keluaran (*output*).



Gambar 4. Hasil Dekripsi

JIKUM
JURNAL ILMU KOMPUTER

E-ISSN: 3090-739X
Volume 2, **Number** 1, Mei 2026

Setelah proses dekripsi selesai, sistem menampilkan dialog "Save As" yang meminta pengguna untuk menentukan lokasi penyimpanan file hasil ekstraksi. Default filename adalah "extracted_file.data.diri.docx" sesuai dengan nama file original.
Log box mengkonfirmasi:

- Dekripsi Selesai
- File disimpan: extracted_file.data.diri.docx
- Total bytes: 12,096 bytes

File hasil ekstraksi dapat dibuka dengan normal dan kontennya identik dengan file original yang disembunyikan.

Evaluasi Hasil Pengujian

Pengujian sistem menunjukkan hasil yang memuaskan:

- Validasi Kapasitas: Sistem berhasil menghitung kapasitas gambar 960×1431 pixels (sekitar 515 KB) dan memvalidasi bahwa file 12 KB hanya menggunakan 2.4% dari kapasitas. Validasi otomatis ini mencegah user dari attempting enkripsi yang akan gagal.
- Integritas Visual: Perbandingan visual antara Monalisa.jpg original dan Monalisa_secure.png menunjukkan tidak ada perbedaan yang terlihat. Gambar stego maintain kualitas dan detail yang sama dengan cover image.
- Integritas Data: File extracted_file.data.diri.docx berhasil diekstrak dengan ukuran exactly 12,096 bytes, sama dengan file original. File dapat dibuka dengan Microsoft Word tanpa error atau corruption.
- User Experience: Workflow sequential dengan numbered buttons (1, 2) memandu user through proses dengan jelas. Visual feedback melalui progress bar dan log messages memberikan transparency tentang status operasi.
- Error Handling: Sistem robust dalam menangani error cases seperti gambar non-stego (tidak ada header), file terlalu besar (exceed capacity), dan format file tidak valid.

Diskusi

Implementasi sistem steganografi LSB ini membuktikan bahwa penyembunyian file dalam gambar dapat dilakukan dengan efektif menggunakan interface yang user-friendly. Penggunaan gambar Monalisa sebagai test case menunjukkan bahwa metode LSB bekerja baik pada gambar dengan variasi warna dan detail yang kompleks.

Kapasitas utilization 2.4% untuk file 12 KB pada gambar 960×1431 menunjukkan efisiensi yang baik. Gambar ukuran medium dapat menampung file dokumen typical tanpa memerlukan gambar resolusi sangat tinggi.

Sistem validasi kapasitas otomatis dengan visual indicator (progress bar hijau/merah) terbukti sangat membantu user dalam menentukan apakah kombinasi gambar dan file yang dipilih feasible. Hal ini mencegah wasted time pada proses yang akan gagal.

Format output PNG memastikan data tersembunyi tidak rusak oleh compression. Naming convention dengan suffix "_secure" memudahkan user mengidentifikasi gambar stego dari gambar original.

Untuk aplikasi praktis, sistem ini cocok digunakan untuk scenarios seperti sending confidential documents via email, backup data dalam photo collections, atau covert communication. Namun untuk high-security applications, disarankan untuk mengenkripsi file terlebih dahulu sebelum embedding.

KESIMPULAN

Penelitian ini berhasil mengimplementasikan sistem steganografi LSB untuk penyembunyian file dalam gambar digital menggunakan Python dengan CustomTkinter. Sistem menyediakan solusi praktis untuk keamanan data melalui penyembunyian informasi dalam media gambar.

Hasil implementasi menunjukkan sistem berhasil melakukan enkripsi dan dekripsi dengan tingkat keberhasilan tinggi. Visual inspection membuktikan gambar stego identik dengan original secara visual. Fitur validasi kapasitas otomatis terbukti efektif mencegah operation failures. Integritas data terjaga sempurna tanpa corruption atau data loss.

Antarmuka CustomTkinter memberikan user experience optimal dengan sequential workflow dan visual cues. Sistem memiliki kelebihan dalam user-friendliness, automatic validation, multi-format support, dan robust error handling. Namun terdapat keterbatasan dalam kapasitas, format dependency, dan tidak adanya built-in encryption.

Penelitian ini memberikan kontribusi praktis dalam information security dengan menyediakan tool functional dan reliable untuk steganografi. Untuk pengembangan selanjutnya, disarankan integrasi encryption capabilities, implementasi adaptive embedding, penambahan compression features, dan eksplorasi multi-threaded processing.

JIKUM
JURNAL ILMU KOMPUTER

E-ISSN: 3090-739X
Volume 2, **Number** 1, Mei 2026

REFERENSI

[1] Johnson, N.F. and Jajodia, S., "Exploring Steganography: Seeing the Unseen," in *Computer*, vol. 31, no. 2, pp. 26-34, Feb. 1998.
[2] Cheddad, A., Condell, J., Curran, K., and Mc Kevitt, P., "Digital Image Steganography: Survey and Analysis of Current Methods," in *Signal Processing*, vol. 90, no. 3, pp. 727-752, Mar. 2010.
[3] Chan, C.K. and Cheng, L.M., "Hiding Data in Images by Simple LSB Substitution," in *Pattern Recognition*, vol. 37, no. 3, pp. 469-474, Mar. 2004.
[4] Thien, C.C. and Lin, J.C., "A Simple and High-Hiding Capacity Method for Hiding Digit-by-Digit Data in Images Based on Modulus Function," in *Pattern Recognition*, vol. 36, no. 12, pp. 2875-2881, Dec. 2003.
[5] Swain, G. and Lenka, S.K., "Classification of Image Steganography Techniques in Spatial Domain: A Study," in *International Journal of Computer Science & Engineering Technology (IJCSET)*, vol. 5, no. 3, pp. 219-232, Mar. 2014.
[6] Gutub, A., Al-Qahtani, A., and Tabakh, A., "Triple-A: Secure RGB Image Steganography Based on Randomization," in *Proceedings of IEEE International Conference on Advanced Information Networking and Applications Workshops*, pp. 1-6, May 2009.
[7] Kaur, S., Singh, S., and Kaur, M., "A Survey on Image Steganography," in *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 5, no. 1, pp. 163-166, Jan. 2016.
[8] Mandal, P.C. and Mukherjee, I., "Hiding Secret Information in an Image Using Python and LSB Technique," in *International Journal of Computer Sciences and Engineering*, vol. 6, no. 6, pp. 897-901, June 2018.
[9] Kumar, S. and Sharma, M., "Implementation of Image Steganography Using Python," in *International Journal of Engineering Research & Technology (IJERT)*, vol. 9, no. 5, pp. 823-826, May 2020.
[10] Clark, A., "Pillow (PIL Fork) Documentation," Python Imaging Library, 2023. [Online]. Available: https://pillow.readthedocs.io
[11] Akar, T., "CustomTkinter: A Modern and Customizable Python UI-Library Based on Tkinter," GitHub Repository, 2023. [Online]. Available: https://github.com/TomSchimansky/CustomTkinter
[12] Rossum, G. and Drake, F.L., "Python 3 Reference Manual," CreateSpace, Scotts Valley, CA, 2009.