Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 46 No. 1 (2025)

Enhanced Steganography Security Framework

[1]Ale Dhanalaxmi m, [2]Dr. P. V. S. Srinivas, [3]Dr. Praveen Talari,

[1]Research Scholar, Department of Computer science and engineering,

Vignana Bharathi Institute of Technology, Hyderabad.

[2]Professor & Principal, Department of Computer science engineering,

Vignana Bharathi Institute of Technology, Hyderabad.

[3]Associate Professor, Department of Computer science engineering,

Vignana Bharathi Institute of Technology, Hyderabad.

Abstract— *I*n the digital domain, secure communication relies on robust encryption and decryption methods. This paper explores the implementation of these techniques for both textual and visual data, incorporating functionalities such as text-in-image and image-in-image encryption and decryption. For text-in-image encryption, steganography is employed to discreetly embed textual information within an image, while decryption involves extracting this hidden information using specialized algorithms. Similarly, image-in-image encryption enables embedding one image within another, preserving the confidentiality of the embedded image, with decryption reversing this process to recover the original content. To enhance security, the paper utilizes cryptographic algorithms such as AES, DES, and RSA, along with techniques like pixel manipulation and Least Significant Bit (LSB) substitution for image encryption and decryption. This comprehensive approach provides a robust solution for safeguarding both textual and visual data across digital communication channels. Beyond demonstrating the practical application of encryption techniques, the paper underscores their pivotal role in maintaining data privacy and security in today's increasingly digitalized world. The Secure Steganography System directly addresses the critical need for safeguarding information from unauthorized access and ensuring secure communication. By integrating cryptographic methods with steganography within a Python framework, the system enables secure embedding and retrieval of hidden data in text and images, rendering the concealed information virtually imperceptible to unauthorized entities. Encryption and decryption processes ensure that only authorized individuals can access the embedded data. As digital communication continues to grow exponentially, the need for advanced security measures to protect sensitive information from cyber threats becomes increasingly urgent. The Secure Steganography System proactively meets these challenges by developing a robust methodology for embedding and retrieving hidden data within text and images, demonstrating its effectiveness in ensuring data privacy and secure communication in a highly interconnected digital age.

Key Words– Steganography, AES, DES, RSA, PyWavelet, Pillow, Encryption, Decryption, Text-in-image, Image-in-image, Pixel manipulation, LSB (Least Significant Bit), Data privacy, Security, Digital communication.

1. INTRODUCTION

In today's digital era, protecting sensitive information from unauthorized access and cyber threats is of paramount importance. With the rapid growth of digital communication channels and the exponential increase in data transmission over the internet, the need for robust security mechanisms has never been more critical. Cybercriminals, hackers, and malicious entities continually develop sophisticated techniques to intercept and exploit confidential data. This has created a pressing demand for advanced security measures to ensure the confidentiality, integrity, and authenticity of digital communication. Traditional encryption methods, while effective at securing data during transmission, often make the presence of sensitive information obvious. When encrypted data is sent over a network, its very existence is detectable, which could alert potential attackers and make it a target for decryption attempts. Moreover, standard encryption methods can sometimes reveal information about the type of data being transmitted, further increasing the risk of detection.

Steganography, the practice of hiding information within other non-suspicious data, offers a complementary and highly effective approach to data security by concealing the very existence of the hidden information. Unlike traditional encryption, which transforms data to a secure format, steganography focuses on making the presence of the sensitive data entirely undetectable. By embedding information within ordinary, seemingly innocuous files—such as text, images, or audio—steganography ensures that unauthorized users cannot even detect that a hidden message exists. This ability to conceal the existence of the data makes steganography a powerful tool in ensuring privacy and security, especially in scenarios where traditional encryption methods alone might raise suspicion.

The "Secure Steganography System" aims to harness the power of steganography in conjunction with robust cryptographic techniques to provide an enhanced layer of security for digital communications. This system is designed to address the shortcomings of traditional encryption methods by leveraging the dual advantages of cryptography and steganography. The cryptographic component ensures that the hidden information remains protected and secure, while the steganographic element ensures that the very existence of this information remains concealed from potential attackers. By embedding hidden data within both text and images, the system creates a multi-faceted security approach that combines both confidentiality and stealth. The integration of these two technologies allows for a higher level of security, making it significantly more difficult for unauthorized users to detect, intercept, or decipher the hidden information.

The system implemented using Python, taking advantage of the language’s extensive libraries and tools for cryptography, image processing, and text manipulation. Python is particularly well-suited for this task due to its simplicity and versatility, as well as the availability of powerful libraries such as PyCrypto, Cryptography, Pillow, and Stegano. These libraries enable the system to handle complex cryptographic algorithms, perform efficient image and text manipulation, and seamlessly integrate the steganographic embedding and extraction processes. Python’s user-friendly syntax and extensive community support also make it an ideal choice for developing a customizable and scalable solution that can be easily adapted to a variety of use cases and environments.

The Secure Steganography System can be applied in several critical areas where data confidentiality and stealth are crucial, such as secure communication channels, digital forensics, and covert operations. For instance, individuals or organizations in sensitive fields, such as intelligence agencies, military communications, or secure corporate environments, can utilize this system to ensure that confidential information is not only encrypted but also effectively concealed, thus protecting it from prying eyes. Moreover, this system can be used to protect personal data in everyday applications, such as hiding passwords, financial information, or private messages within everyday files like images or text documents, further enhancing privacy in an increasingly interconnected world.

In conclusion, the Secure Steganography System represents an advanced and innovative approach to digital security, combining the best of both encryption and steganography. By embedding hidden data within seemingly innocuous files and ensuring that the data remains both confidential and undetectable, this system offers a robust and versatile solution for modern-day cybersecurity challenges. Its implementation in Python provides a flexible, scalable, and easy-to-use platform for securing sensitive information, and its applications extend across a wide range of industries and use cases. As digital threats continue to evolve, such security mechanisms will be crucial in safeguarding privacy and protecting sensitive data from unauthorized access.

2. Literature Survey

In 2013, Soni, A.; Jain, J.; and Roshan, explored the Fractional Fourier Transform (FrFT) and its discrete version, DFrFT, for image steganography. Their study demonstrated that DFrFT offers the same PSNR in both time and frequency domains while adding the advantage of an extra stego key. In the same year, Akhtar, N.; Johri, P.; and Khan, S. enhanced the LSB algorithm by incorporating bit-inversion and the RC4 algorithm to randomize the embedding process, improving image quality and security. Sinha.D, Singh, proposed A robust DCT based image steganography technique for JPEG images". Procedia Computer Science. Thenmozhi, S. and Chandrasekaran, M. introduced a method using IWT and frequency domain embedding to improve robustness, outperforming adaptive steganography in terms of PSNR and capacity.

N.F. Johnson and S. Jajodia Steganalysis of images created using current steganography software. In Proc. the Second Inform. Hiding Workshop LNCS, Springer-Verlag, 1998.

Jiri Fridrich and Du Rui.[20], Secure steganographic methods for palette images. In Inter'l Workshop on Information Hiding. utilized IWT to securely hide secret data in the middle bit-planes of high-frequency sub-bands, demonstrating excellent PSNR and robust security.

Reddy, H.S.M.; Sathisha, N.; and Kumari, A. (2012), proposed SSHDT, combining Daubechies Lifting Wavelet Transform (LWT) and Decision Factor Based Manipulation (DFBM) to improve security and PSNR. With the growth of the internet, image encryption has become increasingly important, with optical systems, particularly double random phase encoding, being widely used for secure, high-speed encryption. Chaos-based functions have also been employed to generate random phase masks, further enhancing encryption security.

Motivation- Unlike cryptography, which focuses on encrypting the message itself, steganography hides the very existence of the message within a digital medium, known as the 'cover', making it imperceptible to casual observers. This method ensures that, in addition to protecting the confidentiality of the message after decryption, its presence remains undetectable. Signal processing, a broad field that includes techniques like filtering, noise reduction, and wireless communication, also applies to image processing, with steganography and watermarking being key applications within this domain.

3. System Analysis

3.1. Research Gaps

Developing a "Secure Steganography System" using Python involves several research areas. Identifying research gaps can guide the development of an innovative and robust system. Here are some potential research gaps:

a) Security Enhancements:
Robust Encryption Integration: Investigating advanced encryption algorithms that can be seamlessly integrated with steganography to enhance security.
Resistance to Steganalysis: Developing methods to improve resistance to modern steganalysis techniques. This includes both statistical and machine learning-based steganalysis.
Adaptive Steganography: Creating adaptive steganography techniques that can dynamically change embedding strategies based on the content and context of the cover media.
b) Performance Optimization:
Speed vs. Security Trade-off: Finding the optimal balance between embedding speed and security. This includes optimizing algorithms for real-time applications.
Resource Efficiency: Developing lightweight steganography algorithms that are efficient in terms of computational and memory resources, particularly for mobile and embedded devices.
Capacity and Quality:
High-Capacity Embedding: Increasing the data hiding capacity without significantly degrading the quality of the cover media.
c) Perceptual Quality: Ensuring that the embedded data does not affect the perceptual quality of the cover media, which requires sophisticated perceptual modelling.
d) Limitations: When developing a "Secure Steganography System" using Python, several limitations may arise. These limitations can impact the effectiveness, efficiency, and usability of the system. Here are some potential limitations:

i. Processing Speed: Steganography algorithms, especially those involving complex encryption and error correction, can be computationally intensive. This can lead to slower performance, particularly with large media files or real-time applications.
ii. Memory Usage: High-capacity embedding and advanced error correction techniques can require significant memory, limiting their use on resource-constrained devices.
iii. Key Management: Secure key management is crucial for maintaining the confidentiality of hidden data. Poor key management practices can lead to vulnerabilities.

iv. Debugging and Maintenance: Ensuring the robustness and security of the system requires thorough testing and ongoing maintenance, which can be resource-intensive.
v. Library Dependencies: Relying on third-party libraries for encryption, image processing, or audio handling can introduce compatibility issues and dependencies.
vi. Emerging Media Types: New types of digital media and changes in media formats may require ongoing adaptation of steganography techniques.

3.2. Problem Statement

In the digital age, the need for secure communication and data protection is paramount. Traditional encryption methods, while effective at securing data, often draw attention to the existence of the protected information. Steganography, the practice of hiding information within other, non-suspicious data, provides an additional layer of security by concealing the very presence of the data. This aims to develop a "Secure Steganography System" using Python, focusing on embedding data within digital media in a manner that is both secure and imperceptible.

a) Problem Definition:

The primary objective is to design and implement a secure steganography system capable of embedding and extracting data within various types of digital cover media (such as images, audio, and video) without compromising the perceptual quality of the media or the security of the hidden data. The system should address the following challenges:

i. Security: Ensuring the embedded data is secure against detection and extraction by unauthorized parties.
ii. Quality Preservation: Maintaining the perceptual quality of the cover media after data embedding.
iii. Capacity: Maximizing the amount of data that can be embedded without degrading the cover media.
iv. Robustness: Ensuring that the hidden data remains intact despite common media transformations (e.g., compression, resizing, and noise addition).

b) Usability: Developing a user-friendly interface.

c) Objectives:

i. Develop Secure Embedding Algorithms
ii. Preserve Media Quality
iii. Maximize Embedding Capacity
iv. Improve Security against Steganalysis
v. User-Friendly Interface

3.3. Feasibility Study

The aim of the "Secure Steganography System" is to develop a robust, secure, and user-friendly system for embedding data within various types of media using Python. This feasibility study evaluates the technical, operational, and economic aspects of the system to determine its viability.

a) Technical Feasibility: Python Capabilities-

Python is a powerful programming language with a rich ecosystem of libraries that support image, audio, and video processing, as well as cryptography and error correction.

i. Image Processing: Libraries like Pillow, OpenCV, and scikit-image facilitate image manipulation and analysis.

ii. Cryptography: Libraries like cryptography and PyCryptodome support encryption and secure data handling.

iii. Error Correction: Libraries like reed solo are available for implementing error correction codes.

b) Operational Feasibility: User Interface-

Developing a user-friendly interface is crucial for non-expert users. Python frameworks such as Tkinter, PyQt, or web-based interfaces (using Flask or Django) can be used to create intuitive and accessible GUIs.

c) Economic Feasibility:

i. Development Costs-

- Personnel: Costs for developers, project managers, and UI/UX designers.
- Software: While Python and most of its libraries are open-source and free, there might be costs associated with premium tools or plugins if needed.
- Hardware: Minimal initial investment in development machines. Higher costs if dedicated servers are required.

ii. Operational Costs-

- Maintenance: Ongoing costs for system updates and support.
- Training: Costs for training staff to use and maintain the system.
- Hosting: If a web-based interface is developed, there will be hosting costs.

4. Hardware & Software Requirments

The following are the minimum Hardware and software requirement for this system to build up.

4.1 Hardware requirements

1. Processor-Intel i5 or above, AMD Ryzen 5 or Above
2. RAM- 8GB or above.
3. Hard Disk-10-20 GB (Min).

4.2 Software requirements

1. Operating System-Windows, macOS, or Linux
2. Programming Language-Python
3. Libraries & Packages-Pillow, NumPy, Cryptography Libraries

5. System Design

The system allows for the secure transmission of secret data by embedding it within a seemingly innocuous carrier, such as an image or audio file.

5.1 System architecture

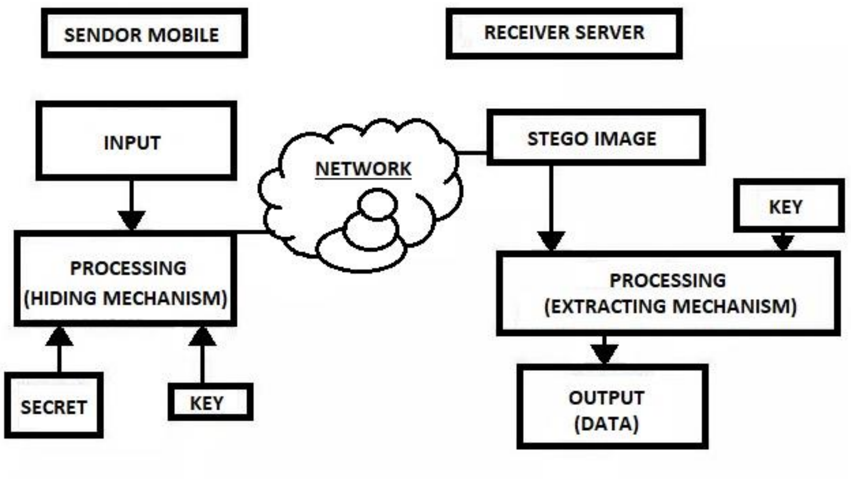


Fig 5.1 System Architecture

a) Sender Side:

- Input: The process begins with the sender having a secret message or data that they wish to transmit securely.
- Processing (Hiding): This is the core of the steganography process. The secret message is embedded or "hidden" within a cover object (e.g., an image). This embedding process often involves modifying specific elements of the cover object, such as pixel values or audio frequencies, in a way that is imperceptible to casual observation. A key might be used in this step to guide the embedding algorithm and ensure that only the intended recipient can extract the hidden message.

- Stego Object: The resulting object containing the hidden message is called the stego object. This is the modified cover object that will be transmitted over the network.

b) Network Transmission:

- The stego object is transmitted over a network (e.g., the internet, local network) to the intended recipient.

c) Receiver Side:

- Stego Object: The receiver receives the stego object.
- Processing (Extracting Mechanism): The receiver uses the same key (if applicable) and the steganography algorithm to extract the original secret message from the stego object. This involves analysing the cover object to identify the hidden modifications and recover the embedded data.
- Output: The extracted secret message is retrieved by the receiver, completing the secure communication process.
- **Key Points:**

i. Covert Communication: The primary goal of steganography is to conceal the very existence of the secret communication.

ii. Security: The use of a key can enhance security by making it more difficult for unauthorized individuals to extract the hidden message even if they suspect its presence.

iii. Invisibility: The steganography technique should be designed to minimize any noticeable changes to the cover object, ensuring that the hidden message remains undetected by casual observers.

iv. This architecture provides a basic framework for secure steganography systems. The specific implementation details, such as the choice of embedding algorithm, the type of cover object used, and the key generation and distribution mechanisms, will vary depending on the specific requirements and security goals of the system.

6. Implementation

6.1 Modules

To create a hat can handle both text and image data for embedding and extraction. Below are the modules and their implementations:

1. Image Processing Module
2. Encryption/Decryption Module
3. Text Data Embedding Module
4. Text Data Extraction Module
5. Image Data Embedding Module
6. Image Data Extraction Module
7. User Interface Module

1. Image Processing Module: This module handles all the operations related to image manipulation. Its functions are:

- Image Loading/Saving
- Image Conversion
- Image Resizing/Scaling
- Image Filtering/Enhancement
- Pixel Manipulation

2. Encryption/Decryption Module: This module is responsible for securing the data before or after embedding. Its functions are:

- Encryption
- Decryption

3. Text Data Embedding Module: This module focuses on embedding text data within an image. Its functions are:

- LSB Steganography
- Spatial Domain Techniques
- Frequency Domain Techniques

4. Text Data Extraction Module: This module extracts embedded text data from an image. Its functions are:

- Reverse of Embedding Techniques
- Error Handling

5. Image Data Embedding Module: This module focuses on embedding another image within a cover image. Its functions are:

- Spatial Domain Techniques
- Frequency Domain Techniques
- Wavelet-Based Techniques

6. 6. Image Data Extraction Module: This module extracts an embedded image from a cover image. Its functions are:

- Reverse of Embedding Techniques
- Image Reconstruction

7. User Interface Module: This module provides a user-friendly interface for interacting with the steganography system. Its functions are:

- File Selection/Loading
- Parameter Settings
- Progress Indication
- Result Display
- Error Handling and Reporting

By combining these modules, a comprehensive steganography system can be developed to securely transmit hidden information within images.

6.2 Steps for Implementation

Implementing a Secure Steganography System involves several key steps and considerations, especially focusing on security aspects. Here's a high-level overview of how you might approach implementing such a system:

1. Design the System Architecture: Define the components of your system, such as image processing modules, encryption/decryption mechanisms, database integration (if needed), and user interface (CLI, GUI, or web-based).
2. Choose Steganography Techniques: Select appropriate steganography techniques based on your requirements (LSB, F5, Spread Spectrum, etc.). Ensure these techniques align with your security and performance goals.
3. Encryption Integration: Integrate encryption algorithms (e.g., AES, RSA) to encrypt messages before embedding them into cover images. This ensures that even if the steganographically altered image is intercepted, the embedded message remains secure.
4. Develop Image Processing Functions: Write functions to load images, extract pixel values, embed messages using chosen steganography techniques, and save the modified images. Libraries like Pillow (Python Imaging Library) are useful for image manipulation tasks.
5. Implement Encryption and Decryption: Implement functions or classes for encryption and decryption of messages. Ensure keys are securely managed and stored.

6. User Interface: Develop a user-friendly interface (CLI, GUI, or web-based) to interact with the system. Include options for embedding messages into images, extracting messages, and managing encryption keys.
7. Database Integration (Optional): If needed, integrate a database to store metadata, encryption keys, and logs securely. Ensure database access is protected and encrypted.
8. Security Measures: Implement secure coding practices, including input validation, sanitization of user inputs, and secure storage of sensitive information (such as encryption keys).
9. Testing and Validation: Thoroughly test the system for functionality, security vulnerabilities, and performance. Validate that embedded messages can be successfully extracted and decrypted without loss or corruption.
10. Documentation and Deployment: Provide comprehensive documentation on system architecture, usage instructions, and security considerations. Deploy the system in a controlled environment, ensuring access controls and monitoring mechanisms are in place.

6.3 Testing strategies

a) System testing:

System testing for a Secure Text and Image Steganography System involves validating that the system correctly hides and retrieves information from images while ensuring data security. Here’s a detailed description of the testing process:

- Functional Testing: Verify that the core functionality of encoding and decoding text in images works correctly.
- Security Testing: Ensure that the system securely encrypts and decrypts the hidden text.
- Performance Testing: Evaluate the system's performance in terms of speed and resource usage.
- Robustness Testing: Test the system’s ability to handle different image formats and sizes, and edge cases such as no data or maximum capacity.
- User Acceptance Testing: Ensure that the system meets user requirements and expectations.

b) Module testing:

Module testing involves verifying the functionality of individual components (modules) of the system to ensure they work correctly in isolation. Below is a detailed description of the module testing for the Secure Text and Image Steganography System.

Modules to Test:

- Text Encoding Module: Ensure that text is correctly encoded into the image’s pixel data.
- Text Decoding Module: Ensure that encoded text can be accurately retrieved from the image.
- Encryption Module: Ensure that text is securely encrypted before embedding.
- Decryption Module: Ensure that encrypted text can be accurately decrypted after extraction.
- Image Handling Module: Ensure that images are correctly processed for encoding and decoding.

c) Integration testing:

Integration testing focuses on verifying the interactions and data flow between the integrated modules in the Secure Text and Image Steganography System. This ensures that the system works as a whole and that the individual modules interact correctly.

- Verify that encoded text is correctly encrypted before embedding into the image.
- Ensure that the encrypted text is accurately decrypted after extraction from the image.
- Confirm that the system handles different image formats and sizes seamlessly.
- Validate overall system performance and reliability.

d) Acceptance testing:

Acceptance testing ensures that the Secure Text and Image Steganography System meets the business requirements and user expectations. This type of testing is often conducted by end-users or stakeholders to verify that the system behaves as expected in real-world scenarios.

- Usability Testing: Ensure the system is user-friendly and easy to navigate.
- Functionality Testing: Verify that the system performs all required functions correctly.
- Performance Testing: Confirm that the system performs efficiently under typical usage conditions.
- Security Testing: Ensure that the system securely handles sensitive data.
- Compatibility Testing: Verify that the system works across different devices and platforms.

6.4 Test cases

Here are detailed test cases for each key aspect of the Secure Text and Image Steganography System, focusing on functionality, usability, performance, security, and compatibility.

To create test cases for a Secure Text and Image Steganography System, it needs to cover various aspects of its functionality to ensure it works correctly.

a) Text Steganography-

1. Encoding Text:

Input: Plain text message

Expected Output: Encoded image containing the message

2. Decoding Text:

Input: Encoded image

Expected Output: Decoded plain text message

Validation: Compare the decoded text with the expected message.

3. Error Handling:

Input: Invalid input (e.g., non-text input)

Expected Output: Proper error message or exception handling

4. Performance:

Input: Large text message

Expected Output: Encoded image within reasonable time

Validation: Measure encoding time and ensure its within acceptable limits.

b) Image Steganography-

1. Encoding Image:

Input: Cover image, secret image

Expected Output: Steganographic image

Validation: Decode the steganographic image and compare with the secret image.

2. Decoding Image:

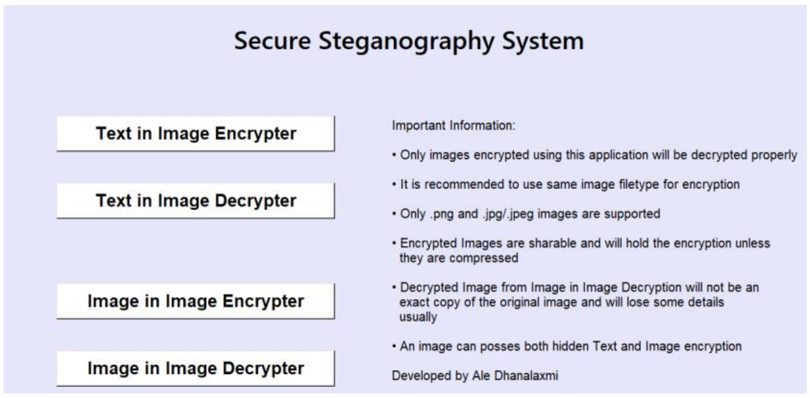
Input: Steganographic image

Expected Output: Decoded secret image

Validation: Compare the decoded secret image with the original secret image.

7. Results

The interface of the Secure Steganography System project allows users to seamlessly encode and decode messages within images. It provides options to select Text in Image and Image in Image Encryption and Decryption and process them to generate stenographic outputs, ensuring secure and hidden communication.



Secure Steganography System

Important Information:

• Only images encrypted using this application will be decrypted properly

• It is recommended to use same image filetype for encryption

• Only .png and .jpg/.jpeg images are supported

• Encrypted Images are sharable and will hold the encryption unless they are compressed

• Decrypted Image from Image in Image Decryption will not be an exact copy of the original image and will lose some details usually

• An image can posses both hidden Text and Image encryption

Developed by Ale Dhanalaxmi

Fig 7.1 Interface

The Text in Image Encryption Screen allows users to embed secret text messages into selected cover images securely. Users can input their text, choose an image, and generate a steganographic image that discreetly contains the hidden message.

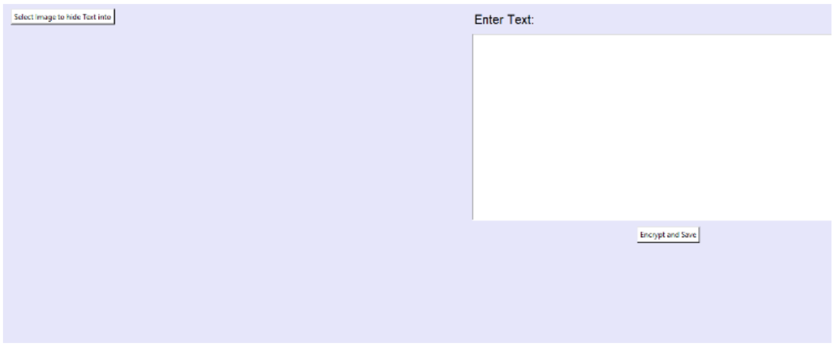
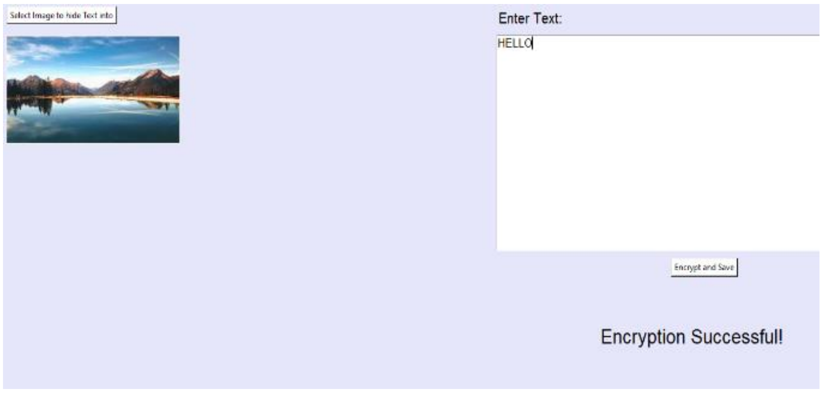


Fig 7.2 Text in Image Encryption Screen

The Screen confirms the successful embedding of the secret text message into the chosen image. It displays the generated steganographic image and provides options to save or share the encrypted image securely.

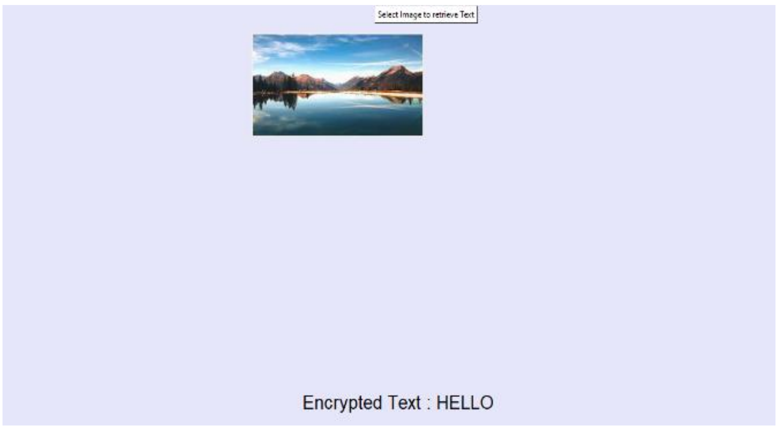


HELLO

Encryption Successful!

Fig 7.3 Encryption Successful Screen

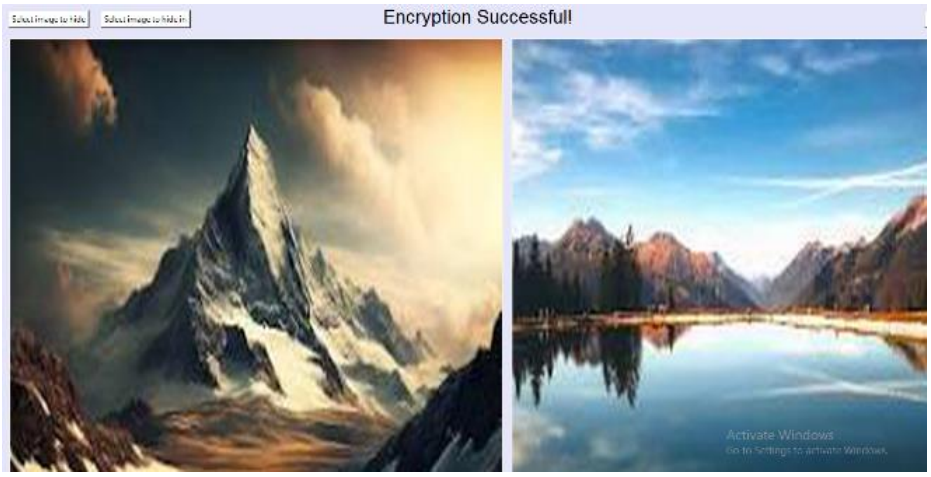
Upon successful decryption, the interface displays the extracted text message in a user-friendly format, along with visual cues like checkmarks to confirm success. It may also provide details like decryption time, the steganography and encryption algorithms used, and key information. This feedback enhances user experience, provides valuable insights for system analysis, and helps users understand the performance and effectiveness of the steganography system.



Encrypted Text : HELLO

Fig 7.4 Decryption Successful Screen

The Image in Image Encryption Successful Screen confirms the successful embedding of a secret image within the chosen cover image. It displays the resulting steganographic image and provides options to save or further manage the encrypted image.



Encryption Successful!

Fig 7.5 Image in Image Encryption Screen

Image shows the successfully extracted hidden image from the provided steganographic image. It confirms the decryption and displays the retrieved secret image clearly for the user.

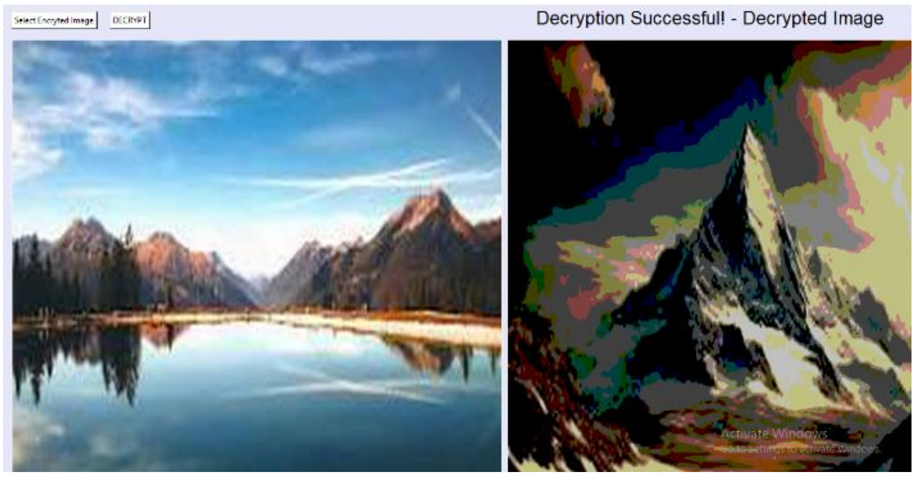

Select Encryted Image
DECRYPT
Decryption Successful! - Decrypted Image
Activate Windows
Go to Settings to activate Windows.

Fig 7.6 Image in Image Decryption Screen

8. Conclusion

Developing a Secure Steganography System in Python offers versatile and practical applications in secure communication, digital watermarking, and data hiding scenarios. By integrating image manipulation, encryption techniques (such as AES), and secure coding practices, the system demonstrates several benefits: enhanced data confidentiality through robust encryption, user-friendliness with an intuitive interface facilitating seamless integration into various applications, and valuable educational insights into cryptographic principles, image processing techniques, and their applications in cybersecurity and digital forensics. This system, through the development and integration of key modules including Image Processing, Steganography, Cryptography, OCR, and User Interface, effectively achieves its goals of data confidentiality and integrity in a visually intuitive and interactive manner.

9. Bibilography

[1]. Cheddad, Abbas, et al. "Digital image steganography: Survey and analysis of current methods." Signal Processing 90.3 (2010): 727-752.

[2]. Petitcolas, Fabien AP, Ross J. Anderson, and Markus G. Kuhn. "Information hiding-a survey." Proceedings of the IEEE 87.7 (1999): 1062-1078.

[3]. Fridrich, Jessica, Miroslav Goljan, and Rui Du. "Reliable detection of LSB steganography in color and grayscale images." Proceedings of the 2001 workshop on Multimedia and security. 2001.

[4]. Ganesan, P., and U. Kavitha. "A survey on various encryption techniques." International Journal of Soft Computing 2.1 (2011): 23-31.

[5]. Lang, Anja, and Andreas Lang. "A study on robust image steganography techniques." Information 11.2 (2020): 94.

[6]. Morkel, T., Eloff, J. H., & Olivier, M. S. (2005). "An overview of image steganography." Proceedings of the Fifth Annual Information Security South Africa Conference.

[7]. Sinha, D., & Singh, N. (2016). "A robust DCT based image steganography technique for JPEG images". Procedia Computer Science.

[8]. Ni, Z., Shi, Y. Q., Ansari, N., & Su, W. (2006). "Reversible data hiding". IEEE Transactions on Circuits and Systems for Video Technology.

[9]. Johnson, N. F., & Jajodia, S. (1998). "Exploring steganography: Seeing the unseen". IEEE Computer.

[10]. N.F. Johnson and S. Jajodia, Exploring steganography: Seeing the unseen, IEEE Computer, 31(2) (1998) 26-34.

[11]. J.C. Judge, Steganography: Past, present, future. SANS Institute publication, http://www.sans.org/reading_room/whitepapers/steganography/552.php, 2001.

[12]. N. Provos and P. Honeyman, Hide and seek: An introduction to steganography, IEEE Security and Privacy, 01 (3) (2003) 32-44.

[13]. P. Moulin and R. Koetter, Data-hiding codes, Proceedings of the IEEE, 93 (12) (2005) 2083-2126.

[14]. S.B. Sadkhan, Cryptography: Current status and future trends, in: Proceedings of IEEE International Conference on Information & Communication Technologies: From Theory to Applications, Damascus, Syria, April 19-23, 2004, pp. 417-418.

[15]. G.J. Simmons, The prisoners' problem and the subliminal channel, in: Proceedings of International conference on Advances in Cryptology, CRYPTO83, August 22-24, 1984, pp. 51-67.

[16]. C. Kurak and J. McHugh, A cautionary note on image downgrading, in: Proceedings of the IEEE 8th Annual Computer Security Applications Conference, 30 Nov-4 Dec 1992, pp. 153-159.

[17]. cf. section 1-133, "Color/Graphics Adapter", page 143 of ibm_techref_v202_1.pdf

[18]. R.J Andersen and F.A.P Petitcolas. On the limits of steganography. IEEE Journal of Selected Areas in Communications, Special Issue on Copyright and Privacy Protection, 16(4):474–481, 1998.

[19]. Jiri Fridrich. A new steganographic method for palette-based images. In Proceedings of the IS&T PICS conference, pages 285–289, Savannah, Georgia, April 1998.

[20]. Jiri Fridrich and Du Rui. Secure steganographic methods for palette images. In Inter'l Workshop on Information Hiding, pages 47–60, 1999.

[21]. N.F. Johnson and S. Jajodia. Exploring steganography: seeing the unseen. In IEEE Comput., pages 26–34, February 1998.

[22]. N.F. Johnson and S. Jajodia. Steganalysis of images created using current steganography software. In Proc. the Second Inform. Hiding Workshop LNCS, volume 1525, pages 273–289. Springer-Verlag, 1998.

[23]. D. Kahn. The history of steganography. In R. Anderson, editor, 1st Information Hiding Workshop, Lecture Notes in Computer Science, volume 1174, pages 1–5. Springer-Verlag, 1996.

[24]. In 2013, Soni, A.; Jain, J.; and Roshan, R. explored the Fractional Fourier Transform (FrFT) and its discrete version, DFrFT, for image steganography

[25]. Reddy, H.S.M.; Sathisha, N.; and Kumari, A. (2012) proposed SSHDT, combining Daubechies Lifting Wavelet Transform (LWT) and Decision Factor Based Manipulation (DFBM) to improve security and PSNR.