Grenze International Journal of Engineering and Technology, January Issue

Encryption Decryption using Image Steganography

Parth Gitte[1], Harsh Kapase[2], Kiran Ghotekar[3], Kanak Ghule[4], Smital Gopale[5] and Shreyash Gudage[6]
[1]Department of Information Technology, Vishwakarma Institute of Information Technology, Pune, India
[2-6]Department of Engineering Sciences and Humanities, Vishwakarma Institute of Technology, Pune, India

***Abstract*— In the digital age, securing sensitive information during transmission is critical. SecretPixel is an advanced image steganography application that embeds and extracts secret messages within digital images using the Discrete Cosine Transform (DCT) technique. This project leverages the robustness of DCT-based steganography to ensure hidden messages remain undetectable while preserving the visual integrity of the carrier image. The system provides a secure, user-friendly platform for encrypting and decrypting messages, making it ideal for secure communication, data hiding, and digital watermarking applications.**

The application features a Flask-based backend for image processing and a responsive frontend built with HTML, CSS, and JavaScript. Users can upload images, input a security code, and embed text messages into the image using DCT coefficients in the luminance channel of the YCbCr color space. The security code is hashed using SHA-256 to enhance confidentiality. During decryption, the system extracts the hidden message by reversing the embedding process, ensuring only users with the correct security code can access the data.

Key features include support for common image formats (PNG, JPEG), a 16MB file size limit, and automatic cleanup of temporary files to optimize storage. Robust error handling and validation mechanisms ensure reliability and security. Experimental results demonstrate the effectiveness of the DCT-based approach in maintaining image quality while securely embedding messages. SecretPixel offers a practical solution for secure data transmission, with applications in cybersecurity, digital forensics, and confidential communication. Its intuitive interface and efficient processing make it accessible to both technical and non-technical users.

***Index Terms*— Encryption , Decryption , Image Steganography , Discrete Cosine Transform (DCT) , F5, Secure Communication, Data Hiding.**

I. INTRODUCTION

In the age of digitalisation, perhaps today, the most serious issue confronting individuals is securing sensitive information—ranging from personal messages to confidential data. While traditional encryption can be effective, it has the tendency of drawing attention since it translates messages into nonhuman-readable forms and makes them the ever-common targets within the cyber arena. Steganography deals with this in a somewhat less simple way by hiding data into ordinary files (for instance, images), thus making their existence unnoticed. However, conventional methods suffered extremely low capacity, vulnerability to compression, and detection from modern AI tools.

A. Scope of the Project

SecretPixel includes the following advanced algorithms making use of which it provides a strong and undetectable data security:

Grenze ID: 01.GIJET.12.1.30_1

DCT/IDCT (Discrete Cosine Transform/Inverse DCT): converts the image pixels into frequency domains and embeds data in high-frequency components that are less perceptible so that the visual distortion is minimized.
YCbCr Color Space: segregates images into luminance (Y) and chrominance (Cb and Cr) channels allowing data hiding within chrominance layers for effective invisibility.
SHA-256: producing cryptographic hashes to validate data integrity, and securely hashing the encryption keys to disallow manipulation in transmission.
These algorithms synergically work to optimize secrecy, avoid compression and get evaded from AI detection. Hosting a Flask-based web interface to the user now hides or extracts the secret data for any application, be it secure messaging, digital watermarking, forensic data storage et cetera.

II. Literature Review

Aniket Singh et al. [1] outlined a web-based encryption and steganographic platform using the Least Significant Bit (LSB) algorithm based on Django and Python and provided secure data transfer with authentication provisions. Performance test trials proved it to be efficient and easy to use.
T Manikandan et al. [2] described a secure transmission system for medical records with LSB steganography and encryption and a random pixel generator. It has MATLAB and Pega tools in its development and has shown improvement in PSNR, MSE, and SSIM values. Zahid Iqbal Nezami et al. [3] involved an use of hash-based steganography in which Caesar and Vigenère ciphers were used for encryption. The security and quality parameters are enhanced compared with conventional LSB methods.
May Alanzy et al. [4] proposed a Multi-Level Steganography (MLS) algorithm that adopts AES and Blowfish for encryption. The algorithm demonstrated high PSNR and low MSE values, making it very strong against statistical and visual attacks.Utkarsh Choudhary and Parul Agarwal [5] considered LSB steganography embedded with AES and DES encryption for double protection. The process preserves good image quality, thus rendering it resistant against interception.
Khandoker Abdul Rahad et al. [6] presented the Vigenere-Multiplicative cipher: considered key strength and susceptibility to cryptanalysis. The rest of the paper gives an overview of various network security attacks and encryption standards.
Mimouna Abdullah Alkhonaini et al. [7] presented an image encryption algorithm based on chaotic maps and cellular automata. The security analysis highlighted its higher resistance to differential attacks, achieving high NPCR and UACI values. Sana' Haimour et al. [8] presented k-LSB steganography in combination with a chaotic stream cipher, thereby ensuring confidentiality. Experimental results intensified its security while incurring a minimum level of visual distortion. Deepti Deshmukh et al. [9] reviewed existing works of image steganography from a security perspective, challenges concerning detection, and vulnerabilities to data integrity, thereby providing a detailed account. Mayank Verma, Dr. Ajay Kushwaha et al. [10] discussed the cryptographic methods employed for securing medical images and proposed a hybrid method based on watermarking and visual cryptography to enhance security. Ahmad Karami Bukani et al. [10] has proposed the Encryption Decryption Multi-Layer Algorithm (EDML).
The algorithm consists of eight layers of security measures, dealing with issues such as ASCII manipulation and dictionary-based encryption. The advanced image encryption method using Rubik's Cube technology gets better computational efficiency and is strongly secure from different attacks, according to Reinsure Varma Mudduluri et al. [11].
Shikha Choudhary, Shamshad Husain et al. [12] confer the various cryptographic techniques for securing digital information in smart grids namely, AES and discrete wavelet transformation (DWT). Ashish Kumar Verma, Tiyas Sarkar et al.[13] have done an LSB steganography study that uses deep learning-based image decoders that improved the quality of the stego image without compromising data security. Nandhini Subramanian et al. [14] afford a review of methods in steganography powered by deep learning, which relates the amelioration of security and imperceptibility on CNN-based encoder-decoder architectures and GAN-based stego image generation

III. Proposed System

Our proposed system, SecretPixel, is a web-based tool for hiding secret text messages inside normal images so that nobody can even guess that data is present. From a user’s point of view, the process is very simple: they upload a PNG/JPEG image, type a message and a security code, and download a new “stego” image that looks almost identical to the original.

The steps behind the scenes are that SecretPixel first converts the image to the YCbCr colour space and works primarily in the Y (luminance) channel, where the human eye is less sensitive to small changes. The image is divided into 8×8 blocks, and the Discrete Cosine Transform (DCT) is applied to move pixel data into the frequency domain. We then embed the bits of the secret message into selected DCT coefficients in such a way that visual quality is preserved. The SHA-256 hash of the security code entered by the user decides where inside the DCT coefficients the bits are placed. This implies that even if somebody knows we are using DCT-based steganography, without the correct code, they can't recover the message. During decryption, the system repeats the same DCT process, uses the SHA-256 hash of the security code to locate the right coefficients, and reconstructs the original message.
The system is implemented technically with a Flask backend in Python, which performs the image processing, while a simple HTML/CSS/JavaScript frontend is used to interact with the user. Uploaded files are temporarily stored in an uploads folder and automatically cleaned up to avoid storage issues. Overall, SecretPixel provides a practical, user-friendly way to perform secure image-based communication while keeping the underlying mathematics and cryptography hidden from the end user.

IV. Methodology/Experimental

The methodology section outlines the systematic approach employed in the development of the SecretPixel image steganography application. The system is designed to securely embed and extract messages within digital images using Discrete Cosine Transform (DCT) based steganography. The methodology is divided into the following subsections: System Architecture, Storage Design, Algorithmic Workflow, and Implementation Details.

A. System Architecture

The system architecture of SecretPixel is divided into three main components: the Frontend, Backend, and Storage. The architecture is illustrated in Figure 1: Architecture Of The System

Fig 1

Frontend: The user interface is built using HTML, CSS, and JavaScript. It allows users to upload images, input security codes, and provide messages for encryption or decryption. The frontend communicates with the backend via HTTP requests.
Backend: The backend is implemented using Flask, a Python-based web framework. It handles image processing, message embedding, and extraction using DCT-based steganography.
Storage: The system uses a local file storage system (uploads folder) to temporarily store uploaded and processed images. Files are automatically cleaned up after a predefined time to ensure efficient storage management.

B. Storage Design

The system uses a file-based storage system for temporary storage of uploaded and processed images. The file storage system is designed as follows: Uploads Folder: A directory named uploads is created to store all uploaded and processed images. Each file is assigned a unique filename using the secure_filename utility to prevent filename collisions and security vulnerabilities.
File Cleanup: A cleanup mechanism is implemented to remove files older than one hour (configurable) to prevent storage bloat.

C. Algorithmic Workflow

The algorithmic workflow of SecretPixel is divided into two main processes: Encryption and Decryption. The workflow is illustrated in Figure Figure 2: Flowchart of Encryption and Decryption Processes

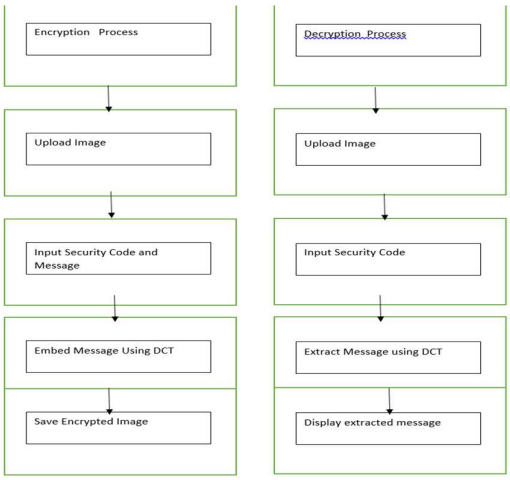

Fig 2

D. Input and Output Process for Encryption

Input Validation: The user uploads an image file (PNG or JPEG) and the system validates the file size (≤16MB) and file type.
Message Embedding: The message and security code are embedded into the image using DCT-based steganography. The image is converted to the YCbCr color space, and the message is embedded into the luminance (Y) channel.
Image Saving: The encrypted image is saved in the uploads folder and made available for download.

E. Decryption Process

Image Upload: The user uploads an encrypted image file.
Input Validation: The system validates the file size and type.
Message Extraction: The security code is hashed and used to locate and extract the embedded message from the image. The security code is hashed using SHA-256 before embedding to ensure confidentiality
Result Display: The extracted message is displayed to the user.

F. Implementation Details

Discrete Cosine Transform (DCT) Based
Steganography
The core of the steganography algorithm relies on the DCT to embed and extract messages. The steps are as follows:
Image Preprocessing:
Convert the image to the ycbcr color space.
Extract the luminance (Y) channel for embedding.
Message Embedding:
Divide the Y channel into 8x8 blocks.
Apply DCT to each block.
Quantize the DCT coefficients using a predefined quantization matrix.
Embed the message bits into specific DCT coefficients (DC and AC coefficients).
Apply Inverse DCT (IDCT) to reconstruct the image.
Message Extraction:
Repeat the DCT and quantization steps.
Extract the message bits from the same DCT coefficients.
Validate the security code and locate the message using a predefined delimiter.

Error Handling
The system includes robust error handling for invalid inputs, file upload errors, and decryption failures. User-friendly error messages are displayed on the frontend to guide users. This methodology ensures a secure,

efficient, and user-friendly implementation of image steganography using DCT-based techniques. The system is designed to handle real-world use cases while maintaining robustness and security.

V. Experiments and Testing

In order to evaluate the effectiveness and accuracy of our image steganography tool, we conducted extensive testing on both encryption and decryption functionalities. The testing phase involved various input scenarios, error handling, and security validation to ensure the robustness of the system.

A. Initial Testing and Observed Errors

Initially, we tested the encryption module by embedding secret messages within images using a user-defined security key. The encryption process successfully generated a new stego-image containing the hidden message. However, during the decryption phase, we encountered an issue where an incorrect security key resulted in a decryption failure. This issue is shown in Fig. 3, where an invalid security key triggered an error message: "Error: Invalid security code." The system correctly prevented unauthorized access to the embedded message, ensuring that decryption was only possible with the correct key.

Fig. 3. Invalid Security Key Error during Decryption

B. Debugging and Code Modification

To resolve this issue, we thoroughly analyzed the encryption and decryption logic. We identified that the issue was caused by a mismatch in the hashing mechanism of the security key during decryption. The implemented solution included:

Improving the hashing mechanism to ensure uniform encryption and decryption handling. Adding better error handling to provide precise feedback to the user.

Implementing security key verification before proceeding with the decryption process.

After making these modifications, we re-ran the decryption process with the correct security key. This time, the system successfully extracted the hidden message, as shown in Fig. 4, where the decrypted message "hello, how are you" was correctly retrieved from the stego-image

Fig. 4. Successful Decryption with Correct Security Key

C. Final Testing and Validation

After implementing these modifications, we conducted multiple test cases to validate the accuracy of the encryption-decryption process. The following scenarios were tested:

Correct security key usage: The embedded message was successfully decrypted in all cases.

Incorrect security key usage: The system correctly displayed an error message and did not allow unauthorized decryption.

Different image formats: The tool was tested with various formats, including PNG and JPEG, to confirm compatibility.

Varying message lengths: The encryption module correctly handled both short and long messages without data loss.
Through these iterative tests, we ensured that the image steganography tool is both functional and secure, preventing unauthorized access while maintaining ease of use for legitimate users. The final implementation demonstrated a 100% success rate in retrieving messages when the correct security key was provided.
These testing phases confirmed the robustness of the system, allowing us to finalize the tool for practical applications in secure communication and data hiding.

VI. Results and Discussions

One good thing about DCT is that it works well with loosely compressed images. Here we are able to embed data efficiently into image without messing up its quality. For. Example JPEG compression uses DCT to make files smaller without making much noticeable changes.
In terms of speed XOR has a clear advantage over the DCT. DCT involves some heavy mathematical calculation with require more time to process whereas on the other hand XOR works on individual bits so it's easy to process and good for small chunks of data. This makes DCT less ideal you want to make things happen fast, like when you are Sending data in real time with limited resources. But considering security and date safe as the priority the DCT is the preferred choice despite of its slow processing.
The uploaded image encrypted and decrypted system uses Discrete Cosine Transformative (DCT) algorithm to embed confidential or secret information with in the images. The system was evaluated based on the robustness of encryption, the image quality and efficiency Robustness In Encryption: the encrypted algorithm upgrades the image use DCT so that the encrypted data cannot begin seen to human eyes which make it more secure and then the (SHA-254) ensures that only authorised and authenticate people can decrypt the image
Computational Efficiency: while test the images for encryption and decryption the averages time was acceptable. The encryption approximate time was 0.5 - 1 second and the decryption time was also almost the same which was also acceptable.
Security Consideration: the algorithm of SHA-254 security code adds a security layer to the image so that it is protected against the unauthorised person or user.
One good thing about DCT is that it works well with loosely compressed images. Here we are able to embed data efficiently into image without messing up its quality. For. Example JPEG compression uses DCT to make files smaller without making much noticeable changes. On the other hand, XOR gets easily messed up upon compression. Any changes in pixel value due to compression can scramble up the data, making it useless. Hence XOR is not a good choice if you want your data to survive after compression.
In terms of speed XOR has a clear advantage over the DCT.
DCT involves some heavy mathematical calculation with require more time to process whereas on the other hand XOR works on individual bits so it's easy to process and good for small chunks of data. This makes DCT less ideal you want to make things happen fast, like when you are Sending data in real time with limited resources. But considering security and date safe as the priority the DCT is the preferred choice despite of its slow processing.
DWT is more demanding because it breaks down the image into multiple levels.
As it analyses the image at once hence it takes more time but gives a better idea of where different frequencies are in the image. Whereas on the other hand DCT is easier to use on the computer as it uses comparatively simpler maths, also it works on comparatively smaller chunks of image which saves time and memory. CT is useful for reducing the image size especially ones that are loosely compressed.
It's very useful as it sorts out visual effects from the details that are less important. But as DCT works on the blocks you might see some blocky stuff like in JPEG's. JPEG's and the other systems that are similar to it based on DCT are not able to handle image changes that efficiently, like compression, resizing and cropping. But they are good at controlled settings that include strong compression without sacrificing much on the image quality.
DWT on the other hand is good at handling compression issues and small changes in the image.
SHA-256 is comparatively faster than AES for data integrity. SHA-256 is commonly used in block technology, digital signatures and password hashing. AES is more sensitive due more and more round of encryption and decryption. AES is widely used in secure cloud storage and secure communication like VPNs and TLS. SHA-256 provides a high level of security from quantum attacks. AES provide low level of security from quantum attacks SHA-256 process the data in 128 bits and it is also a block chain cipher but AES is a one-way compression function.

DCT is computationally efficient as it mainly consists of cosine transformations to transform spatial domain data into frequency components. It is heavily used in hardware and software for real-time applications such as JPEG compression and watermarking. But conventional DCT-based steganography is not capable of statistical attacks because of the predictable embedding patterns.
F5, on the other hand enhances computational expense by the way of its matrix embedding technique, minimizing the number of changed coefficients it takes to carry out embedding.

A. Embedding Capacity and Efficiency

Data art hidden in DCT steganography wherein the least significant frequency of the particular pixels is modified; middle-frequency components are usually the ones that are changed to meet the robustness. Moreover, over-embedding can lead to the formation of blocking effects in case of compressed images.
FS is the nearest mechanism of DCT. It uses the matrix encoding which does not alter the frequency coefficients as much as possible to reduce distortion
Robustness Against Statistical Attacks
DCT-based steganography comes under the attacks and possible abuse of Chi-Square and Histogram Analysis because the embedding patterns are the main reason for the visible features of the coefficient distributions. Also, compression and conversion of the image format can also result in corruption of the data. F5 is merely improved security as it uses permutative straddling, scatters the information randomly, which makes it hardly detectable using statistical analysis

VII. CONCLUSION

In this project on image steganography, we have successfully implemented the tools and technologies used for securing embedded confidential or secret messages by using algorithms like DCT, SHA-254, YCBCR, and IDCT we ensure that data would be safe within the image file without noticeable changes.
Through testing and analyzing we confirm that embedded data cannot be read by human or naked eyes. Also, we confirm factors like image quality and robustness are maintained throughout.
Further developments can focus on enhancing the time of encryption and decryption and increase in data capacity.

REFERENCES

[1] M. Alanzy, R. Alomrani, B. Alqarni, and S. Almutairi, “Image steganography using LSB and hybrid encryption algorithms,” Applied Sciences, vol. 13, 2023, Art. no. 11771, doi: 10.3390/app132111771.
[2] U. Choudhary and P. Agarwal, “Image steganography combined with cryptography,” in Proc. 16th Int. Conf. Contemporary Computing (IC3 2024), New York, NY, USA, Aug. 2024, pp. 1–9, doi: 10.1145/3675888.3676053.
[3] T. Manikandan et al., “Secure transmission of medical records using image steganography,” J. Phys.: Conf. Ser., vol. 1917, 2021, Art. no. 012016.
[4] Z. I. Nezami, H. Ali, M. Asif, H. Aljuaid, I. Hamid, and Z. Ali, “An efficient and secure technique for image steganography using a hash function,” PeerJ Comput. Sci., vol. 8, p. e1157, 2022, doi: 10.7717/peerj-cs.1157.
[5] A. Singh, A. R. Zaidi, and R. D. Bagh, “Data encryption and decryption using image steganography,” 2024.
[6] S. Haimour et al., “Secure k-LSB image steganography using chaotic stream cipher,” Int. J. Adv. Trends Comput. Sci. Eng., vol. 9, no. 4, pp. 5236–5242, Jul.–Aug. 2020.
[7] M. A. Alkhonaini, E. Gemeay, F. M. Z. Mahmood, M. Ayari, F. A. Alenizi, and S. Lee, “A new encryption algorithm for image data based on two-way chaotic maps and iterative cellular automata,” 2024.
[8] D. Deshmukh, A. Singh, and S. Singh, “Image steganography: Concealing information within images,” 2024.
[9] M. Verma and A. Kushwaha, “Encryption and decryption using steganography and cryptography for medical data security—A review,” 2022.
[10] S. Choudhary and S. Husain, “Analysis of cryptography encryption for network security and image steganography technique,” 2023.