STEGASHIELD – A Multi-Technique Image Steganography for Enhanced Security and Undetectability

Athira Varma Jayakumar
Virginia Commonwealth University
Richmond, Virginia, USA

ABSTRACT
Image steganography, the technique of concealing information within digital images, faces challenges in achieving both security and imperceptibility. This paper presents STEGASHIELD, a novel image steganography system that combines diverse techniques to enhance these critical aspects. STEGASHIELD employs random embedding of secret pixels, AES encryption of secret data, and false image embedding to achieve a high level of security. Imperceptibility is improved through new algorithms for optimal seed and position calculation, along with pixel-by-pixel analysis for minimal distortion. To counter histogram attacks, the research introduces a histogram preservation technique that compensates for tonal value losses resulting from secret image embedding. Experimental results demonstrate that STEGASHIELD significantly outperforms the plain LSB method in both imperceptibility and security, as evidenced by improved PSNR values and histogram comparison tests. This research contributes to the field of information hiding by providing a robust steganographic solution that effectively balances the dual requirements of security and undetectability in digital image steganography.

General Terms
Security, Image processing

Keywords
Image Steganography, LSB Substitution, Security, AES Encryption, Fisher Yates Shuffle, False Image Embedding, Histogram Preservation

1. INTRODUCTION
Steganography is the art of hiding data in a cover media in such a way that others do not notice it. The different digital carriers that are used to embed secret data helps differentiate the five different types of steganographic approaches: (i) Text Steganography – to hide secret messages or information within ordinary text documents or other textual data, ii) Image Steganography – to hide secret data in an image file, (iii) Audio Steganography – to hide secret data in an audio file, (iv) Video Steganography – to hide secret data in a video file and (v) Network Steganography – to modify a single network protocol through a protocol data unit for highly secure and robust communication. While any digital carrier mentioned above can be used for steganography, this work uses Image Steganography for hiding the secret data in digital images. While hiding the secret data into an image, it is also needed to ensure that large amounts of data do not distort the quality of the cover image making the existence of secret message inside the cover image suspicious. For this reason, the steganographic method used for hiding secret data is expected to have the following characteristics:

Successful steganographic systems rely on five key properties: imperceptibility, which ensures the hidden message is visually undetectable; robustness, which allows the message to withstand distortions during transmission; security, which protects against unauthorized extraction even if detected; capacity, which determines the maximum amount of data that can be embedded while maintaining imperceptibility; and computational complexity, which considers the processing cost of embedding and extracting data. These properties collectively determine the effectiveness and efficiency of a steganographic technique in concealing and protecting secret information within cover media. This work mainly concentrates on enhancing the security and imperceptibility factors of steganography.

Image steganography techniques are broadly categorized into spatial domain and transform domain methods. Spatial domain techniques, such as Least Significant Bit (LSB) substitution, directly modify pixel intensities, offering larger embedding capacity but risking slight quality deterioration. LSB substitution, the most common spatial technique, replaces the least significant bits of cover image pixels with secret data, causing minimal visible distortion. However, it's vulnerable to attacks, and embedding large amounts of data can degrade image quality and alter histograms. Transform domain techniques, like Discrete Wavelet Transform and Discrete Fourier Transform, utilize transformed coefficients for message hiding, potentially offering better security but with lower capacity.

There are many shortcomings of the simple LSB substitution method discussed and researchers have enhanced these limitations through a variety of advanced algorithms that keep up with the quality of the cover image with increased embedding capacity. This project introduces STEGASHIELD, a system that leverages state-of-the-art steganographic algorithms to produce high-quality steganographic images, ensuring minimum distortion to the cover image while maintaining a high level of security.

2. RELEVANT WORK
Researchers have developed methods to improve LSB steganography's resistance to attacks by avoiding direct message embedding. [1] introduced a technique where the binary addition of n-image pixel bits reveals the secret data, improving imperceptibility, robustness, and capacity. Swain [2] combined LSB substitution on two LSBs with quotient value differencing (QVD) on the remaining six bits. Authors in [3] proposed the ILSB technique, providing enhanced security without complex mathematical functions. To counter unauthorized extraction, recent LSB techniques employ random pixel selection. [4] proposed an algorithm based on chaos theory, using chaotic random number generators to

determine embedding locations and color channels. [5] suggested distributing secret data across multiple cover images. [6] achieved randomness through double scrambling of the cover image before secret bit insertion, significantly improving steganographic security. Several studies aim to reduce pixel distortions after message embedding. [7] introduced a series of evaluations, scored bit rotations, and inversion operations to minimize distortion. Their approach uses extra bits in RGB bytes to indicate rotation position and inversion status. [9] proposed a simple technique of flipping all message bits if more than 50% of affected pixels would change due to LSB embedding.

Recent works have focused on enhancing steganography security by incorporating strong encryption techniques. Ajib [10] combined cryptography with steganography, encrypting secret data before embedding. [11] proposed an efficient cryptographic algorithm combining RC4 Stream cipher and RGB pixel shuffling. Muhammed et al. [12] introduced a secure color image steganographic approach using a stego key-directed adaptive LSB substitution method and multi-level cryptography. This method employs a two-level encryption algorithm (TLEA) for the stego key and a multi-level encryption algorithm (MLEA) for the secret data before embedding using adaptive LSB substitution. Rustad et al. [13] developed an inverted LSB image steganography method using adaptive patterns to improve imperceptibility by adapting the embedding process based on image characteristics. [14] introduced a hybrid cryptography and steganography method for secure data transmission in IoT. This approach combines LSB techniques with cryptographic methods to enhance overall security in Internet of Things applications. As digital communication security remains crucial, LSB steganography continues to adapt and improve, offering more robust and efficient methods for hiding sensitive information within digital images.

3. STEGANOGRAPHY TECHNIQUES

The success of steganography is to ensure that the secret message or image embedded within the container image is not detected by the warden who maybe closely monitoring the communication channel. Even if the presence of an embedded message is detected the adversary should not be able to extract the secret message/image and decode it. The main idea of this research was to design and build a steganographic system that encompasses the advantages of multiple techniques employed by various researchers and individually proven to be effective. The different concepts for improving the security and imperceptibility of steganography were explored and few were chosen to be incorporated into the STEGASHIELD image steganography system.

3.1 False Image Embedding

This steganographic technique embeds both a secret image and a false image in the cover image. The false image, easily detectable using simple LSB substitution, is designed to mislead adversaries into believing they've discovered the hidden message, potentially halting further analysis. This approach, inspired by Marek and Katarzyna [14], enhances security by concealing the true secret image. While the false image is embedded using direct LSB substitution, making it vulnerable to statistical attacks like histogram comparison, the actual secret image is secured using multiple techniques including encryption, random embedding, and pixel distortion minimization. This multi-layered approach significantly increases the difficulty of detecting and extracting the genuine secret image.

3.1 Encryption of Secret Image

Encryption of secret image offers confidentiality of the secret even if steganography is detected and adversary is successful enough to extract the embedded bits from the cover image. The developed image steganography uses the NIST certified Advanced Encryption Algorithm (AES-128) to encrypt the secret image. The AES encryption has proven to be the most secure symmetric key encryption standards. Its large key length makes brute force attack and exhaustive search of key space impossible. Its carefully designed 10 rounds of sub operations ensures complete diffusion of input bits, giving absolutely no scope for differential or linear cryptanalysis. The key used for AES encryption is only known to the sender and receiver which makes the decryption of the cipher impossible to the adversary.

3.2 Random Embedding

Another major vulnerability of the plain LSB method is the deterministic sequential embedding strategy. This weakness can be easily exploited by attackers extracting messages from suspected cover images. Inspired by the work in [1], STEGASHIELD uses the 'Random embedding' technique for embedding the secret image thereby spreading the secret image into random pixels within the cover image. Extracting the secret data bits from the cover image, reordering them and reconstructing the actual secret image becomes impossible for the attackers. On the other hand, since the false image is embedded using classical LSB substitution, all the secret bits are embedded sequentially, that makes extraction and reconstruction of false image easy.

Below techniques raise the imperceptibility of steganography:

3.3 Histogram Preservation

Histogram attack is a common attack against steganography with the attacker analyzing the histogram of cover image and detecting anomalies. Color histogram is a graphical representation of the tonal distribution in a digital image. It plots the number of pixels for each tonal value. The horizontal axis of the graph represents the (0-255) tonal variations for red, green and blue planes, while the vertical axis represents the number of pixels in that particular tone. The original tone of a cover image pixel is changed to a new tone because of modifications in LSBs of the cover image pixel when embedding the secret image bits. As the color tones in the cover image pixels get affected due to LSB substitution of secret image on cover image, histogram differences arise on the cover image. Histogram preservation technique tries to modify some extra bits to compensate for the tonal changes thereby preserving the histogram shape from being changed.

3.4 Reducing Pixel Distortion

Imperceptibility is the most important characteristic of steganography, that differentiates it from cryptography. In order to achieve imperceptibility, it is needed to ensure that only a minimal number of cover image pixel bits are affected by the LSB substitution. For ensuring this, three major optimizations are implemented in this steganography design:

1. Finding the optimal position to embed with Sequence matching.
2. Finding the optimal seed for randomization
3. Pixel-by-Pixel Distortion Analysis

These optimizations try to parse through the cover image bits that would get replaced by the encrypted secret image bits to find the optimal seed for shuffling the cover image and optimal pixel position to start embedding the secret image that yields the minimal bit distortions in the cover pixels. Pixel-by-Pixel

Distortion analysis is an analysis to reduce the bit distortions at pixel level granularity by flipping the secret bits. This concept was inspired from the work of Erna Zuni Astuti et al in [13] where they were flipping the entire secret image if the total number of bit distortions were more than 50%.

4 DETAILED DESIGN

STEGASHIELD, the novel image steganography system developed in this research, utilizes 24-bit color images with dimensions of 512x512 pixels as cover images for data concealment. The 24-bit image pixels are composed of RGB values with each of these colors having 8-bits for its tone representation. The Secret and False Images considered in this implementation are 8-bit Grayscale images. This implementation of steganography, AES encryption, the Graphical User interface and evaluation metrics that include PSNR and histogram analysis is in Java. Considering here is an application where the intention is to transfer the secret image with utmost security, but the quality of the image retrieval is not of much concern. Hence, only 4 MSB bits of the 8-bit grayscale secret image is embedded into the cover image and reconstruction of secret image happens with those 4 MSB bits. Thus, the recovered image is of lesser quality than the original image, but the communication security is guaranteed.

4.1 False image embedding

The false image considered in these experiments is a grayscale of 189x182 size. The embedding is done by extracting the 4 MSB bits of the 8-bit secret image pixel and placing that onto the 4 LSB bits of the 24-bit cover pixel. This affects the blue tone of the cover pixels. As the false image size is smaller than the cover image, only a portion of cover image is used up for the false image. The remaining part of the cover image is used up for hiding the secret image.

4.2 AES Encryption of secret image

Advanced Encryption Standard (AES) algorithm is implemented to encrypt secret images. The process involves converting the image into 128-bit blocks, generating a 256-bit encryption key, and applying the AES encryption algorithm. AES applies 10 rounds of transformations to each block of the secret image data. Each round involves several steps: a.SubBytes: Substitution of each byte using a fixed table. b.ShiftRows: Shifting the rows of the state array. c.MixColumns: Mixing the data within each column. d. AddRoundKey: XORing the data with the round key.

The resulting encrypted image appears as random noise, concealing the original content while allowing for decryption by authorized parties possessing the key.

4.3 Random Embedding using Cover Image Shuffling

To embed secret bits randomly in the cover image, first all pixels in the cover image after the last pixel, where false image was embedded, are shuffled using the Fisher-Yates algorithm. The secret image bits are then sequentially embedded into the shuffled cover image. Further, upon un-shuffling the cover image, the secret bits would be randomly distributed in the cover image. Fisher-Yates shuffle produces a uniform shuffle of an array in which each permutation of the array elements is equally likely to be produced. Unlike other inefficient algorithms, which do over shuffling of elements resulting in biased permutations, in Fisher Yates algorithm each element is only considered for a random swap once. Fisher Yates shuffle uses a random number generator to select random pixels for swapping. Fisher Yates shuffle uses a seeded random number generator to select random pixels for swapping. Implementation involves copying pixels after the false image embedding to an ArrayList and passing it with the seeded random number generator to the FisherYatesShuffle function, effectively randomizing secret bit placement while maintaining retrievability.

4.4 Reducing Pixel Distortion

4.4.1Finding optimal seed for cover image shuffling

Seed used for shuffling the cover image can be randomly chosen. The optimal seed strategy aims at minimizing the pixel bit distortions by finding the seed for shuffling the cover image that yields the least bit differences in the cover image pixels. The search algorithm tries all seed values between 0 and 1000 for shuffling the cover image starting from the position where false image embedding has been ended and calculates the total number of pixel bit differences when embedding the 256x256 encrypted secret image pixels on the shuffled cover image portion with each seed. The seed that produces the minimum pixel bit differences is considered as the optimal seed and is used for actual shuffling of the cover image. The “optimal seed” is also passed to the receiver for retrieval of the secret message from steg image.

4.4.2 Finding optimal position through Bit Sequence Matching

The optimal pixel in the shuffled part of the cover image to start embedding encrypted secret image bits is found with bit sequence matching. The 4 MSB bits from all the encrypted secret image pixels are extracted and written into an array ‘secretbits’. The 2 LSB bits from the green plane of the cover pixel and 2 LSB bits from the blue plane of the cover pixel are concatenated and written into another array ‘coverbits’. Bit sequence matching of these two arrays is performed by shifting the ‘secretbits’ array through the length of the ‘coverbits’ array and calculating the total number of bit matches. The cover pixel index at which there is maximum bit match is identified as the ‘optimal position’ for embedding the encrypted secret image in the shuffled cover image. Figure 1 demonstrates the bit sequence matching approach with start positions 1 and 2. In this example the maximum match is obtained at start position 9 with 13-bit matches.

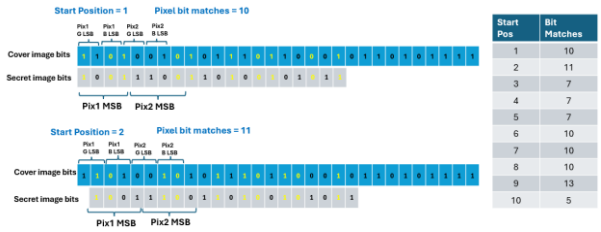

Figure 1: Bit Sequence Matching

4.5 Pixel by Pixel Distortion Analysis

In an effort to reduce the bit distortions further, this work introduces another technique of distortion analysis at each pixel

level. Secret bits to be embedded in each pixel are compared with the cover pixel bits before embedding. When embedding the 4 secret bits in the Green and Blue LSB bits of a cover pixel, if the bit difference is found to be greater than 2, then the secret bits are flipped before embedding and the flipping is indicated by setting the LSB of the red plane 8bits. To reduce bit distortions in each pixel.

- G1 G0 B1 B0 in cover pixels are compared with S7 S6 S5 S4 bits of encrypted secret bits to be embedded.
- If the bit difference is > 2, flip the secret bits and indicate that with '1' in Red LSB R0.
- If the bit difference is <= 2, retain the secret bits and indicate that with '0' in Red LSB R0.

This flipping technique can limit the maximum bit deviation per pixel after embedding the secret bits to 3 bits in the worst case.

- 0 bits matching between cover pixel and secret pixel ➔ Flipping secret bits ➔ Ends in 4 same, 1 different OR 5 same bits
- 1-bit matching between cover pixel bits and secret pixel bits ➔ Flipping secret bits ➔ Ends in 3 same and 2 different OR 4 same and 1 different bits
- 2 bits matching between cover pixel bits and secret pixel bits ➔ No Flipping of secret bits ➔ Ends in 2 same and 3 different OR 3 same and 2 different bits
- 3 bits matching between cover pixel bits and secret pixel bits ➔ No Flipping of secret bits ➔ Ends in 3 same and 2 different OR 4 same and 1 different bits
- 4 bits matching between cover pixel bits and secret pixel bits ➔ No Flipping of secret bits ➔ Ends in 4 same and 1 different OR 5 same bits

This shows the scenario where the difference between secret and cover pixel is only 1 bit:

Secret pixel **1101**1100

Cover Pixel 10101100 111001**10** 001100**01**

10 01 if changed to 11 01 ➔ Only 1 bit difference, so retain the secret bits and indicate no flip with 0 in Red LSB

Cover Pixel after embedding 1010110**0** 111001**11** 001100**01**

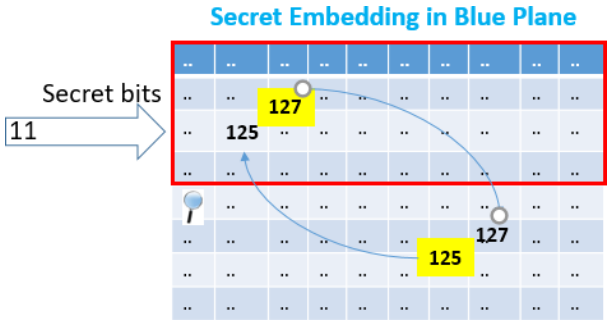

Figure 2: Histogram Preservation

This shows the scenario where the difference between secret and cover pixel is 3 bits:

Secret pixel - **0010**1100

Cover pixel - 10101101 111001**10** 001100**01**

10 01 if changed to 00 10 ➔ 3-bit differences, so flip the secret bits to "1101" so that the bit difference reduces to 1 and indicate the flip with 1 in Red LSB.

Cover Pixel after embedding 1010110**1** 111001**11** 001100**01**

4.6 Histogram Preservation

Color histogram is a graphical representation of the tonal distribution in a digital image. It plots the number of pixels for each tonal value. The horizontal axis of the graph represents the (0-255) tonal variations for red, green and blue planes, while the vertical axis represents the number of pixels in that particular tone. Histogram differences always exist between cover and stego images, even with flip, optimal seed or optimal position strategies applied. When modified secret bits (after the flipping) are embedded into the cover pixel, the original tone is replaced with a new tone because of change in LSBs of the red, green and blue plane. Histogram Attack is a common Steganalysis attack for LSB steganography. Anomalies in Histogram shape can be identified and steganalysts thereby detect steganography.

The original color tone is replaced with a new tone when secret bits are embedded into the cover pixel. To prevent staganalysis through histogram attack, this work introduces a technique to preserve the histogram shape by compensating for lost tones during the secret embedding process. This preservation technique can only be applied when part of the cover image remains unused even after embedding the secret bits and false image bits, which is possible only when the secret and false images are smaller than cover image. In this experimentation, a 512x512 pixel cover image and a 256x256 pixel secret image and a 189x182 pixel false image are used. To preserve the histogram, a pixel in the unused part of the cover image is identified that contains the new tone value after the secret image pixel embedding. This will result in the total count of the pixels with the original tone and the new tone remains unaffected. The tone swapping steps are performed separately in each color planes (R, G and B). In the example shown in Figure 2, when secret bits 11 need to be embedded onto a pixel with G value 125, it will be changed to 127. Thus, a pixel with a tone value of 125 is lost and a pixel with a tonal value 127 is gained. Many pixel changes in this manner would result in histogram shape distortion. To compensate for the lost 125 tone and the gained 125 tone due to secret embedding, the Histogram preservation algorithm searches in the unused part of the image for another pixel with value 127 and replaces that with 125. Thus, the total number of pixels with tone value 125 and 127 remains the same retaining the histogram shape.

5 EMBEDDING AND DE-EMBEDDING

5.1 Embedding Process

The embedding process as shown in Figure 3 starts with the user providing in cover image, false image and secret image files as inputs. The 16-character password that acts as the key for AES encryption of secret image is also provided as input by the user in the GUI. The ten separate 16-byte keys for the 10 rounds of AES are calculated from the initial 16-byte key provided by the user. Then the secret image is encrypted using AES encryption. The false image is embedded into the top part of the cover image using classical LSB substitution. The cover image and encrypted secret image are fed into the Optimal Seed search algorithm to find out the seed, for shuffling the remaining part of the cover image, that causes minimal bit distortions in cover image. The optimal seed found out is then fed into the Fisher Yates shuffling algorithm which shuffles the remaining part of the cover image on which the secret image is going to be embedded. Further, the shuffled cover image pixels and encrypted secret image pixels are fed into an Optimal Position Search algorithm to find out the optimal position in the shuffled cover image to start embedding the secret image. The

optimal position is the pixel index in the shuffled cover image, where beginning the embedding process results in the fewest bit distortions. The calculated optimal position pixel index is fed into the Modified LSB embedding algorithm which embeds the secret bits on to the shuffled cover image pixels sequentially starting from the optimal pixel index. The modified LSB substitution algorithm embeds secret bits onto cover pixels by performing a Pixel-by-Pixel distortion analysis and Histogram preservation. Finally, the shuffled cover image with embedded secret bits is unshuffled to obtain the Steg image that visually resembles the cover image.

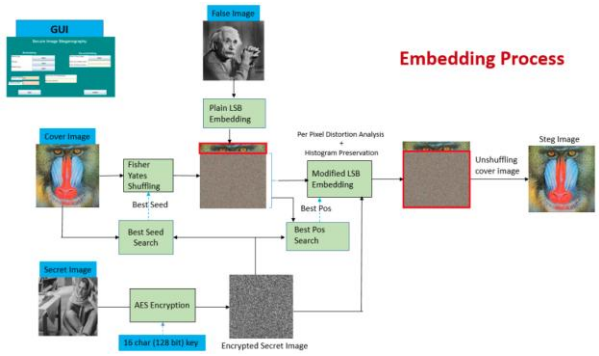

Figure 3: Embedding Process

5.2 Embedding Process Workflow

The Embedding Process workflow in Figure 5 shows the control flow of the java program to implement the diverse security and imperceptibility enhancement techniques for embedding secret image on cover image. The workflow starts with reading the cover image, false image and secret image. The 16-character key is also read from the GUI. The AES encryption of secret image is performed by only considering the 4 MSB bits of each pixel in the secret image. The optimal seed and optimal position algorithms are run. The optimal seed found out is used for Fisher yates shuffling algorithm on part of cover image. The false image is then embedded in the unshuffled topmost part of the cover image. The secret image is then embedded from the optimal pixel position in the shuffled cover image. During the embedding process, pixel-by-pixel distortion analysis and histogram preservation are carried out. Finally, the shuffled part of the cover image is unshuffled to obtain the stego image.

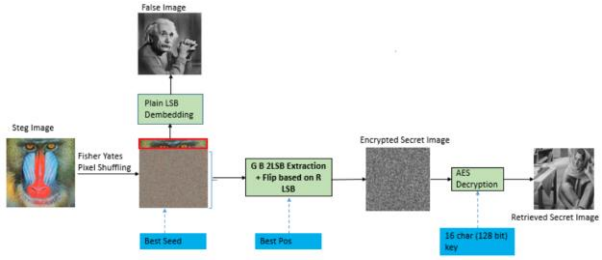

Figure 4: De-embedding Process

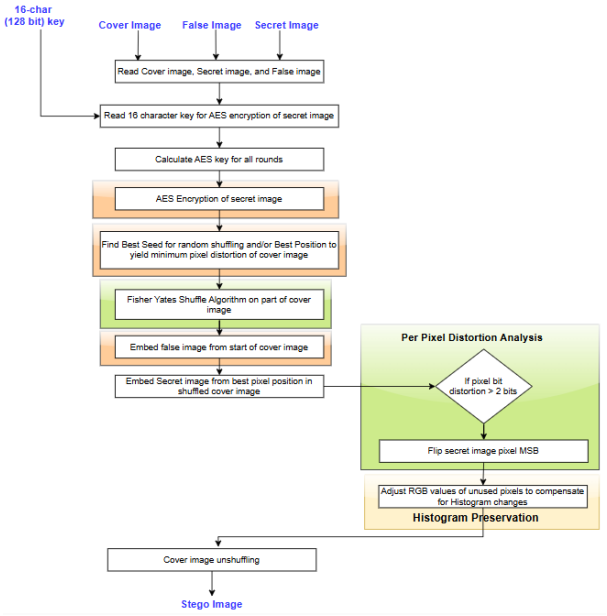

Figure 5: Embedding Process Design Workflow

5.3 De-Embedding Process

The de-embedding process as shown in Figure 4 starts with the user providing the stego image as input. In addition to stego image, the 16-character password used as the AES key, the optimal seed and optimal position used for shuffling and embedding the secret bits need to be passed to the receiver to be able to successfully extract the secret image. False image extraction does not need any of this secret information. The extraction of false image is carried out by extracting the 4 LSB bits of the 24-bit pixels in the top of cover image and then recreating another image with each of the 4 bits becoming the 4 MSB bits of the secret image pixels. The optimal seed information is then fed into the Fisher yates shuffling algorithm. The optimal position information is fed into LSB extraction algorithm which extracts the 2 LSBs from the green plane of the pixel and 2 LSBs from the red plane of the pixel, concatenate them and store them as the 4 MSB bits of the 8-bit pixels. This extraction for all the 256x256 pixels recovers the encrypted secret image. The encrypted secret image is then decrypted using AES decryption algorithm by passing the 16-character password as the 128-bit initial key for the AES. The retrieved secret image represents the actual secret image with a decent quality as shown in Figure 10.

5.4 Graphical User Interface

The graphical user interface of this novel secure steganography tool STEGASHIELD is as given in Figure 6. ‘Hide’ button press will start the embedding process and ‘Unhide’ button press will start the de- embedding process. For embedding, there are fields to provide the cover image, false image and secret image files. The JFileChooser API is used to prompt the user to choose a file or a directory. Additionally, the 16-character key for AES encryption which will be only shared with the receiver can be entered in the key field. After embedding, the optimal seed and optimal position found out by the embedding algorithm are displayed in the Chosen Seed and Embed position fields. For de-embedding, there are fields in the GUI to choose the stego image file, seed to be used for shuffling

and the position of the secret image in stego image.

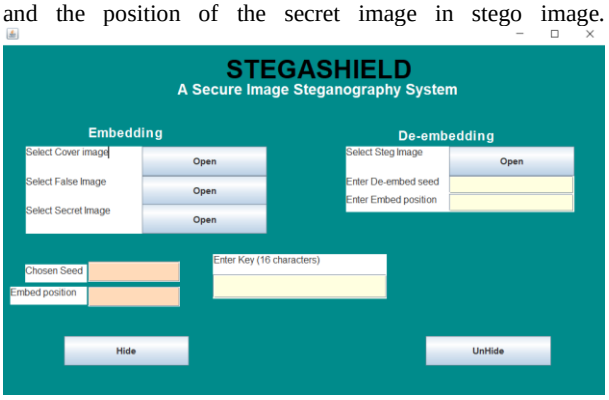

Figure 6: Secure Image Steganography GUI

6 RESULTS AND DISCUSSION

6.1 Image Quality Assessment Metrics

The goal of Image quality assessment is to quantitatively model the perception of image quality to human visual system (HVS). Full reference Image quality assessment (IQA) measures the relative quality of an image with respect to another image. Image steganography whose success depends on the imperceptibility of hidden data within the cover image, relies on the various full reference IQA metrics to assess the quality of the stego image with respect to the original cover image.

6.1.1 PSNR Test

Digital image steganography uses PSNR to evaluate the quality of stego images or the imperceptibility of steganography methods. PSNR (Peak Signal-to-noise ratio) test is commonly used in order to measure the difference between the two series of numbers, and it is based on the Mean Squared Error (MSE). MSE is computed by performing byte-by-byte comparisons of the cover and stego-image. PSNR is the difference between corresponding pixel values of the pre-algorithm to post-algorithm image and represents a measure of the peak error. The higher the PSNR, the lesser is the difference in quality of the image. A good stego image has a PSNR value of at least 40dB or greater and PSNR is calculated from the MSE as given below.

$$MSE = \frac{1}{N * M} \sum_{i=1}^{M} \sum_{j=1}^{N} (X_{ij} - Y_{ij})^2$$

$$PSNR = 10 \log \left(\frac{R^2}{MSE} \right) (dB)$$

R is the maximum fluctuation in the cover image data type. For example, if the cover image has a double-precision floating point data type, then R is 1. If it has an 8-bit unsigned integer data type, R is 255, etc. Table 1 compares the performance of PSNR and MSE for the different steganography techniques, and for different combinations of cover image and secret images.

6.1.2 Histogram comparison

An image histogram graphically represents the tonal distribution in a digital image, plotting pixel count against tonal values. For grayscale images, it spans from black (left) to white (right). In color images, separate histograms for red, green, and blue channels can be produced. In steganography, histogram similarity between cover and stego images indicates good quality steganography. However, histogram comparison is a statistical attack method that can reveal embedded messages by highlighting differences between cover and stego image histograms. For a 24-bit color image, 256 different intensities for each of the 3 channels (red, green, blue) are possible. Therefore, a histogram for each channel can be drawn separately, or an average histogram of all channels can be produced. Figure 7 is the histogram of the cover image, Baboon.jpg. From Figure 8, with Plain LSB embedding, the Histograms of cover and stego images look visually similar. But comparison of histograms shows that there is a difference in the histograms and thus it is prone to histogram attack.

From Figure 9, with Pixel-by-Pixel Distortion analysis and 'finding the optimal position to embed' approach, a higher PSNR is achieved but histogram deviation is visible between cover and stego images. Therefore, though visual imperceptibility has increased, the approach is still prone to histogram steganalysis. From Figure 10, combining Optimal Seed/Optimal Position approach and Pixel by Pixel distortion analysis with Histogram preservation, the Histograms of cover and stego images are exactly similar with zero difference. This shows that all these techniques combined are completely robust against histogram attack. From Figure 11, with an additional security of adding False image to the cover image, a deviation is seen in the blue plane histogram as the secret bits of false image are only embedded in the blue plane. But if an interim person extracts out the false image and further analyzes the remaining part of the cover image, they will find zero histogram difference. This might assure them that there is no more embedded data within the cover image and thus prevent the interim person from further steganalysis and retrieving the secret image.

7 CONCLUSION

This work presents a multi-level steganography system, called STEGASHIELD that brings together different optimization techniques in order to build a highly secure and imperceptible steganography system. Algorithms to find the optimal shuffling seed and optimal position in cover image that gives minimum bit distortions during the sequential embedding of secret image, help in raising the imperceptibility of the stego image. Pixel by Pixel bit distortion analysis and flipping of secret bits to reduce pixel distortions with more than 2-bit distortions gives a finer level of granularity for the control of imperceptibility in steganography systems. AES encryption of secret image and random embedding of the secret bits in the cover image pixels enhances the confidentiality of the secret image to a great extent, by ensuring that deciphering of the secret is almost impossible to the adversary even if he/she detects the presence of steganography. Adding an easily retrievable false image along with the secret image raises the security aspect of steganography to a higher level with the aim of misguiding the adversary. Another great contribution of this work is to prevent the Histogram attack by implementing a Histogram preservation technique to retain the histogram shape of cover image even after embedding the secret image. STEGASHIELD's high PSNR values and ability to maintain histogram integrity demonstrate its effectiveness in creating stego images that are both secure against various attacks and visually indistinguishable from their cover images. As digital communication continues to evolve, STEGASHIELD offers a promising foundation for developing even more sophisticated information hiding systems, potentially extending its applications beyond image steganography to other forms of digital media.

Table 1: PSNR Assessment

Cover and Secret images	Plain LSB	Histogram Adjustment	Pixel-by-Pixel Distortion Analysis + Optimal Position/Seed	Pixel-by-Pixel Distortion Analysis + Optimal Position/Seed + Histogram Adjustment	Pixel-by-Pixel Distortion Analysis + Optimal Seed + Histogram Adjustment + False Image Embedding
Cover Image – Baboon.jpg Secret image – Barbara.jpg	PSNR = 41.42 dB MSE = 4.68	PSNR = 47.682 dB MSE =1.109 Zero Histogram deviation	PSNR =52.39 dB MSE = 0.374	PSNR = 49.40 dB MSE = 0.745 Zero Histogram deviation	PSNR = 50.32 dB MSE = 0.604 Zero Histogram deviation in secret embedded area
Cover image – Peppers.jpg Secret image – Sailboat.jpg	PSNR = 41.24 dB MSE = 4.87	PSNR = 47.65 dB MSE = 1.116 Zero Histogram deviation	PSNR = 52.4 dB MSE = 0.374	PSNR = 49.368 dB MSE = 0.752 Zero Histogram deviation	PSNR = 50.12 dB MSE = 0.633 Zero Histogram deviation in secret embedded area
Cover image – lena.jpg Secret image-peppers.jpg	PSNR = 41.47 dB MSE = 4.63	PSNR = 47.67 dB MSE = 1.112 Zero Histogram deviation	PSNR = 52.415 dB MSE = 0.373	PSNR = 49.404 MSE = 0.746 Zero Histogram deviation	PSNR = 50.41 dB MSE = 0.591 Zero Histogram deviation in secret embedded area

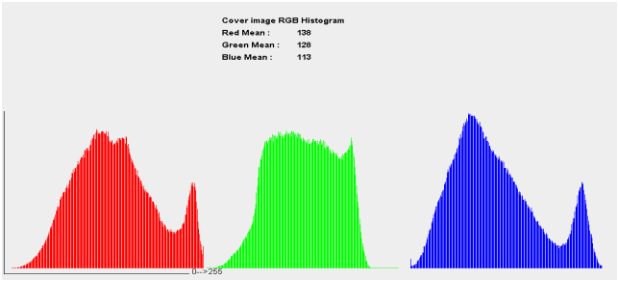

Figure 7: Histogram of Cover image – Baboon.jpg

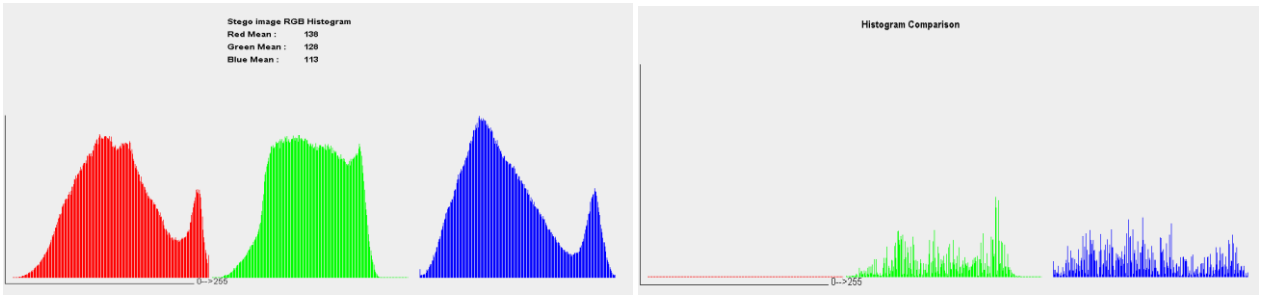

Figure 8: Histogram of Stego image using Plain LSB Embedding and Histogram Comparison of Cover and Stego images, PSNR = 50.687 dB

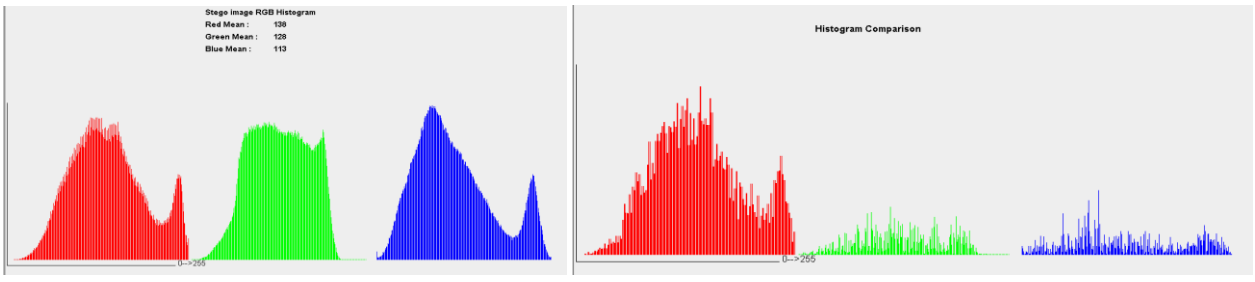

Figure 9: Histogram of Stego image using Plain LSB + Optimal Position Search + Pixel-by-Pixel Distortion analysis embedding, Histogram Comparison of Cover & Stego images, PSNR = 52.39 dB

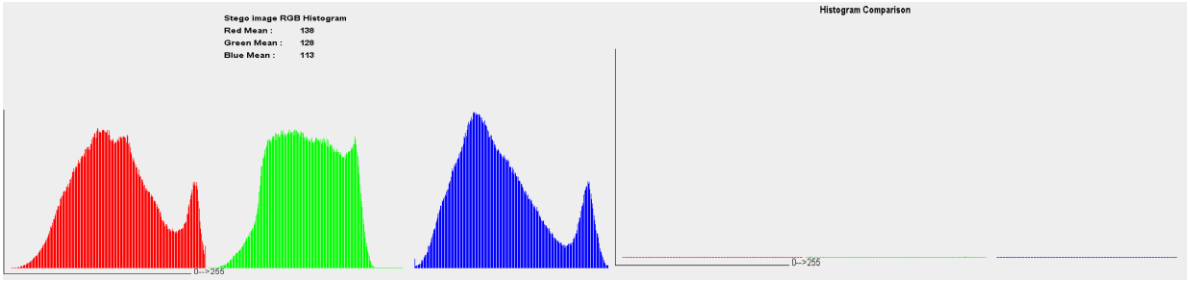

Figure 10: Histogram of Stego image using Plain LSB + Optimal Seed search + Pixel-by-Pixel Distortion Analysis + Histogram Preservation embedding, Histogram Comparison of Cover & Stego images, PSNR = 49.4 dB

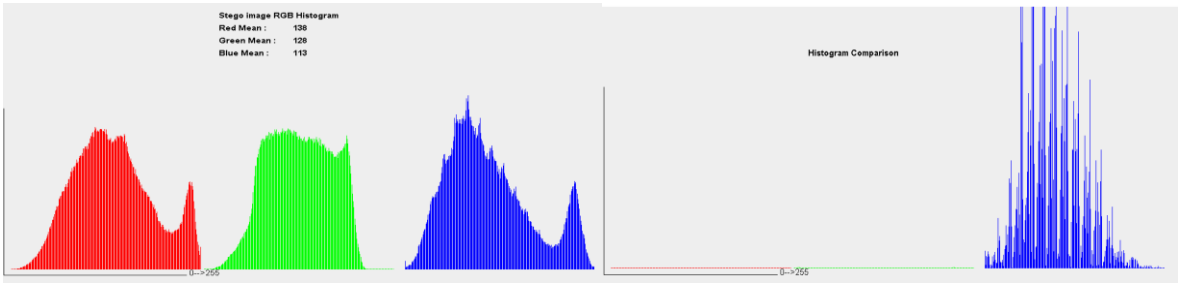

Figure 11: Histogram of Stego image using Plain LSB + Optimal Seed search + Pixel-by-Pixel Distortion Analysis + Histogram Preservation + False image embedding, Histogram Comparison of Cover & Stego images, PSNR = 49.39 dB

8 REFERENCES

[1] Datta, Biswajita, Upasana Mukherjee, and Samir Kumar Bandyopadhyay. "LSB layer independent robust steganography using binary addition." Procedia Computer Science 85 (2016): 425-432.

[2] Swain, Gandharba. "Very high capacity image steganography technique using quotient value differencing and LSB substitution." Arabian Journal for Science and Engineering 44.4 (2019): 2995-3004.

[3] Senarathne, A., and Kasun De Zoysa. "ILSB: indexing with least significant bit algorithm for effective data hiding." Int J Comput Appl 161.5 (2017): 28-42.

[4] Tutuncu, Kemal, and Baris Demirci. "Adaptive LSB Steganography Based on Chaos Theory and Random Distortion." Advances in Electrical and Computer Engineering 18.3 (2018): 15-22.

[5] Gowda, Shreyank N., and Sumit Sulakhe. "Block Based Least Significant Bit Algorithm For Image Steganography." Annual Int'l Conference on Intelligent Computing, Computer Science & Information Systems (ICCSIS-16). 2016.

[6] Mukherjee, Srilekha, and Goutam Sanyal. "A multi level image steganography methodology based on adaptive PMS and block based pixel swapping." Multimedia Tools and Applications (2019): 1-16.

[7] Subong, Ryan A., Arnel C. Fajardo, and Yoon Joong Kim. "LSB Rotation and Inversion Scoring Approach to Image Steganography." 2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE). IEEE, 2018.

[8] Astuti, Erna Zuni, et al. "Flipping the Message Bits to Increase Imperceptibility in the Least Significant Bit Image Steganography." Journal of Physics: Conference Series. Vol. 1201. No. 1. IOP Publishing, 2019.

[9] Susanto, Ajib, et al. "Dual Security Method for Digital Image using HBV Encryption and Least Significant Bit Steganography." Journal of Physics: Conference Series. Vol. 1201. No. 1. IOP Publishing, 2019.

[10] Abood, May H. "An efficient image cryptography using hash-LSB steganography with RC4 and pixel shuffling encryption algorithms." 2017 Annual Conference on New Trends in Information & Communications Technology Applications (NTICT). IEEE, 2017.

[11] Muhammad, Khan, et al. "CISSKA-LSB: color image steganography using stego key-directed adaptive LSB substitution method." Multimedia Tools and Applications 76.6 (2017): 8597-8626.

[12] Rustad, Supriadi, Abdul Syukur, and Pulung Nurtantio Andono. "Inverted LSB image steganography using adaptive pattern to improve imperceptibility." Journal of King Saud University-Computer and Information Sciences 34.6 (2022): 3559-3568.

[13] Ray, Atrayee Majumder, et al. "Hybrid Cryptography and Steganography Method to Provide Safe Data Transmission in IoT." International Conference on Data Analytics & Management. Singapore: Springer Nature Singapore, 2023.

[14] Ogiela, Marek R., and Katarzyna Koptyra. "False and multi-secret steganography in digital images." Soft Computing 19.11 (2015): 3331-3339.