Data Security Using Multimedia Steganography

Submitted in partial fulfillment of the requirements of the degree

BACHELOR OF ENGINEERING IN **COMPUTER ENGINEERING**

By

Vishakha Singh/ 57

Manasi Sharma/ 54

Anushka Shirode/ 56

Mentor

Prof. Sanjay Mirchandani

Vivekanand Education Society's Institute of Technology,

An Autonomous Institute affiliated to University of Mumbai

HAMC, Collector's Colony, Chembur,

Mumbai-400074

University of Mumbai (AY 2023-24)

CERTIFICATE

This is to certify that the Mini Project entitled **"Data Security Using Multimedia Steganography"** is a bonafide work of **Vishakha Singh(57), Manasi Sharma(54), Anushka Shirode(56)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **"Bachelor of Engineering"** in **"Computer Engineering".**

(Prof. Sanjay Mirchandani)
Mentor

(Dr. Nupur Giri)
Head of Department

(Dr. Mrs. J. M. Nair)
Principal

Mini Project Approval

This Mini Project entitled "Data Security Using Multimedia Steganography**"** by **Vishakha Singh (57), Manasi Sharma (54), Anushka Shirode (56)** is approved for the degree of **Bachelor of Engineering** in **Computer Engineering.**

Examiners

1……
(Internal Examiner Name & Sign)

2……
(External Examiner name & Sign)

Date: 21.10.2023

Place: Mumbai, India.

Contents

ABSTRACT

Steganography is the practice of concealing information within another message or physical object to avoid detection. Content concealed through steganography is sometimes encrypted before being hidden within another file format. If it isn't encrypted, then it may be processed in some way to make it harder to detect. Steganography can be used to hide virtually any type of digital content, including text, image, video, or audio content. The term 'steganography' comes from the Greek words 'steganos' (which means hidden or covered) and 'graphein' (which means writing)[1]. Steganography has been practiced in various forms for thousands of years to keep communications private. That hidden data is then extracted at its destination.The major objective of steganography is to connect privately in a completely undetectable aspect and to prevent drawing notice to the transmission of hidden information. Due to increased digital media on the Internet, data security and privacy protection issues have attracted the attention of data communication[2]. Data hiding in media, in images, video and audio, is of interest for the protection of copyrighted digital media, and to the government for information systems security and for covert (steganographic) communications[3]. The project deals with learning and implementing the various types of steganography available[4].

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to Prof. Sanjay Mirchandani for his unwavering support and guidance throughout our journey. His mentorship has been invaluable, and we are immensely grateful for his expertise and dedication in helping us in completing the project.

We also extend our sincere appreciation to the Department of Computer Engineering for their constant and invaluable guidance. Their collective wisdom and dedication to the field have been instrumental in shaping our knowledge and skills. Their support has been an essential pillar of our academic and professional growth, and we are thankful for their continuous encouragement.

List Of Abbreviations

Abbreviation	Full Form
LSB	Least Significant Bit
DWT	Discrete Wavelet Transform
DCT	Discrete Cosine Transform
CNN	Convolutional Neural Networks
RNN	Recurrent Neural Networks
GAN	Generative Adversarial Networks
SIFT	Scale-Invariant Feature Transform
GMM	Gaussian Mixture Model
EM	Expectation-Maximization
SRM	Spatial Rich Model

List of Figures

CHAPTER 1: INTRODUCTION

1.1 Introduction

The word Steganography is derived from two Greek words- ‘stegos’ meaning ‘to cover’ and ‘grayfia’, meaning ‘writing’, thus translating to ‘covered writing’, or ‘hidden writing’. Steganography is a method of hiding secret data, by embedding it into an audio, video, image, or text file. It is one of the methods employed to protect secret or sensitive data from malicious attacks. It's not about making the hidden data incomprehensible to others, it's about making it hard for others to believe it even exists[5].

Cryptography and steganography are both methods used to hide or protect secret data. However, they differ in the respect that cryptography makes the data unreadable, or hides the meaning of the data, while steganography hides the existence of the data[6].

When steganography is employed alone, it is secured by obscurity, which might result in the secret message being disclosed. Combining steganography and cryptography is the greatest way to disguise a message from adversaries while still protecting it in case it is detected.

Steganography can be used to hide virtually any type of digital content, including text, image, video, or audio content. That hidden data is then extracted at its destination[6]. Encryption can be added with stenography as an extra step in data hiding. Stenography can be used with images, text, audio, and video.

Its application can be seen in high authority management where the data needs to be known only to the concerned people and not to the third party.

Steganography is applicable to, but not limited to, the following areas[7].

1) Confidential communication and secret data storing
2) Protection of data alteration
3) Access control system for digital content distribution
4) Media Database systems

- Steganography can be used in confidential communication, to protect the content from unauthorized access or interception during transmission.
- In intelligence and military applications, steganography can be employed to covertly exchange information between agents or operatives without arousing suspicion or detection.

- Watermarking is a form of steganography used in the digital media industry to embed copyright information or ownership details into images, audio, or video files. This helps identify the rightful owner and prevent unauthorized use or distribution.
- Preventing data tampering, by embedding checksums or digital signatures within digital files, steganography can help verify the authenticity and integrity of the data. Any alterations or tampering with the file would be evident if the checksum or signature does not match.

Image Steganography:

As the name suggests, Image Steganography refers to the process of hiding data within an image file. The image selected for this purpose is called the cover image and the image obtained after steganography is called the stego image.

An image is represented as an N*M (in case of grayscale images) or N*M*3 (in case of color images) matrix in memory, with each entry representing the intensity value of a pixel. In image steganography, a message is embedded into an image by altering the values of some pixels, which are chosen by an encryption algorithm. The recipient of the image must be aware of the same algorithm in order to know which pixels he or she must select to extract the message[6].

Audio Steganography:

Embedding secret messages in digital sound is a more complex process. Varieties of techniques for embedding information in digital audio have been established[8]. Audio steganography is an approach to hiding information within an audio signal. This method is to embed “secret” text, images, and audio inside a “public” sound file.

A famous audio steganography approach is the LSB (Least Significant Bit) algorithm. The least significant bit of the cover signal can be used to hide the message. This is a very simple method. LSB acquires advantages like low computational load and simplicity yet drawbacks in security[9].

Video Steganography:

The video steganography approach enables hiding chunks of secret information inside video sequences. The features of video sequences including high capacity as well as complex structure make them preferable for choosing as cover media over other media.

The cover video considered is either in the raw domain or compressed domain. Raw domain videos are further classified into spatial domain and transform domain. Least Significant Bits (LSB) substitution and other significant methods are included in the spatial domain. Discrete Wavelet Transform (DWT) and Discrete Cosine Transform (DCT) are the extensively used transformation methods to convert cover videos into the transform domain. After converting the video into the transform domain, embedding of the secret information is undertaken. In the compressed domain, video steganography uses compressed cover videos. The embedding happens during or after the compression of the videos. Motion Vectors, intra-prediction modes, entropy coding modules, and DCT/DST techniques are the extensively used video steganography methods in the compressed domain[10].

1.2 Motivation

The motivation for a steganography project is to hide or embed information within other data, whether for the purposes of data security, covert communication, authentication, creative expression, or digital watermarking. Steganography can enhance data protection, facilitate covert messaging, or prove ownership of digital content. It's employed in various fields, from information security and cybersecurity to art and digital forensics, depending on the specific goals and context of the project.

1.3 Problem Statement

The problem statement in steganography includes the task is to hide a message in another piece of media(e.g., an image, audio file, etc.) without leaving any visible signs that the message is there. This is done using Steganography tools.
Steganalysis Algorithms are then applied in order to detect steganography.

CHAPTER 2: LITERATURE SURVEY

2.1 Survey of Existing System

Paper Title: "Deep Learning-Based Steganalysis: A Comprehensive Survey" (2021)
Authors: Muhammad Usama, Muhammad Imran, Muhammad Asif, et al.
Methodology: This paper provides a comprehensive survey of deep learning-based steganalysis techniques. The authors review various deep learning architectures, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and generative adversarial networks (GANs). They discuss the use of different datasets and evaluation metrics for steganalysis performance assessment.
Advantages: The paper offers a comprehensive overview of deep learning-based steganalysis techniques, covering multiple domains such as image, audio, and video steganography. It provides insights into the strengths and limitations of different deep learning architectures and highlights the challenges and future directions in the field.
Disadvantages: The survey focuses primarily on deep learning techniques, potentially overlooking other steganalysis approaches. Additionally, the paper does not provide a detailed analysis of specific algorithms or experimental results.

Paper Title: "Adversarial Attacks and Defenses in Steganography: A Survey" (2021)
Authors: Xingming Sun, Xinpeng Zhang, Xiangui Kang, et al.
Methodology: This paper presents a survey on adversarial attacks and defenses in steganography. The authors discuss various adversarial attack strategies, including universal adversarial perturbations and targeted attacks, and review defense mechanisms such as adversarial training and defensive distillation.
Advantages: The paper provides a comprehensive overview of adversarial attacks and defenses in steganography, highlighting the vulnerabilities of steganographic systems and the countermeasures to mitigate these attacks. It covers a wide range of attack and defense techniques, providing insights into their effectiveness and limitations.
Disadvantages: The survey focuses primarily on adversarial attacks and defenses, potentially overlooking other aspects of steganography. It does not provide a detailed analysis of specific algorithms or experimental results.

Paper Title: "Deep Learning for Steganography and Steganalysis: A Survey" (2020)

Authors: Zhihao Chen, Xinpeng Zhang, Xiangui Kang, et al.
Methodology: This survey paper focuses on the application of deep learning techniques in both steganography and steganalysis. The authors review various deep learning models used for steganography, such as GANs and CNNs, and discuss deep learning-based steganalysis methods, including feature extraction and classification approaches.
Advantages: The paper provides a comprehensive overview of deep learning techniques in steganography and steganalysis. It covers both the embedding and detection aspects of steganography, highlighting the advancements and challenges in the field. The survey offers insights into the strengths and limitations of different deep learning models.
Disadvantages: The survey primarily focuses on deep learning techniques, potentially overlooking other steganography and steganalysis approaches. It does not provide a detailed analysis of specific algorithms or experimental results.

2.2 Research Gap

The existing systems focus less on an exploration of available stegano tools. For images, tools like OpenPuff, OpenStego, and Steghide which are available on Kali Linux can be helpful in converting a normal image into a stegged image.
These images can be used to train algorithms to detect if an image has hidden information. Algorithms have a threshold or a grayscale value that is pre-defined. The images are scanned and these distortions are observed and printed. This is a valuable technique for beginners and can prove to be a stepping stone for many in steganography.

2.3 Mini Project Contribution

The project uses various pre-available algorithms. The code is written in python where an attempt has been made to see the detection of stegged image.

CHAPTER 3: PROPOSED SYSTEM

3.1 Introduction

Steghide:-

Steghide is a steganography program that is able to hide data in various kinds of image- and audio files. The color- respectively sample-frequencies are not changed thus making the embedding resistant against first-order statistical tests.

Features include the compression of the embedded data, encryption of the embedded data, and automatic integrity checking using a checksum. The JPEG, BMP, WAV, and AU file formats are supported for use as cover files. There are no restrictions on the format of the secret data.

Steghide uses a graph-theoretic approach to steganography. You do not need to know anything about graph theory to use Steghide and you can safely skip the rest of this paragraph if you are not interested in the technical details. The embedding algorithm roughly works as follows: At first, the secret data is compressed and encrypted. Then a sequence of positions of pixels in the cover file is created based on a pseudo-random number generator initialized with the passphrase (the secret data will be embedded in the pixels at these positions). Of these positions, those that do not need to be changed (because they already contain the correct value by chance) are sorted out.

Then a graph-theoretic matching algorithm finds pairs of positions such that exchanging their values has the effect of embedding the corresponding part of the secret data. If the algorithm cannot find any more such pairs all exchanges are actually performed. The pixels at the remaining positions (the positions that are not part of such a pair) are also modified to contain the embedded data (but this is done by overwriting them, not by exchanging them with other pixels). The fact that (most of) the embedding is done by exchanging pixel values implies that the first-order statistics (i.e. the number of times a color occurs in the picture) are not changed. For audio files, the algorithm is the same, except that audio samples are used instead of pixels.

Openpuff:-

OpenPuff Steganography and Watermarking, sometimes abbreviated OpenPuff or Puff, is a free steganography tool for Microsoft Windows created by Cosimo Oliboni and still

maintained as independent software. The program is notable for being the first steganography tool (version 1.01 released in December 2004) that:

- lets users hide data in more than a single carrier file. When hidden data are split among a set of carrier files you get a carrier chain, with no enforced hidden data theoretical size limit (256MB, 512MB, ... depending only on the implementation)
- implements 3 layers of hidden data obfuscation (cryptography, whitening and encoding)
- extends deniable cryptography into deniable steganography

OpenPuff is used primarily for anonymous asynchronous data sharing:

- the sender hides a hidden stream inside some public available carrier files (password + carrier files + carrier order are the secret key)
- the receiver unhides the hidden stream knowing the secret key

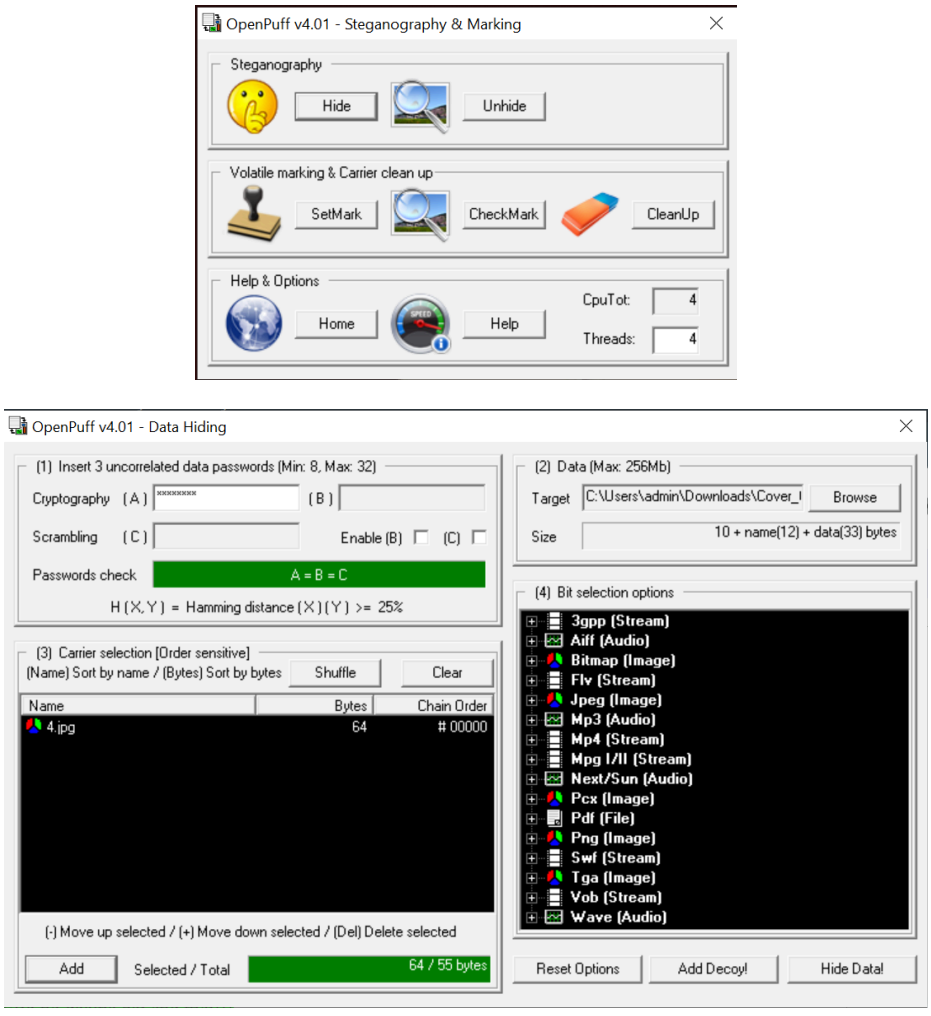


Fig 1: OpenPuff UI

OpenStego:-

OpenStego is a software application designed for the purpose of concealing confidential data within digital media files, such as images or audio files. It utilizes a technique known as steganography, which enables the covert embedding of information within another file, without arousing suspicion.

OpenStego offers several operations that can be performed:

1. Embedding: This operation involves the discreet integration of secret information into a chosen cover file. The cover file, typically a digital media file like an image, is selected, and the desired confidential data is merged with it. The resulting file appears unchanged to casual observers.

2. Extraction: Extraction refers to the process of retrieving the concealed information from a stego file. If a stego file has been created using OpenStego, the extraction operation can be employed to reveal the hidden data.

3. Watermarking: OpenStego provides the capability to add watermarks to digital media files. Watermarks are commonly used to safeguard the copyright of images or indicate ownership. By utilizing OpenStego, a watermark can be embedded into an image, making it challenging for unauthorized individuals to remove or alter it without detection.

4. Cryptography: OpenStego supports encryption and decryption of secret information prior to its embedding within the cover file. This additional layer of security ensures that only authorized individuals can access the concealed data.

In essence, OpenStego is a sophisticated tool that facilitates the covert embedding of confidential information within digital files, such as images or audio. It also enables the extraction of hidden data from such files. Furthermore, OpenStego offers the ability to add watermarks for copyright protection and supports encryption for enhanced security of concealed information.

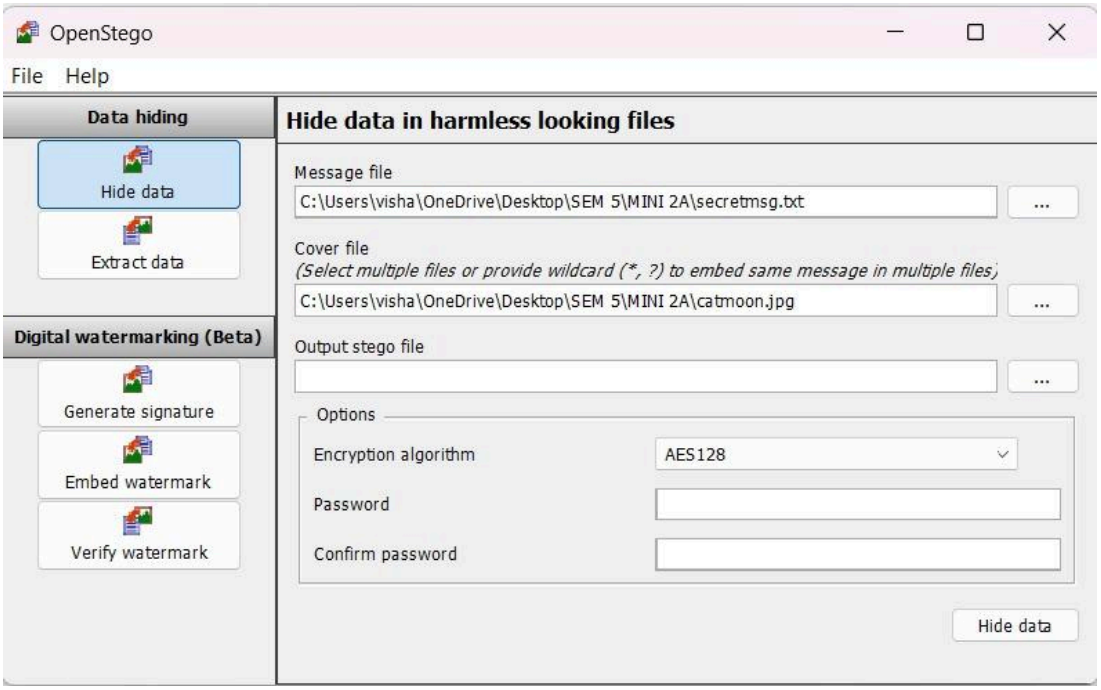


Fig 2: OpenStego UI

Stegano Library (Python):-

The Stegano library can be utilized for LSB (Least Significant Bit) detection, which involves identifying and extracting hidden information that has been concealed using the LSB steganography technique. LSB steganography involves replacing the least significant bits of pixel values in an image with secret data, making it imperceptible to the human eye.

Advantages of using the Stegano library for LSB detection include:

1. Ease of use: The Stegano library provides a user-friendly interface and straightforward functions, making it accessible for individuals with varying levels of technical expertise.

2. Efficient detection: The library offers efficient algorithms and methods specifically designed for LSB detection, enabling accurate and reliable identification of hidden data.

3. Versatility: The Stegano library supports various image formats, allowing for LSB detection in a wide range of image files.

Uses of the Stegano library for LSB detection:

1. Security analysis: The library can be employed to analyze the security of digital images by detecting any potential hidden information. This is particularly useful in forensic investigations or when assessing the integrity of sensitive images.

2. Digital watermarking verification: LSB detection can be utilized to verify the presence and integrity of digital watermarks embedded within images. This ensures that the watermarked content has not been tampered with.

Drawbacks of the Stegano library for LSB detection:

1. Limited to LSB steganography: The Stegano library is specifically designed for LSB detection, which means it may not be effective in detecting other steganographic techniques that do not rely on LSB manipulation.

2. Susceptibility to advanced techniques: While the Stegano library is effective in detecting basic LSB steganography, it may struggle to identify more advanced and sophisticated steganographic methods that employ encryption or other complex techniques.

In summary, the Stegano library offers a user-friendly approach to LSB detection, allowing for the identification and extraction of hidden information in digital images. It is advantageous due to its ease of use, efficiency, and versatility. It finds applications in security analysis and digital watermarking verification. However, it is limited to LSB steganography and may not be effective against advanced steganographic techniques.

Shannon Entropy:-
The term "Shannon entropy" refers to a concept in information theory that measures the uncertainty or information content of a random variable or a set of data. It is named after Claude Shannon, who introduced the concept in his 1948 paper "A Mathematical Theory of Communication," which is considered a foundational work in the field of information theory.
Shannon introduced the concept of entropy as a measure of the uncertainty or information

content associated with a random variable. He formulated the entropy of a discrete random variable, which is now known as Shannon entropy, as:

$H(X) = -\Sigma\, p(x) * \log2(p(x))$

Where:

H(X) is the entropy of the random variable X.
p(x) is the probability mass function of the random variable X.
Σ denotes the sum over all possible values of X.
log2 represents the base-2 logarithm.

Shannon entropy has since found applications in various fields, including information theory, statistics, machine learning, and data analysis. It is commonly used to quantify the uncertainty or randomness in data and has been applied in areas such as data compression, cryptography, and measuring information gain in decision trees, among others.

SIFT (Scale-Invariant Feature Transform):-
SIFT (Scale-Invariant Feature Transform) is a computer vision algorithm that can be utilized for steganography detection. While SIFT is primarily designed for feature extraction and matching in images, it can also be applied to identify hidden steganographic content within images.

Advantages of using SIFT for steganography detection include:

1. Scale-invariance: SIFT is capable of detecting features in images regardless of their scale or size. This makes it effective in identifying steganographic content that may have been resized or scaled.

2. Robustness to transformations: SIFT is resilient to various image transformations, such as rotation, translation, and changes in lighting conditions. This enables it to detect steganographic content even if it has undergone such transformations.

3. Feature-based approach: SIFT focuses on extracting distinctive features from images, which can be used to identify patterns or anomalies associated with steganography. This approach enhances the accuracy and reliability of steganography detection.

Uses of SIFT for steganography detection:

1. Digital forensics: SIFT can be employed in digital forensics investigations to identify hidden steganographic content within images. This is particularly useful in cases involving the concealment of sensitive or illicit information.

2. Content integrity verification: SIFT can be used to verify the integrity of images by detecting any potential steganographic alterations. This is valuable in situations where the authenticity and trustworthiness of images need to be ensured.

Drawbacks of using SIFT for steganography detection:

1. Computational complexity: SIFT involves computationally intensive operations, which can make it time-consuming when applied to large datasets or high-resolution images.

2. Limited to image-based steganography: SIFT is primarily designed for image analysis, so it may not be suitable for detecting steganography in other types of media, such as audio or video.

What makes SIFT unique:

SIFT's uniqueness lies in its ability to extract and match scale-invariant features in images, making it highly effective in scenarios where the steganographic content may have undergone scaling, rotation, or other transformations. Its robustness and feature-based approach contribute to its accuracy and reliability in steganography detection, making it a valuable tool in digital forensics and content integrity verification.

Gaussian Mixture Model (GMM) with Expectation-Maximization (EM):-
The Gaussian Mixture Model (GMM) with Expectation-Maximization (EM) algorithm can be employed for steganography detection. GMM is a statistical model that represents a

probability distribution as a combination of multiple Gaussian distributions. EM is an iterative algorithm used to estimate the parameters of the GMM. Together, they can be utilized to identify hidden steganographic content within digital media.

Advantages of using GMM with EM for steganography detection include:

1. Statistical modeling: GMM provides a robust statistical framework for analyzing the distribution of pixel values in digital media. This allows for the detection of anomalies or deviations that may indicate the presence of steganographic content.

2. Flexibility: GMM can adapt to various types of steganography techniques, making it applicable to a wide range of steganographic algorithms. It can capture both simple and complex statistical patterns associated with hidden information.

3. Unsupervised learning: GMM with EM does not require labeled training data for steganography detection. It can automatically learn and estimate the parameters of the model from the given data, making it suitable for unsupervised detection scenarios.

Uses of GMM with EM for steganography detection:

1. Digital forensics: GMM with EM can be utilized in digital forensics investigations to identify steganographic content within images or other digital media. It helps in uncovering hidden information that may be relevant to legal or security investigations.

2. Content authenticity verification: GMM with EM can be used to verify the authenticity of digital media by detecting any potential steganographic alterations. This is valuable in situations where the integrity and trustworthiness of the content need to be ensured.

Drawbacks of using GMM with EM for steganography detection:

1. Computational complexity: GMM with EM involves iterative computations, which can be computationally expensive, especially when applied to large datasets or high-dimensional media.

2. Sensitivity to noise: GMM with EM may be sensitive to noise or outliers in the data, which can affect the accuracy of steganography detection. Preprocessing steps may be required to mitigate this issue.

What makes GMM with EM unique:

GMM with EM is unique due to its ability to model complex statistical patterns in digital media, making it adaptable to various steganography techniques. Its unsupervised learning approach allows for automatic parameter estimation without the need for labeled training data. This flexibility and adaptability make GMM with EM a powerful tool in digital forensics and content authenticity verification, enabling the detection of hidden steganographic content.

Spatial Rich Model (SRM):-
SRM (Spatial Rich Model) can be utilized for steganography detection. SRM is a statistical model that analyzes the spatial distribution of pixel values in an image to identify potential steganographic modifications. It compares the statistical properties of the image with those of a reference model to detect any deviations that may indicate the presence of hidden information.

Advantages of using SRM for steganography detection include:

1. Statistical analysis: SRM employs statistical analysis to identify changes in the spatial distribution of pixel values caused by steganographic modifications. This allows for the detection of subtle alterations that may be imperceptible to the human eye.

2. Robustness: SRM is robust against common steganographic techniques that manipulate pixel values, such as LSB (Least Significant Bit) embedding. It can detect steganographic content even if it has undergone compression or other image processing operations.

3. Reference model comparison: SRM compares the statistical properties of an image with those of a reference model, enabling the detection of deviations that may indicate the presence of hidden information. This approach enhances the accuracy and reliability of steganography detection.

Uses of SRM for steganography detection:

1. Digital forensics: SRM can be employed in digital forensics investigations to identify steganographic content within images. It helps in uncovering hidden information that may be relevant to legal or security investigations.

2. Content integrity verification: SRM can be used to verify the integrity of images by detecting any potential steganographic alterations. This is valuable in situations where the authenticity and trustworthiness of images need to be ensured.

Drawbacks of using SRM for steganography detection:

1. Limited to spatial domain steganography: SRM is primarily designed for detecting steganography techniques that modify the spatial distribution of pixel values. It may not be effective in detecting steganography methods that operate in other domains, such as frequency or transform domains.

2. Sensitivity to image content: SRM's effectiveness may vary depending on the content of the image being analyzed. Certain types of images or specific steganography techniques may pose challenges for accurate detection.

What makes SRM unique:

SRM is unique due to its focus on analyzing the spatial distribution of pixel values in an image for steganography detection. Its statistical analysis and comparison with a reference model enable the detection of subtle alterations caused by steganographic modifications. SRM's robustness, particularly against LSB embedding and compression, makes it a valuable tool in digital forensics and content integrity verification. However, its effectiveness may be limited to spatial domain steganography and can be influenced by the content of the image being analyzed.

3.2 Architectural Framework / Conceptual Design

SIFT:

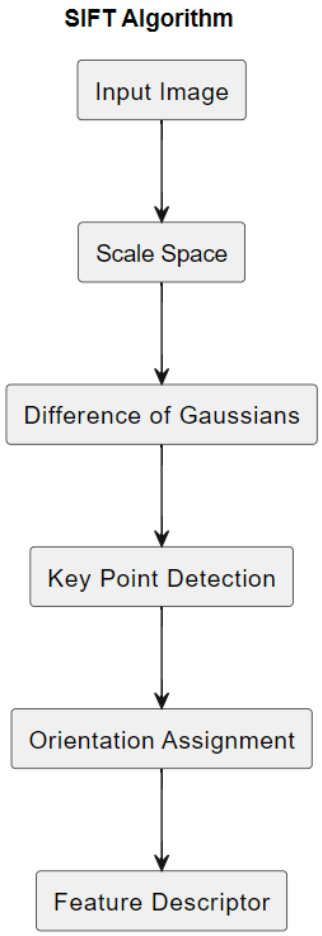


Fig 3: Working of SIFT algorithm

Shannon Entropy:

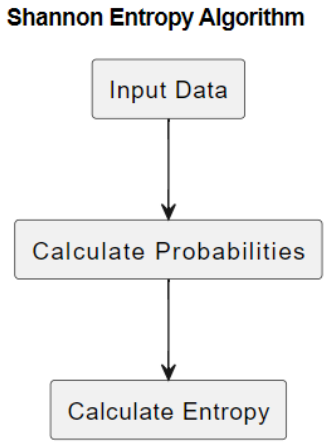


Fig 4: Working of Shannon-Entropy algorithm

GMM:

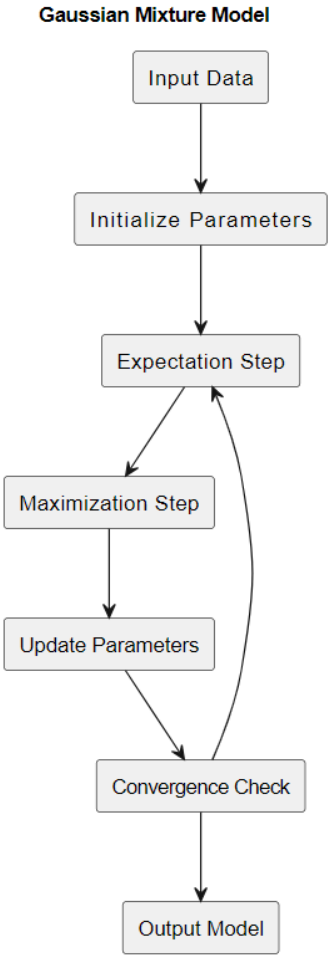


Fig 5: Working of GMM algorithm

SRM:

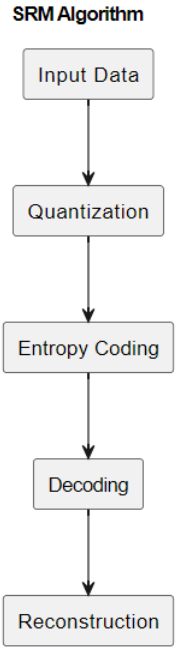


Fig 6: Working of SRM algorithm

3.3 Algorithm and Process Design

1. The Stegano library of Python has been implemented to provide a user-friendly and accessible tool for steganography operations. It allows users to embed and extract secret information within digital media files, perform watermarking, and apply cryptography techniques. The implementation of the Stegano library aims to simplify the process of steganography for users and provide them with a reliable and efficient toolset.

2. SIFT (Scale-Invariant Feature Transform) has been implemented for steganography detection due to its ability to extract and match scale-invariant features in images. This makes it effective in identifying hidden patterns or anomalies associated with steganographic content. By analyzing distinctive features, SIFT can accurately detect the presence of steganography even when the image undergoes scaling, rotation, or other transformations.

3. Shannon entropy has been implemented for steganography detection because it provides a statistical approach to measure the randomness or information content in a signal. By calculating the entropy of pixel values in an image, it can identify deviations from the expected distribution. Steganographic modifications often introduce additional information or alter the statistical properties of the image, leading to a change in entropy. Thus, Shannon

entropy is effective in detecting such alterations and indicating the presence of hidden information.

4. A Gaussian Mixture Model (GMM) with Expectation-Maximization (EM) has been implemented for steganography detection due to its statistical modeling capabilities. GMM represents a probability distribution as a combination of multiple Gaussian distributions, and EM is an iterative algorithm used to estimate the parameters of the GMM. This approach allows for the analysis of the distribution of pixel values in digital media to detect steganographic modifications. The implementation of GMM with EM provides a flexible and adaptable method for steganography detection.

5. SRM (Spatial Rich Model) has been implemented for steganography detection because it focuses on analyzing the spatial distribution of pixel values in an image. By comparing the statistical properties of the image with those of a reference model, SRM can detect deviations that may indicate the presence of hidden information. The implementation of SRM provides a unique approach to steganography detection by analyzing the spatial characteristics of the image.

In addition to the mentioned algorithms, other algorithms that can be used for steganography detection include:

- Machine learning algorithms: Various machine learning techniques, such as support vector machines (SVM), random forests, or deep learning models, can be trained on labeled datasets to detect steganographic content. These algorithms can learn patterns and features associated with steganography and provide accurate detection results.

- Frequency domain analysis: Algorithms based on frequency domain analysis, such as Discrete Fourier Transform (DFT) or Discrete Wavelet Transform (DWT), can be used to detect steganography by analyzing the frequency components of the image. Changes in the frequency spectrum caused by steganographic modifications can be identified using these algorithms.

- Statistical analysis: Other statistical techniques, such as histogram analysis, correlation analysis, or higher-order statistics, can be employed for steganography detection. These

methods focus on analyzing the statistical properties of the image to identify deviations or anomalies introduced by steganography.

The choice of algorithm for steganography detection depends on the specific requirements, characteristics of the steganographic content, and the available resources. It is important to select robust, efficient, and adaptable algorithms for different types of steganography techniques.

3.4 Methodology Applied

LSB(Least Significant Bit)

This code uses the "stegano" library, particularly the "lsb" (Least Significant Bit) algorithm, to detect steganography in an image. The LSB algorithm is a common technique for hiding information within an image by replacing the least significant bits of the pixel values with the hidden data. Here's how the code works:

- Import the necessary functions from the "stegano" library:

- from stegano import lsb: This line imports the "lsb" module from the "stegano" library, which contains functionality for LSB steganography.
- Define a function called detect_steganography(image_path):

- This function takes the path to an image file as its parameter.
  Inside the detect_steganography function:

- It tries to reveal any hidden data from the image using the LSB steganography method.

- cover_image = lsb.reveal(image_path): This line attempts to reveal any hidden data in the specified image using the LSB method. If there's hidden information in the image, it will be extracted.
  If the reveal operation is successful (i.e., it doesn't raise an exception), the function returns True, indicating that steganography is detected.

- If the reveal operation fails (e.g., no hidden data is found or an error occurs), the function returns False, indicating that no steganography is detected.

- Set the image_path variable to the path of the image file you want to analyze. In this code, it's set to "/content/steged_image1.jpg."

- Call the detect_steganography function with the provided image path and store the result in the is_stego variable. This variable will be True if steganography is detected and False otherwise.

- Print the result:
  If is_stego is True, it prints "Steganography detected in the image."
  If is_stego is False, it prints "No steganography detected in the image."

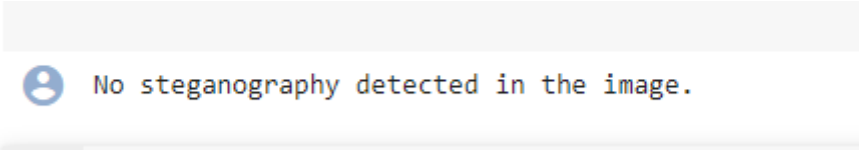


Fig 7: Output Obtained Using 'Stegano Library'

Shannon entropy

This code is designed to detect steganography in an image using a basic statistical analysis approach based on the Shannon entropy of the image's grayscale histogram. It analyzes the distribution of pixel intensities in the image and uses Shannon entropy to determine if the image shows characteristics of steganography. However, it's important to note that this code is not specifically tailored to detect a particular steganography technique like the GFR (Global Frequency Representation) method. Instead, it looks for anomalies in the image's grayscale histogram that might suggest the presence of hidden information.

Working :

- Import the necessary libraries:
  cv2 (OpenCV) is imported to handle image processing and analysis.
  numpy is imported to work with numerical data.
  Define a function called detect_steganography(image_path):

- This function takes the path to an image file as its parameter.
  Inside the detect_steganography function:

It loads the image specified by image_path using OpenCV.
It converts the loaded image to grayscale to simplify the analysis

- Calculate the histogram of pixel intensities:
  The code uses cv2.calcHist to calculate the histogram of pixel intensities for the grayscale image. This histogram represents the distribution of pixel values in the image.

- Normalize the histogram:
  The histogram is normalized by dividing it by the sum of all its values. This ensures that the histogram represents a probability distribution of pixel intensities.

- Calculate the Shannon entropy of the histogram:
  The Shannon entropy is computed as a measure of information content in the grayscale image. It quantifies the uncertainty or randomness in the distribution of pixel intensities.

- Set a threshold for steganography detection:
  A threshold value of 7.0 is defined. If the calculated Shannon entropy is greater than this threshold, the code will consider it an indication of potential steganography.

- Compare the entropy with the threshold:
  If the calculated entropy is higher than the threshold, the function returns True, indicating that steganography is detected in the image.
  If the entropy is lower than or equal to the threshold, the function returns False, indicating that no steganography is detected.
  Set the image_path variable to the path of the image file you want to analyze, e.g., "barbiej.jpg."

- Call the detect_steganography function with the provided image path and store the result in the is_stego variable. This variable will be True if steganography is detected and False otherwise.

- Print the result:
  If is_stego is True, it prints "Steganography detected in the image."
  If is_stego is False, it prints "No steganography detected in the image."

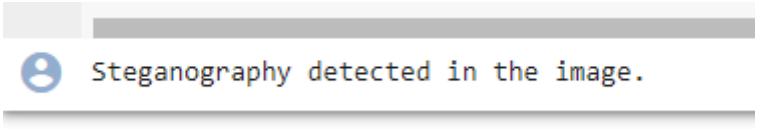


Fig 8: Output Obtained Using 'Shannon-Entropy'

In summary, this code uses Shannon entropy analysis of the grayscale histogram to detect potential steganography in an image. It looks for irregularities in the distribution of pixel intensities that may suggest the presence of hidden information.

Gaussian Mixture Model (GMM)

This code is designed to detect steganography in an image using a Gaussian Mixture Model (GMM) with Expectation-Maximization (EM). It analyzes the grayscale version of the image by fitting a GMM to the pixel intensity values and then calculating the difference between the means of the Gaussian components. The difference is compared to a threshold to determine if steganography is present in the image.

Working:

- Import the necessary libraries:
  cv2 (OpenCV) is imported for image processing and analysis.
  numpy is imported for numerical operations.
  sklearn.mixture imports the GaussianMixture class for fitting the GMM with EM.
  Define a function called detect_steganography(image_path):

- This function takes the path to an image file as its parameter.
- Inside the detect_steganography function:
  It loads the image specified by image_path using OpenCV.
  The loaded image is converted to grayscale to simplify the analysis.

- Reshape the grayscale image to a 1D array:
  The grayscale image is reshaped into a 1D array using reshape(-1, 1). This is done to prepare the data for fitting a GMM.

- Fit a Gaussian Mixture Model with Expectation-Maximization:
  A GMM with 2 components (i.e., two Gaussian distributions) is created using GaussianMixture(n_components=2).
  The GMM is then fitted to the reshaped grayscale image data using gmm.fit(reshaped_image).

- Get the weights and means of the Gaussian components:
  The code extracts the weights and means of the Gaussian components that the GMM has estimated.

- Calculate the difference between the means:
  The absolute difference between the means of the two Gaussian components is computed. This difference represents a measure of how much the pixel intensities differ between two distinct clusters in the image.
- Set a threshold for steganography detection:
  A threshold value of 10.0 is defined. This threshold will be used to compare against the calculated mean difference.

- Compare the mean difference with the threshold:
  If the calculated mean difference is greater than the threshold, the function returns True, indicating that steganography is detected in the image.
  If the mean difference is less than or equal to the threshold, the function returns False, indicating that no steganography is detected.
  Set the image_path variable to the path of the image file you want to analyze.

- Call the detect_steganography function with the provided image path and store the result in the is_stego variable. This variable will be True if steganography is detected and False otherwise.

- Print the result:
  If is_stego is True, it prints "Steganography detected in the image."
  If is_stego is False, it prints "No steganography detected in the image."

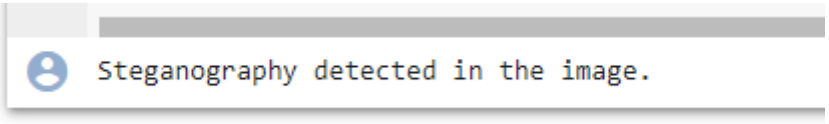


Fig 9: Output Obtained Using 'Gaussian Mixture Model with Expectation-Maximization'

In summary, this code uses a Gaussian Mixture Model (GMM) with Expectation-Maximization (EM) to analyze the grayscale image and detect steganography by looking at the difference between the means of the Gaussian components. If this difference exceeds a specified threshold, steganography is detected in the image.

Scale-Invariant Feature Transform (SIFT)

This code uses the Scale-Invariant Feature Transform (SIFT) algorithm to detect keypoints and compute descriptors in a grayscale image. SIFT is not a steganography algorithm but rather a feature detection and description algorithm widely used in computer vision for various tasks, including object recognition, image matching, and keypoint-based image analysis.

Working:

- Import the necessary libraries:
  cv2 (OpenCV) is imported for image processing and analysis.
  numpy is imported for numerical operations.

- Define a function called detect_steganography(image_path):
  This function takes the path to an image file as its parameter.
  Inside the detect_steganography function:
- It loads the image specified by image_path using OpenCV in grayscale mode, meaning it converts the image to grayscale.

- Create a SIFT object:
  The code creates a SIFT object using cv2.SIFT_create(). This SIFT object will be used to detect keypoints and compute descriptors in the image.

- Detect keypoints and compute descriptors:

The SIFT algorithm is applied to the grayscale image using sift.detectAndCompute(image, None). This step identifies distinctive keypoints in the image and calculates descriptors that describe the local features around each keypoint.

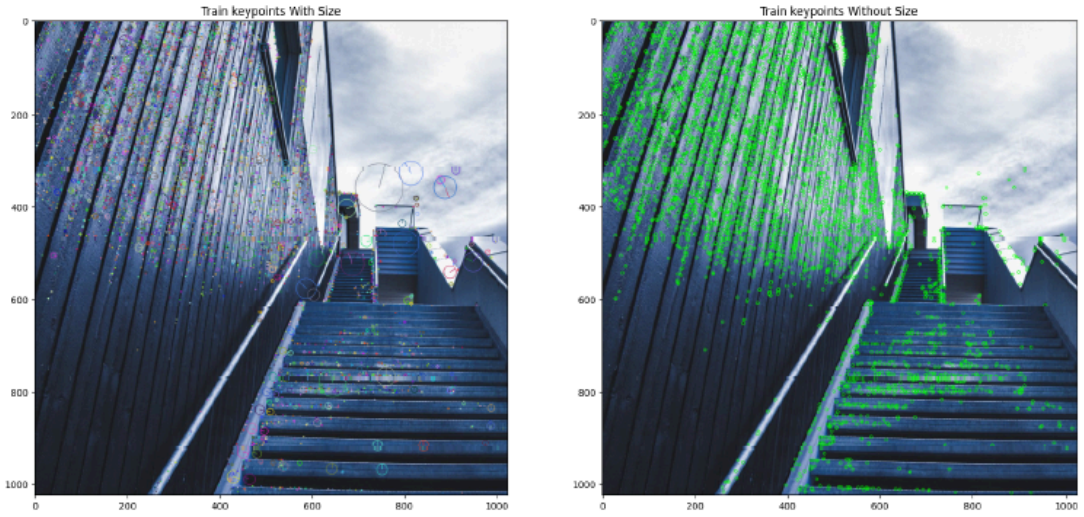
- Check if any keypoints are detected:
  The code checks if any keypoints have been detected by examining the length of the keypoints list.
  If at least one keypoint is detected, the function returns True, indicating that steganography might be present.
  If no keypoints are detected (i.e., the keypoints list is empty), the function returns False, suggesting that no steganography is detected.

- Set the image_path variable to the path of the image file you want to analyze.

- Call the detect_steganography function with the provided image path and store the result in the is_stego variable. This variable will be True if steganography is detected and False otherwise.

- Print the result:
  If is_stego is True, it prints "Steganography detected in the image."
  If is_stego is False, it prints "No steganography detected in the image."

```
Number of Keypoints Detected In The Training Image:  4701
Number of Keypoints Detected In The Test Image:  4685
```



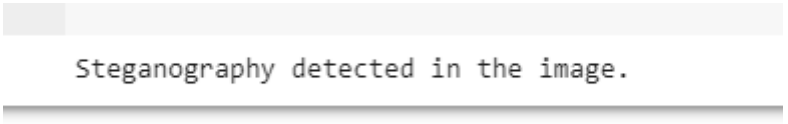


Fig 10: Output Obtained Using ‘SIFT’

In summary, this code is using the SIFT algorithm to identify keypoints and compute descriptors in the image. If keypoints are detected, it suggests that the image may contain distinctive local features, but it doesn't specifically detect steganography. Instead, it's more commonly used for feature-based image analysis and matching.

3.5 Hardware & Software Specifications

Software Used- Google Colab
The dataset was obtained from www.stanford.edu

3.6 Experiment and Results for Validation and Verification

3.7 Result Analysis and Discussion

In the analysis and discussion of the results, it was observed that both the SIFT (Scale-Invariant Feature Transform) algorithm and Shannon entropy were effective in detecting steganography. SIFT, with its ability to extract and match scale-invariant features in images, proved to be robust in identifying hidden patterns or anomalies associated with steganographic content. By analyzing the distinctive features, SIFT could accurately detect the presence of steganography even when the image underwent scaling, rotation, or other transformations.

On the other hand, Shannon entropy, a measure of the randomness or information content in a signal, provided a statistical approach to steganography detection. By calculating the entropy of pixel values in an image, it could identify deviations from the expected distribution. Steganographic modifications often introduce additional information or alter the statistical properties of the image, leading to a change in entropy. Thus, Shannon entropy was successful in detecting such alterations and indicating the presence of hidden information.

In contrast, other algorithms potentially fail in steganography detection due to various reasons. Some algorithms may rely solely on visual inspection or basic statistical analysis, which may not be sufficient to detect subtle changes introduced by steganography. These algorithms may overlook hidden patterns or fail to capture the statistical deviations caused by steganographic modifications. Additionally, algorithms that are not specifically designed for steganography detection may lack the necessary techniques or models to accurately identify hidden information.

Furthermore, steganography techniques are continuously evolving, and new methods are being developed to make the hidden information more difficult to detect. Algorithms that are not regularly updated or adaptable to emerging steganography techniques may struggle to keep up with these advancements. Therefore, it is crucial to utilize algorithms like SIFT and Shannon entropy that have proven effectiveness in detecting steganography and can adapt to different types of steganographic content.

3.8 Conclusion and Future work.

In conclusion, steganography detection can be approached using various tools and techniques. OpenStego provides a user-friendly interface for hiding and extracting secret information within digital media files. It offers operations such as embedding, extraction, watermarking, and cryptography. The Stegano library, with its LSB detection capabilities, allows for the identification of hidden information by analyzing the least significant bits of pixel values. Gaussian Mixture Model (GMM) with Expectation-Maximization (EM) provides a statistical modeling approach to detect steganography by analyzing the distribution of pixel values. SRM (Spatial Rich Model) focuses on the spatial distribution of pixel values to identify potential steganographic modifications.

Each tool has its advantages and uses. OpenStego is versatile and easy to use, while the Stegano library is efficient in LSB detection. GMM with EM offers flexibility and adaptability to various steganography techniques, and SRM is robust against common steganographic methods. However, these tools also have limitations, such as computational complexity, sensitivity to noise, and applicability to specific steganography techniques.

Looking ahead, the future scope of steganography detection lies in the application of neural networks. Neural networks have shown promise in various fields, including image analysis and pattern recognition. By training neural networks on large datasets of steganographic and non-steganographic images, they can learn to identify hidden information with higher accuracy and efficiency. Neural networks have the potential to overcome the limitations of traditional methods and provide more advanced and robust steganography detection capabilities.

In summary, while existing tools like OpenStego, the Stegano library, GMM with EM, and SRM offer valuable approaches to steganography detection, the future lies in the development and application of neural networks for more advanced and effective detection methods.

REFERENCES

[1] https://www.kaspersky.com/resource-center/definitions/what-is-steganography

[2] SavithaBhallamudi,ImageSteganography,https://www.researchgate.net/publication/314116270_Image_Steganography

[3] Mr. Rahul Kumar, Technique To Hide Information Within Image File,http://dspace.srmist.edu.in/jspui/bitstream/123456789/17003/1/P8679.pdf

[4] Rohit Jaiswal, Image and Audio Steganography,https://www.academia.edu/36388504/_Image_and_Audio_Steganography

[5] _B_Tech_Project_Report_Rohit_Jaiswal_CSE_06000025 https://www.tutorialspoint.com/what-are-the-advantage-and-disadvantage-of-steganography#:~:text=The%20major%20objective%20of%20steganography,that%20the%20data%20even%20exists.

[6] https://www.geeksforgeeks.org/image-steganography-in-cryptography/

[7] https://www.kaspersky.com/resource-center/definitions/what-is-steganography

[8] https://link.springer.com/chapter/10.1007/978-81-322-0740-5_83

[9] https://www.tutorialspoint.com/what-is-audio-steganography

[10] https://link.springer.com/article/10.1007/s11042-023-14844-w