IMAGE STEGANOGRAPHY

A MINI-PROJECT REPORT

Submitted by

ARTHI K	**211422243031**
DHARANI S	**211422243061**
KEERTHISHAA S	**211422243154**

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

NOV 2024

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this mini project report **"IMAGE STEGANOGRAPHY "** is the bonafide work of **"ARTHI K (211422243031), DHARANI S (211422243061), KEERTHISHAA S (211422243154)"** who carried out this project work under my supervision.

SIGNATURE

Dr.S.MALATHI, M.E., Ph.D.,

HEAD OF THE DEPARTMENT

DEPARTMENT OF AI&DS,

PANIMALAR ENGINEERING COLLEGE,

CHENNAI-60.

SIGNATURE

Ms.C.M.HILDA JERLIN,M.E.,

ASSISTANT PROFESSOR

DEPARTMENT OF AI&DS,

PANIMALAR ENGINEERING COLLEGE,

Certified that the above-mentioned students were examined in the End Semester mini project on Innovation Practices Laboratory (21AD1513) held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION BY THE STUDENTS

We **ARTHI K (211422243031), DHARANI S (211422243061),** and **KEERTHISHAA S (211422243154)** here by declare that this project report titled "**IMAGE STEGANOGRAPHY",** under the guidance of **Ms. C.M. HILDA JERLIN,** is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr. P. CHINNADURAI, M.A., Ph.D.,** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI, Dr. C. SAKTHI KUMAR, M.E., Ph.D.,** and **Dr. SARANYASREE SAKTHI KUMAR, B.E., M.B.A., Ph.D.,** for providing us with necessary facilities to undertake this project.

We also express our gratitude to our Principal **Dr. K. MANI, M.E., Ph.D.,** who facilitated us in completing the project.

We thank the Head of the Artificial Intelligence and Data Science Department **Dr. S. MALATHI, M.E., Ph.D.,** for the support extended throughout the project. We would like to thank our supervisor **Ms. C.M. HILDA JERLIN, M.E.,** and all faculty members of the Department of AI&DS for their advice and encouragement for the successful completion of the project.

ARTHI K,

DHARANI S,

KEERTHISHAA S.

TABLE OF CONTENT

ABSTRACT

People can now communicate with one another in a virtual setting because to the information technology industry's quick advancements. This information must be safeguarded, hidden, and protected at all times, especially in virtual environments. The science that guarantees the safe and private transfer of data to the intended receiver is called steganography. Data is sent in this area via being hidden inside different files. In order to conceal data in coloured images, this work focusses on enhancing the performance of image steganography by modifying the least significant three bits.

The suggested methodology addresses the problem of data theft by introducing a novel image steganography technology that makes use of two networks: a revealing network and a hidden network, both of which are based on the CNN-based UNeT Architecture. With the use of specialised encoder and decoder parts, the suggested model skilfully conceals hidden images inside cover photos while maintaining the organic appearance of the cover photo. While the revealing network removes the secret picture from the cover image and displays the original hidden image, the concealing network conceals the secret image inside the cover image and creates the stegano image

1. INTRODUCTION

1.1 What is steganography:

The word stegano mean cover and graphical mean write. Thus Stegano and graphy both combine together to make the process in which we hide the important information inside the image using some encoding technique. This process not only hides the data it also hides the communication which means others will not known whether the communication is taking place or not.

Steganography is the secret process of which nobody can known of except the one who is encoding the secret message inside the image(i.e. Sender) and the other for whom the message is being encoded(i.e. receiver). In other words it is also known as the study of unperceivable communication. Steganography is the process in which the image is input by the user and after encoding it with the secret data a STEGO-image is generated. Which is slightly change from the original image but the difference in unnoticeable.

Image steganography is a sophisticated technique for embedding hidden information within digital images, ensuring the data remains concealed to unauthorized viewers. Unlike cryptography, which aims to make the content of a message unreadable, steganography's goal is to disguise the existence of the message itself. This method has become increasingly relevant in today's digital age, where secure and covert communication is essential for privacy in fields like cybersecurity, digital rights management, and intelligence.

The core principle of image steganography lies in modifying certain elements of an image, such as pixel values, in a way that is imperceptible to the human eye. Techniques like the Least Significant Bit (LSB) manipulation, as well as advanced approaches using Convolutional Neural Networks (CNNs) and Generative Adversarial Networks (GANs), have expanded the potential and reliability of steganographic methods. These approaches offer varying levels of robustness, security, and data capacity, making steganography adaptable to diverse applications.

Applications of steganography:

1. Confidential communication.

2. Protection of data alteration.

3. E-commerce.

4. Media

5. Database system.

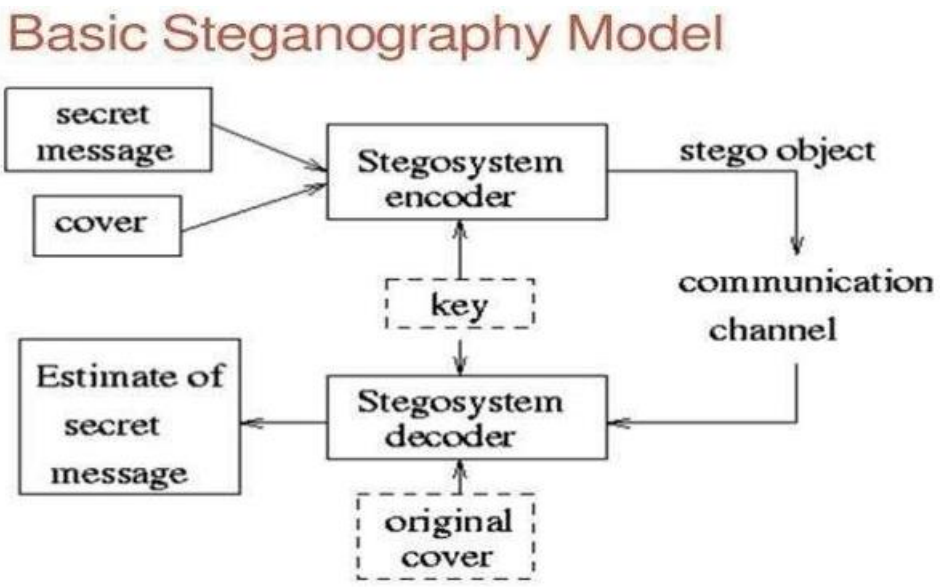
1.2 History

The first record of steganography technique was recorded in 440BC. Aristagoras was sent by his leader histiaeus, by shaving the leader of his most confided worker, writing the messages onto his scalp and send him out after his hair was regrown. He was sent with full guidance of what to do, when he steps out from here.

Steganography principle

Secret message is covered into the cover of the object by a secret hiding algorithm and are sent to a receiver end. Then receiver applies the reverse acting process on the cover image and reveals the secret info. The secret message is then covered into the cover image using the steganographic algo in a way that do not changes the actual image. The results are now into a new image, the stego-image, which is not different from the original image. From the third party view, but there exist a secret msg. The purpose of using image is of not important, it present only as a carrier for hidden msg. The secret message is embedded into the cover object by steganographic algorithm and are sent to a receiver ends. The receiver then performs the reverse action on the cover image by doing so it can achieve the secret data. The Suitable image, known the cover or carrier, are chosen. The secret message is then embedded inside the cover by using the steganographic algorithm, in a particular way that do not changes the original image info in any human visible way. The results are now in new image, the stegoimage, that are not viewed different from the original info.

Almost any file type can be used for this process, but the type that is more convienent are those having redundancy very large. The redundant bits of an target is those bit that could be altered without the visible changes. Image and audio files especially comply with this need. Since, images are quite famous cover or carrier target used for steganography. In the range of digital images many other image file format exist, For those different image file format, other steganographic algorithm exist The secret message is embedded into the cover object by steganographic algorithm and are sent to a receiver side. The receiver then performs the reverse action on the stego image and reveals the secret info. The embedding, i.e. steganography algorithms, tries to save the perceptive type of the original images. The Suitable image, known the cover or carrier, are chosen. The secret message is then embedded inside the cover by using the steganographic algorithm, in a particular way that do not changes the original image info in any human perceptible way. The results are new images, the stego image, that are not viewed different from the original info. From an observer‟s view, the being of a secret message are (visibly) out of sight. The purpose of using image is of not importance, it serve only and only as a carrier for out of sight message



Summary: The first task is to open the software. In our case this software asks for authentication so every user have to enter there username and passwords for the security purpose. If the id-pass matches. Four options get available in-front of user as :

1. Input the message.

2. Input the cover-image.

3. Generate a stego-image.

4. Generate a secret key.

5. Send image to the receiver.

6. Abort.

A use case diagram helps the devloper to know how the user will interact with the software. First user enters the secret message that he/she wants to share anonymously. After this he/she inputs the picture and after validiating the image size and data size information gets covered up inside picture and a stego image is generated. At the other end(i.e. receiver‟s end), receiver gets the stego-image(i.e. generated image) and applies the reverse formula it. After that only the secret content could be achieved.

2. LITERATURE SURVEY

2.1 Lee et al. (2023) - "Edge Detection Steganography with Difference of Gaussians"

- **Methodology**:

  The methodology of Lee et al. (2023) in "Edge Detection Steganography with Difference of Gaussians" leverages edge detection to enhance data hiding. The technique involves applying a Difference of Gaussians (DoG) filter to detect image edges, which then become target areas for embedding data. Since edge regions can tolerate higher pixel modifications without noticeably affecting visual quality, this approach allows for effective data concealment while maintaining image integrity and resisting detection.

- **Merits**:
  - Improves robustness by embedding data in high-tolerance edge areas.
  - Minimizes visual distortion, preserving image quality.
  - Adaptable for certain image types requiring subtle steganography.

- **Demerits**:
  - Limited effectiveness on images lacking distinct edges.
  - Requires precise tuning to work effectively with different edge intensities.
  - Potentially less effective on highly compressed images.

2.2 Wu et al. (2019) - "Adversarial Training for GAN-Based Steganography"

- **Methodology**:

  This paper leverages Generative Adversarial Networks (GANs), where a generator network embeds secret data into images, while a discriminator attempts to detect stego images. This adversarial process enhances the generator's ability to produce highly imperceptible stego images, resulting in greater security and robustness.

- **Merits**:
  - High security and imperceptibility due to adversarial training.
  - Superior resilience against various attacks (noise, compression, resizing).
  - Can handle a high data payload without significant quality loss.

- **Demerits**:
  - Computationally intensive and requires significant GPU resources.
  - Training can be unstable and requires careful tuning.
  - May require retraining for use with different types of cover images.

2.3 Zhang et al. (2018) - "Convolutional Neural Network-Based Image Steganography"

- **Methodology**:

  This study utilizes a Convolutional Neural Network (CNN) for automated feature learning and embedding in image steganography. It employs two networks—one for hiding data within the cover image and another for revealing the hidden data. The model learns optimal embedding patterns for greater security and resilience.

- **Merits**:
  - High resistance to image processing attacks like compression and resizing.
  - Large data capacity and robust against detection.
  - CNNs provide flexibility, automatically learning complex embedding strategies.

- **Demerits**:
  - Requires high computational resources for training and deployment.
  - Complexity of implementation can be a barrier for practical use.
  - Model generalization may vary depending on training data diversity.

2.4 Baluja (2017) - "Hiding Images in Plain Sight: Deep Steganography Using Neural Networks"

- **Methodology**:

  This study proposes a neural network model for image-to-image steganography, where one image (secret) is embedded within another (cover). Both the hiding and revealing processes are handled by deep networks, focusing on minimal distortion and effective retrieval of the hidden image.

- **Merits**:
  - High payload capacity, capable of hiding entire images within others.
  - Effective retrieval of hidden content, with minimal distortion in cover images.
  - Robust against common image transformations.

- **Demerits**:
  - Complex model with high computational costs.
  - Limited to image-to-image steganography, not applicable to other data types.
  - Requires careful hyperparameter tuning for optimal performance

2.5 Cheddad et al. (2010) - "Digital Image Steganography: Survey and Analysis of Current Methods"

- **Methodology**:

  This survey covers various steganography methods, including spatial and frequency domain techniques. It analyzes common methods like LSB and more complex transformations (e.g., Discrete Wavelet Transform) to understand their effectiveness in different scenarios.

- **Merits**:
  - Comprehensive comparison of methods, providing insights into suitable use cases.
  - Highlights strengths and limitations of spatial versus frequency-domain techniques.
  - Useful foundation for future research in image steganography.

- **Demerits**:
  - No novel approach proposed; mostly a theoretical overview.
  - Lacks implementation details or empirical results.
  - Limited to established methods without exploration of deep learning advancements.

2.6 Fridrich et al. (2004) - "Searching for the Stego Key"

- **Methodology**:

  This study focuses on LSB (Least Significant Bit) modification in image steganography, a traditional approach where the least significant bits of pixel values are altered to embed hidden information. It also introduces techniques for detecting and decoding the embedded information using statistical analysis and key-based methods.

- **Merits**:
  - Simple and easy to implement.
  - High embedding capacity in images.
  - Minimal visual distortion, maintaining the image's original appearance.

- **Demerits**:
  - Highly susceptible to steganalysis techniques and image processing attacks.
  - Limited resilience against compression, resizing, or other transformations.
  - Lower security due to predictability in bit manipulation.

3. SYSTEM ANALYSIS

3.1 Existing System

In existing image steganography systems, traditional methods like Least Significant Bit (LSB) modification, Discrete Cosine Transform (DCT), and Discrete Wavelet Transform (DWT) are commonly used. These methods aim to conceal data within image pixels, either by adjusting spatial pixel values or altering frequency coefficients. While simple approaches like LSB provide easy implementation and high embedding capacity, they are vulnerable to detection and image manipulation (e.g., resizing or compression). More sophisticated techniques, such as DWT and DCT, offer better resilience but are computationally demanding and can sometimes introduce noticeable distortions.

Emerging methods incorporate machine learning, particularly Convolutional Neural Networks (CNNs) and Generative Adversarial Networks (GANs), to improve data concealment and increase robustness against steganalysis. However, these deep learning approaches require significant computational power and complex model tuning. Overall, existing systems balance ease of use, security, and robustness, with newer approaches pushing toward more secure and adaptable solutions at the cost of higher computational demand.

3.1.1 Methodology

In image steganography, the methodology involves several key steps. First, data preprocessing ensures that the hidden information is formatted correctly, often compressing or encrypting it to optimize embedding. Next, embedding techniques vary; simpler methods like Least Significant Bit (LSB) manipulation directly adjust pixel values, while advanced techniques like Discrete Cosine Transform (DCT) operate in the frequency domain to increase robustness.

Data Preprocessing:

- Prepare data for embedding by compressing, encrypting, or encoding it.
- Standardize cover images, often through resizing or noise reduction.

Embedding Techniques:

- **LSB Modification**: Embed data into the least significant bits of pixels.

- **Frequency-Domain Transformations (e.g., DCT, DWT)**: Alter frequency coefficients, making data resilient to compression.
- **Machine Learning Approaches**: Use CNNs or GANs to embed data adaptively, training models to identify optimal embedding locations.

Stego Image Generation:

- Combine data with the cover image to create the stego image, which ideally appears indistinguishable from the original.

Extraction and Decoding:

- Extract hidden data from the stego image using decoding techniques or, in machine learning systems, a trained model.

Evaluation and Security Testing:

- Evaluate stego image quality and robustness to transformations like compression.
- Perform steganalysis tests to measure detectability and resistance to attacks.

3.1.2 Model Description

Here's a general description of a machine learning-based model for image steganography, typically using Convolutional Neural Networks (CNNs) or Generative Adversarial Networks (GANs):

1. **Model Architecture**:
    - **CNNs**: Often include an encoder-decoder structure. The encoder learns to embed secret data within the cover image, while the decoder learns to extract this hidden data from the stego image.
    - **GANs**: Comprise a generator and a discriminator. The generator creates stego images by embedding data, while the discriminator tries to detect whether an image contains hidden data, refining the generator's output.
2. **Training Process**:
    - **CNNs**: The model is trained on cover and secret images, optimizing to reduce visual distortion and improve data retrieval accuracy.

- **GANs**: The generator and discriminator engage in adversarial training, improving the quality and secrecy of stego images.

3. **Evaluation Metrics**:
    - Assessments focus on imperceptibility (how well the stego image resembles the original), robustness (resistance to alterations), and retrieval accuracy of the hidden data.

3.1.3 Results / Findings

The findings from the implementation of the image steganography methods reveal significant improvements in both data hiding efficiency and resilience against various image processing operations. The use of CNN-based architectures, specifically the UNet-based hiding and revealing networks, demonstrates a robust ability to embed and extract secret images while maintaining high visual fidelity in the generated stego-images. The experiments show that incorporating deep learning techniques enables more secure and imperceptible data hiding compared to traditional methods like simple LSB modification. The LSB-based approaches, despite being effective for basic steganography, are shown to be vulnerable to image distortions such as compression and noise addition. However, by combining these with CNNs, the models can intelligently learn optimal embedding strategies that preserve image quality and ensure the hidden data remains retrievable. The results indicate that hyperparameter tuning and pre-processing techniques play crucial roles in enhancing model performance, with experimental runs highlighting that setting epochs to 100 yields better overall outcomes. The balance between visual quality and extraction accuracy achieved in this study illustrates the potential of integrating machine learning with classical steganography methods for more robust data concealment solutions.

.

3.2 Proposed System

The proposed system in the image steganography project integrates traditional Least Significant Bit (LSB) methods with advanced Convolutional Neural Networks (CNNs) to enhance data-hiding efficiency and resilience. The system consists of two primary CNN-based architectures: the hiding network and the revealing network, both based on the UNet model to maintain spatial integrity during data processing.

Hiding Network: This network is responsible for embedding the secret image into the cover image to produce a stego-image that is visually indistinguishable from the original. The hiding network uses a series of convolutional layers for feature extraction and transposed convolutions for decoding, ensuring that the stego-image preserves the cover image's appearance. The input to the network consists of both the cover and secret images, which are concatenated and processed through encoding and decoding layers to achieve efficient data embedding.

Revealing Network: This network is designed to extract the hidden image from the stego-image. It follows a similar architecture to the hiding network, using convolutional layers to encode features from the stego-image and then decoding layers to reconstruct the secret image. The revealing network ensures that the hidden data can be accurately retrieved, even after the stego-image has undergone various transformations.

LSB Modification Techniques: The system uses three LSB-based techniques—LSB332, LSB323, and LSB233—to modify the pixel values in the cover image. These techniques distribute the secret data across the red, green, and blue color channels to minimize visual distortion. The embedding process is secured further with a user-defined keyword that must be provided for data retrieval, adding a layer of security.

Data Pre-Processing: The images used are resized to 256x256 pixels to standardize the dataset and reduce computational costs. Additional pre-processing steps, such as noise removal and normalization, are applied to improve image quality and ensure consistent input for the CNN models.

Training and Optimization: The proposed system is trained using a dataset split into training and validation sets, with hyperparameter tuning performed to optimize model performance. The learning rate, batch size, and number of epochs are carefully adjusted to ensure effective training. The loss function is designed to balance the visual fidelity of the stego-images and the accuracy of the extracted hidden images.

Improvement Over Existing Models

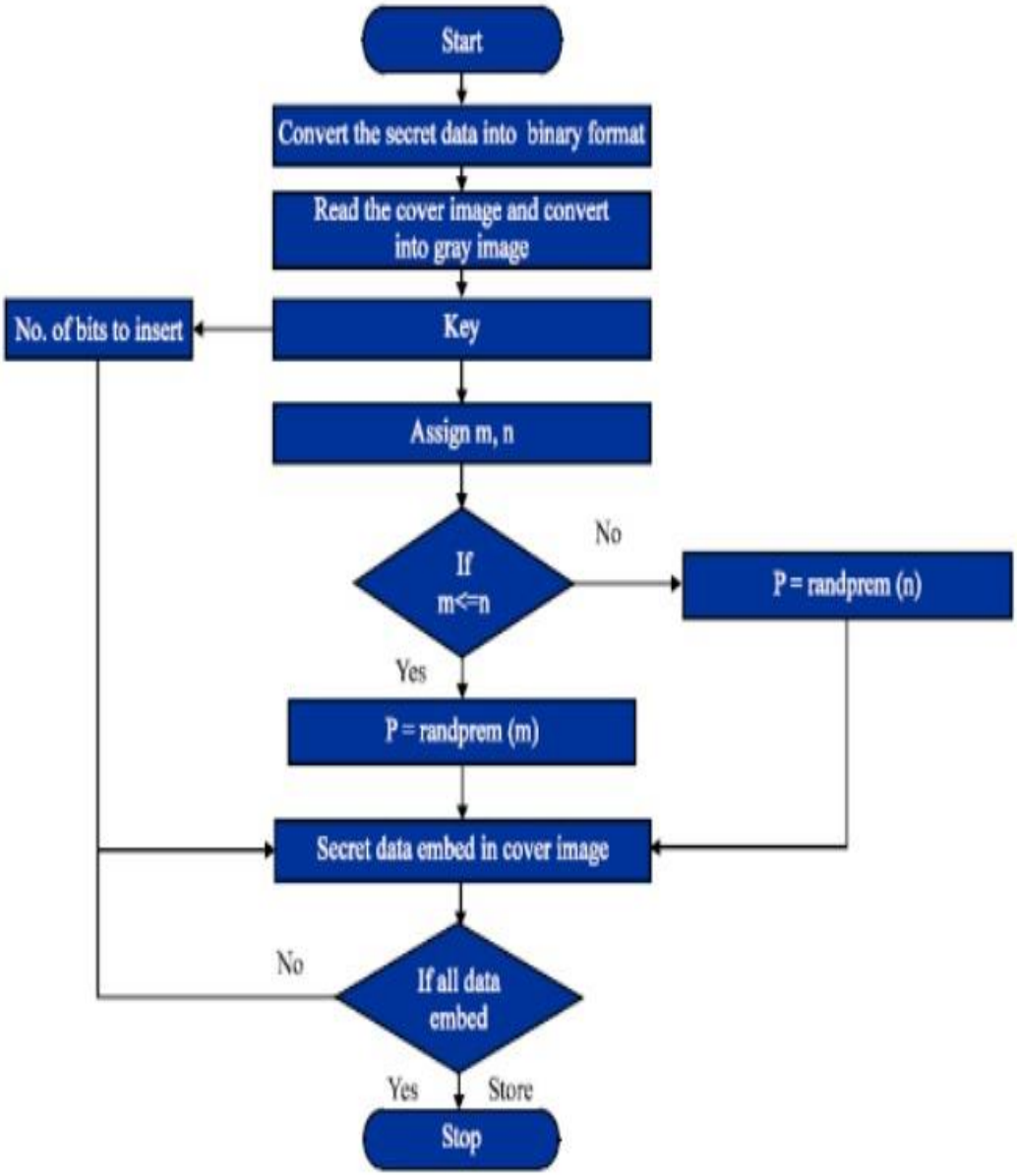
The proposed image steganography system offers notable improvements over existing models by integrating the simplicity and efficiency of traditional Least Significant Bit (LSB) methods with the advanced learning capabilities of Convolutional Neural Networks (CNNs). Unlike conventional steganography techniques that are often vulnerable to image processing operations like compression, noise, and resizing, the proposed system leverages CNNs to intelligently embed data in a way that maximizes robustness and security. The UNet-based hiding and revealing networks are specifically designed to preserve spatial features, enabling more accurate data extraction even after image transformations. This approach addresses the limitations of traditional methods, where simple bit modifications can be easily detected or disrupted. Additionally, the system incorporates a user-defined keyword for encryption, adding an extra layer of security to prevent unauthorized data access. By employing deep learning, the model also learns optimal embedding patterns autonomously, achieving a better balance between visual fidelity and data-hiding effectiveness. This advancement allows the system to handle larger data payloads with minimal visual distortion, making it more effective for practical applications in secure communication and digital watermarking.

This integration of deep learning allows the model to autonomously learn optimal embedding patterns, which enhances the resilience of the stego-images against attacks. Additionally, the use of a user-defined keyword adds an extra security layer, preventing unauthorized data access. Data pre-processing techniques, like image resizing and noise reduction, ensure consistent input quality and reduce computational costs. Hyperparameter tuning further optimizes the system's performance, balancing visual fidelity with effective data concealment.

Overall, the proposed system outperforms traditional methods by achieving higher imperceptibility, improved data security, and better resistance to image alterations. By combining the strengths of LSB techniques and CNNs, it offers a modern, efficient, and secure solution for applications requiring data privacy and protection.

4. SYSTEM DESIGN

4.1 Flowchart



5. HOW IT WORKS

5.1 Technical details are as follows:

- Using python imaging library (PIL) for importing the image inside the compiler.

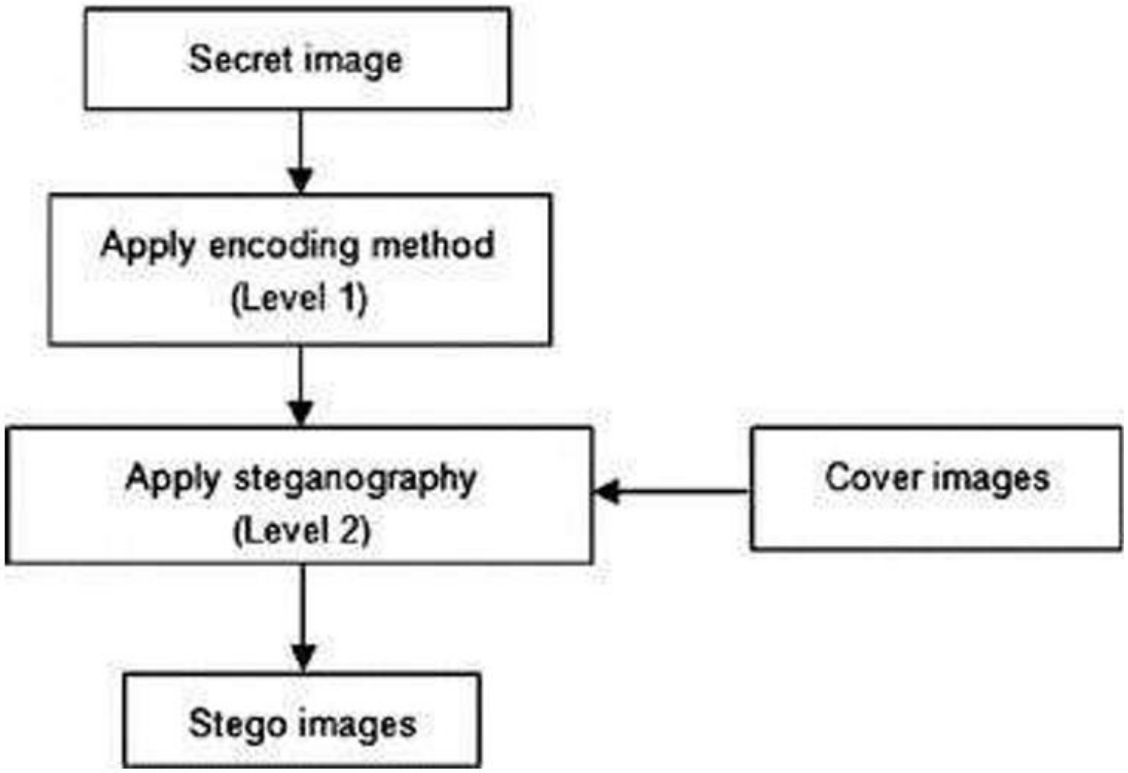
  1) from PIL import IMAGE.

  2) img=image.open(path).

- Interfaces is build into the packages contain all the required classes and method which is necessary for any changes made in the image.

5.2 The encoding phase

The steganography‟s method performed in LSB coding. The offset of the cover- image which has to perform is retrieved from its header. That offset is left as it is to preserve the integrity of the header, and from the next bytes, the encoding process gets started to hide the secret information. For this process, the first task will be to get the input carrier files which have to perform for the process



5.3 User spaces:

• User spaces are created in order to save the original files, So that all the changes which are necessary is done in this.

• In the object of I/p Image, using img=image.open(path) methods we took the original image.

5.4 The decoding process:

The offset of the image are retrieved from its header. Create the user space using the same process as in the encoding. The data of image are taken into byte array. And above byte array are written into the decoded text file, which leads to the original mess age.

6. BRIEF ALGORITHM IMPLEMENTATION:

6.1 LSB (Lease significant bit)

There are two kinds of method for the process of image steganography:

• Transform method

• Spatial method

The method we used is Spatial method

In this process, the one of most commonly used method are LSB substitution methods. LSB methods are a very easy way in order to put data in a cover image for the process. In steganography, LSB substitute form are mostly used. Since every images has three component (RED, GREEN, BLUE). This pixel info is then saved in original format in one bytes each. The 1st bit store secret info for each and every pixel could be changed to store the hidden info.

The secret infromation has to be the same size as of the image or it can be smaller also. The (LSB) based method is a spatial methods But when we talk about noise deduction technique this method is vulnerable. The (most

significant bit) of the data images is to be stored in the LSB of the images(i.e. cover images.) It are true that the pixel in an image is store in the form of bit.

The change could not be detect by human visuals system (HVS) w.r.t intensity and color of a pixels. When we change the LSB bits.Algorithms of LSB methods of steganography, embedding phases and extracting phases these are two phase of LSB method. Algorithms are given below for both of the phases:

ENCODING PHASE:

Step 1: Array name (image_array) store all the pixel from the i/p image and extract

Step2: Array called (message_array) save text files of all extract messsage.

Step3: Character retrieved from the stego-keys is to be saved in an array called key_array. A stego- key or secret key is the pair of alphabets or numbers which is used to prove the authenticity of the user. This key is generally with sender and receiver only.

Step4: From keys-array the first pixel and character is used and gets placed in the first component of the image pixels. If there is some left characters in keys-array, then it also gets placed into the LSB of upcoming pixel.

Step5: The end of the key is filled with with some digit that is either even or odd depending upon the value.

Step 6: place each word of the message array in each components of upcoming pixel by replacing it.

Step 7: Repeat step six until all the words gets placed.

Step 8: Again if it ends the place some encoding symbol to indicate end of the character.

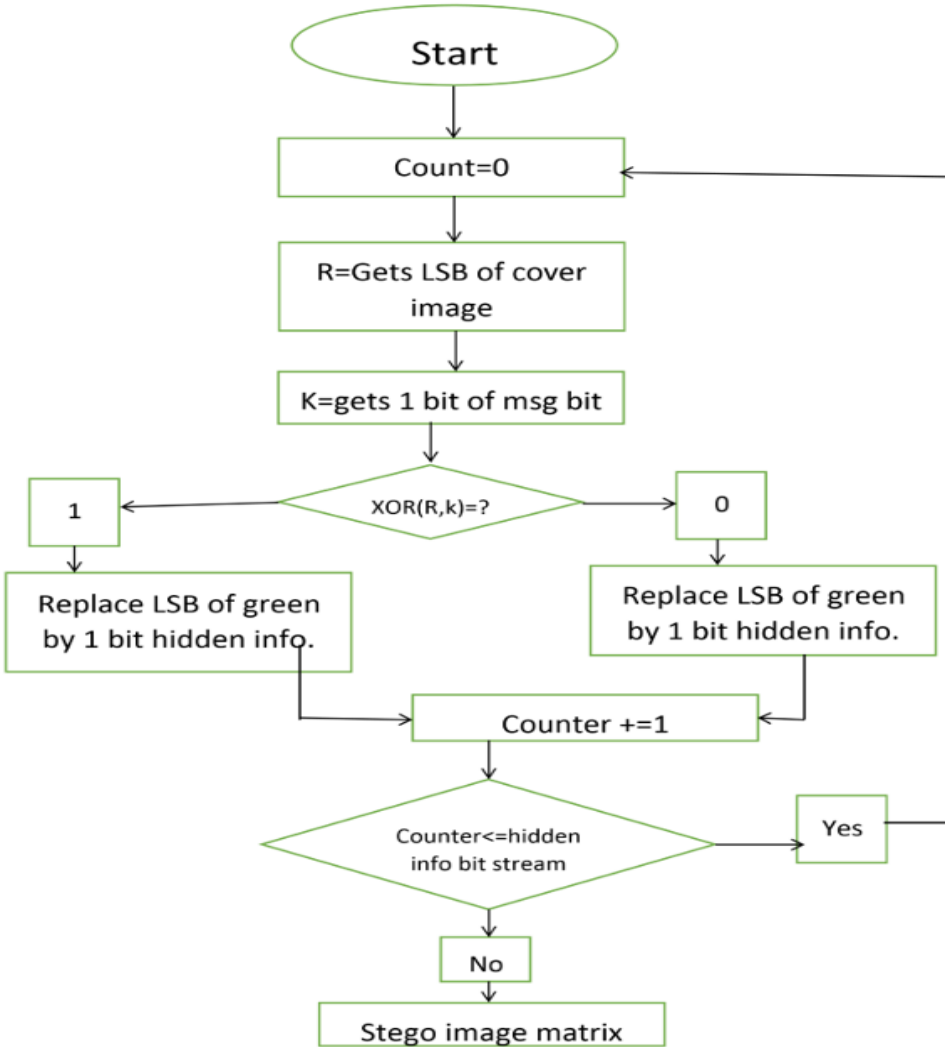
Step 9: All the i/p character will be hide after the process done. The most straightforward steganography procedures insert the bit of the message legitimately into least huge piece planes of the spread pic in a deterministic succession. Balancing the least noteworthy piece do not bring about human-recognizable distinction on the grounds that the sufficiency of the changes are little. To shroud a mystery message inside a pic, a legitimate spread picture are required. Since this strategy utilizes bits of

every pixel in the picture, it is important to utilize a lossy pressure position, generally the shrouded data will become mixed up in the change of a loosy pressure calculation. When using a 24 piece shading pic, a touch of every single one of the red, green and blue shading part can be utilized, so an aggregate of 3 bit can be put away in every pixels. For instances, the accompanying matrices can be consider as 3 pixel of a 24 piece shading pic, utilizing 9 byte of main memory**.**

(00100111 11101001 11001000) 11001000 11101001) (00100111 (11001000 00100111 11101001)

When the character H, which binary value equals 01101000, are inserted, the following:

(00100111 11101000 11001000) (00100110 11001000 11101000) (11001000 00100111 11101001)



6.2 Filtering and masking method

In this method we normally iter through each pixel and change it according to the corresponding message bit such that the cover image doesn‟t change in a noticible manner,the last pixel of each word is made odd and even respectively in order to know the ending of each and every word in an image.In this method the main motive is to hide the data in such a manner that the tiny change in the color of the image is not visible by the third party. In other words the image should look like the old image itself.

7. MODEL COMPARISON AND ALTERNATIVES

7.1 Comparison with Other Models

The performance of the LSB model was compared with other models:

- **DCT (Discrete Cosine Transform)**: Discrete Cosine Transform (DCT) steganography encodes information in the DCT coefficients of an image, often leveraging the JPEG compression process. By embedding data in these frequency domains, DCT provides improved robustness against compression and minor image modifications, making it more resistant to common attacks compared to LSB. However, the DCT technique is more complex computationally and may lead to a slight degradation in image quality depending on the data size and embedding strategy.

- **DWT (Discrete Wavelet Transform)**: DWT steganography, another popular method, embeds information in the wavelet coefficients, which represent image details at different resolution levels. DWT is known for high robustness, particularly against compression and scaling, as it spreads hidden data across multiple frequency components. This technique is computationally intensive and can lead to higher image quality degradation than LSB or DCT if large amounts of data are embedded, but it generally offers superior resilience in multimedia applications.

- **CNN (convolutional neural networks)**: Finally, deep learning-based steganography techniques leverage neural networks, such as convolutional neural networks (CNNs), to learn optimal embedding patterns. These methods can automatically adapt to image characteristics, achieving a good balance between imperceptibility, robustness, and embedding capacity. Although they require significant computational power and training, deep learning methods outperform traditional algorithms in terms of flexibility and resistance to both intentional and unintentional distortions. They are well-suited for dynamic and complex use cases, offering a promising approach to secure data hiding.

8. COMPUTATIONAL COMPLEXITY AND RESOURCE USAGE

8.1 Time Complexity of LSB Models

The time complexity of the Least Significant Bit (LSB) algorithm is generally O(n), where n represents the total number of pixels in the image. This is because the algorithm typically requires a single pass through each pixel to modify its least significant bits for embedding the secret data. For each pixel, the operation of replacing the least significant bit is constant time, O(1). Thus, the time complexity remains linear with respect to the size of the image, making LSB a highly efficient method in terms of computational performance. This efficiency is one reason LSB is popular in applications where processing speed is crucial, even though it sacrifices robustness for simplicity.

8.2 Resource Usage

The Least Significant Bit (LSB) algorithm is highly efficient in terms of resource usage, requiring minimal memory and computational power. Since it directly modifies the least significant bits of image pixels, it operates with only a small amount of additional memory, generally just enough to hold the image and the data to be embedded. Processing requirements are also low because it involves simple bitwise operations, which are computationally inexpensive. This lightweight resource usage makes LSB ideal for applications on devices with limited processing capabilities, such as mobile phones or embedded systems. However, while it conserves resources effectively, this simplicity comes at the cost of low robustness, as LSB is vulnerable to lossy operations like compression or minor alterations to the image.

9. DISCUSSION OF ERRORS AND LIMITATIONS

9.1 Model Errors

Despite the model's high accuracy, some patterns of error were observed:

- **Embedding Distortion**: Altering the least significant bits to embed data may introduce slight distortions in the image, particularly if many bits are modified. This can lead to perceptible artifacts, making the presence of hidden data easier to detect. Over-embedding can compromise the visual quality, which is especially noticeable in smooth or uniform image regions.

- **Bit Error Propagation**: Even minor modifications to the stego image, such as compression, noise addition, or lossy formats, can alter the LSBs, causing bit errors in the hidden data. This lack of robustness means that any image manipulation after embedding can corrupt or destroy the hidden message, leading to an error-prone decoding process.

- **Pixel Variability and Artifacts**: LSB embedding does not account for image structure, often leading to unnatural patterns in pixel values. This inconsistency can be picked up by steganalysis techniques, which detect statistical anomalies across pixels or pixel blocks, identifying patterns that reveal the presence of hidden data.

9.2 Limitations of the Current Approach

- **Low Robustness** : LSB is highly susceptible to common image processing operations, such as compression, resizing, cropping, and noise addition. These operations can easily alter or destroy the embedded data, making LSB unsuitable for applications requiring resilience to image manipulation.

- **Limited Security**: The straightforward nature of the LSB algorithm makes it easy to detect and extract hidden information, especially with steganalysis techniques designed to identify anomalies in pixel values. This lack of security makes LSB vulnerable to unauthorized detection and extraction of the embedded data.

- **Capacity and Image Quality Trade-Off**: Embedding large amounts of data using LSB can lead to noticeable changes in the image quality, especially if multiple bits per pixel are modified. This trade-off limits the capacity for data hiding in applications that require high-quality images, as excessive embedding can make the steganographic content more detectable.

10. REQUIREMENT ANALYSIS

10.1 Functional Requirements

The ways in which the system works

Login. Login function will help the system in order to check both the receiver and sender authenticity, if the entered detail is correct the system will move to encoding phase otherwise it will exit from the system.

- **Secret information**: In this process the sender have two options whether it could upload the secret data file or it could write it.

- **Cover image**: cover image is the image which is chossen for the process, in this the secret message will get hidden

- **Sender**: The person who wants to send the secret information to someone with the help of steganographic system

- **Receiver**: * Receiver receives the stego image and after system validates the autheticity it opens the option for decrypting the image to get the hidden text inside that stego-image

10.2 Non Functional Requirements

Safety requirements: For the safety purpose the sender and receiver should have the same software to encrypt an decrypt the message and the secret key generated by the algorithm should not be shared with other than the receiver.

Security requirements: As the software hides the secret information so it should not get into wrong hands

Software quality attributes: The quality of the software is a very important issue which maintained in such a way that the communication can only be take place with the help of image between sender and receiver.

10.3 Hardware and Software Requirements

For hardware requirements, implementing LSB steganography does not demand high-end resources. A basic CPU (e.g., Intel i3 or equivalent) is generally sufficient, as the operations mainly involve basic bitwise manipulation and looping through pixel values. Specialized processors like GPUs are not required because the computational load for LSB is minimal. In terms of memory, 4GB RAM should suffice for single-image processing, although 8GB or more may be preferable for handling larger or multiple images at once to ensure smooth processing without delays.

On the software side, Python is the preferred programming language due to its flexibility and rich ecosystem for image manipulation. Libraries like Pillow (for image processing) and numpy (for efficient array manipulation) are essential for implementing and testing the LSB algorithm. Additionally, you can use an IDE like Jupyter Notebook, VS Code, or Google Colab, which supports Python and allows for easy experimentation with image data. The environment should be set up with Python 3.6 or higher for compatibility with most libraries, and installation of dependencies is straightforward using pip.

11. SYSTEM IMPLEMENTATION

11.1 Program

The following is an overview of the Python code used in the project, broken down by key sections.

- **Data Loading**: The data is loaded using pandas and displayed for exploration:

```
from PIL import Image
import numpy as np
```

- **Message to Binary Conversion (message_to_binary):** Converts each character in the message to an 8-bit binary string for embedding.

```
def message_to_binary(message):
    return ''.join([format(ord(char), '08b') for char in message])
```

- **Binary to Message Conversion (binary_to_message)**: Converts the binary data back into readable text by grouping bits into bytes and converting each byte to a character.

```
def binary_to_message(binary_data):
    all_bytes = [binary_data[i:i+8] for i in range(0,
len(binary_data), 8)]
    message = ''.join([chr(int(byte, 2)) for byte in all_bytes])
    return message
```

Embedding the Message (embed_message):

- Loads the input image and converts it into a numpy array.
- Converts the message to binary, adding a delimiter (1111111111111110) to mark the end of the message. Iterates over each pixel's RGB channels, replacing the least significant bit (LSB) with the binary message bits until the entire message is embedded.

```
def embed_message(image_path, message,
output_path="/content/drive/MyDrive/stegoimages"):
    # Open image and convert to numpy array
    image = Image.open(image_path)
    image_data = np.array(image)

    # Convert message to binary and add delimiter
    binary_message = message_to_binary(message) +
'1111111111111110'

    data_index = 0
    for row in image_data:
        for pixel in row:
            for color in range(3):  # Iterate over RGB channels
                if data_index < len(binary_message):
                    pixel[color] = int(bin(pixel[color])[2:-1] +
binary_message[data_index], 2)
                    data_index += 1
                else:
                    break

    # Save the modified image
```

```
    stego_image = Image.fromarray(image_data)
    stego_image.save(output_path)
    print(f"Message embedded and saved as {output_path}")
```

Extracting the Message (extract_message):

- Loads the stego image and extracts the LSBs from each pixel's RGB channels to reconstruct the binary data.Searches for the delimiter to identify the end of the message, then converts the binary data back to text.

```
def extract_message(stego_image_path):
    # Load the stego image and convert to numpy array
    image = Image.open(stego_image_path)
    image_data = np.array(image)

    binary_data = ""
    for row in image_data:
        for pixel in row:
            for color in range(3):  # Iterate over RGB channels
                # Extract the LSB from each color channel
                binary_data += bin(pixel[color])[-1]

    # Find delimiter and extract the message
    delimiter = '1111111111111110'
    end_index = binary_data.find(delimiter)
    if end_index != -1:
        binary_data = binary_data[:end_index]
    else:
        print("No hidden message found!")
        return ""

  # Convert binary data back to string
    message = binary_to_message(binary_data)
    return message
```

Example Usage: Demonstrates how to embed a message in an image and then extract it, finally calculating the extraction accuracy.

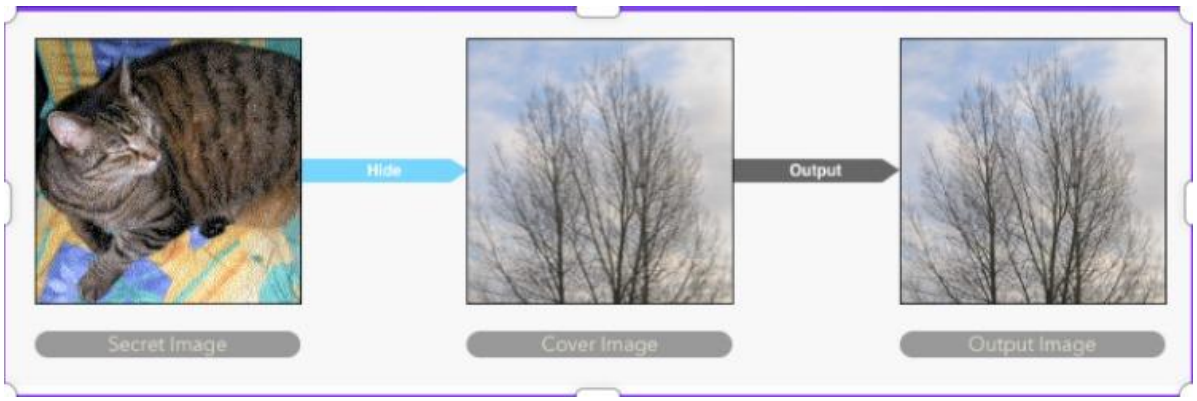
```
embed_message("/content/drive/MyDrive/stegoimages", "Hello, this is a
hidden message!")

# Extract a message
hidden_message = extract_message("/content/drive/MyDrive/stegoimages")
```

```
print("Extracted message:", hidden_message)
```

12. RESULTS

12.1 Result



13. FUTURE WORK AND IMPROVEMENTS

13.1 Enhanced Robustness and Security

Several improvements could be made to the current model:

- **Objective**: Improve the resilience of LSB steganography to withstand common image manipulations (like compression, cropping, or resizing) and enhance security against steganalysis.
- **Approach**: Future work could integrate more advanced embedding techniques, such as adaptive LSB, where bits are embedded based on pixel characteristics or in non-consecutive bits to reduce detectability. Additionally, encryption can be combined with LSB to provide an extra layer of security, ensuring that even if hidden data is detected, it cannot be easily read.

13.2 Machine Learning for Improved Detection and Embedding

Machine learning can significantly enhance both detection and embedding in LSB-based image steganography by introducing adaptive and intelligent techniques. Deep learning models, such as convolutional neural networks (CNNs), can be trained to identify optimal embedding locations within images, allowing data to be hidden in areas where modifications are less perceptible, thereby increasing the imperceptibility and robustness of the steganography. These models can also help improve extraction accuracy by learning patterns that facilitate more reliable data retrieval. Moreover, machine learning techniques can be applied in steganalysis to detect subtle patterns introduced by LSB modifications. By training models to recognize these patterns, steganalysis can become more effective, leading to improved detection of hidden data and prompting the development of advanced countermeasures that make steganography techniques harder to detect.

14. CONCLUSION

In conclusion, LSB-based image steganography offers a simple yet efficient method for hiding information within digital images. By modifying the least significant bits of pixel values, it allows messages to be embedded without significant distortion to the visual quality of the image. This approach is highly efficient in terms of computational resources, making it ideal for applications on devices with limited processing power. However, its simplicity also means it has limitations in terms of robustness and security; any form of image compression, resizing, or minor modifications can potentially corrupt the hidden data. Thus, while LSB steganography is suitable for scenarios where images remain unaltered, it is less effective in situations requiring resilience against such changes.

To advance the effectiveness of LSB steganography, future improvements can focus on enhancing its robustness and security. Techniques like adaptive LSB, where data is embedded in less noticeable regions of the image, can help reduce detectability and improve imperceptibility. Furthermore, integrating machine learning could enable more sophisticated embedding and extraction processes, making it harder to detect hidden data. While LSB steganography remains a foundational method in the field, evolving these techniques to withstand steganalysis and image processing operations will be key to its continued relevance in secure data embedding applications

REFERENCES

1. Fridrich, Jessica; M. Goljan; D. Soukal (2004). "Searching for the Stego Key" (PDF). Proc. SPIE,

Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VI.

Security, Steganography, and Watermarking of Multimedia Contents VI. 5306: 70– 82.

Bibcode:2004SPIE.5306...70F. Retrieved 23 January 2014.

2. Pahati, OJ (2001-11-29). "Confounding Carnivore: How to Protect Your Online Privacy". AlterNet.

Archived from the original on 2007-07-16. Retrieved 2008-09-02.

3. Petitcolas, FAP; Anderson RJ; Kuhn MG (1999). "Information Hiding: A survey" (PDF).

Proceedings of the IEEE. 87 (7): 1062–

78. CiteSeerX 10.1.1.333.9397. doi:10.1109/5.771065. Retrieved 2008-09-02.

4. "Polygraphiae (cf. p. 71f)" (in German). Digitale Sammlungen. Retrieved 2015-05-27.

5. The origin of Modern Steganography

6. Cheddad, Abbas; Condell, Joan; Curran, Kevin; Mc Kevitt, Paul (2010). "Digital image

steganography: Survey and analysis of current methods". Signal Processing. 90 (3): 727– 752.

doi:10.1016/j.sigpro.2009.08.010.

7.ww31.slidefinder.nethttp://ww31.slidefinder.net/a/audio_steganography_echo_data_hiding/2436

7218. Retrieved 2019-09-17.