DATA SECURITY
CSC8370
COURSE PROJECT

"Secure Data Communication Using Image Steganography"

Contributions:

- ➢ Grace Selina Pagadala: Encryption Code, Testing
- ➢ Jasmika Vemulapalli: Novel Idea, Decryption Code
- ➢ Poojasri Gadde: Algorithm, Decryption Code
- ➢ Sri Haneesha Davuluri: Encryption Code, Overall Code Integration

Contribution Ratio of Team Members: 25% each

Abstract

In the contemporary realm of cyber communications, the sophistication of cyber crimes has surged, posing significant threats to data confidentiality and integrity. According to Security Magazine, there are over 2,200 attacks each day which breaks down to nearly 1 cyberattack every 39 seconds. To counteract such threats on data, traditional security systems have expanded their arsenal to include not only cryptography but also steganography. While cryptography obscures the content of a message, transforming plaintext into an unintelligible cipher, steganography takes a more surreptitious approach by embedding messages within other non-sensitive data, leaving no apparent trace of tampering. This paper addresses the application of Image Steganography, comparing its efficacy against traditional cryptographic measures and evaluating its potential to thwart unauthorized data interception. We delve into the nuances of Image Steganography, underscoring its strategic advantages and potential drawbacks within the context of data security.

This project represents an interdisciplinary endeavor to fortify data transmission through the adept use of steganographic techniques. Each team member contributed equally, integrating encryption and decryption code development, algorithm refinement, and overall code integration to achieve a harmonious balance between data confidentiality and seamless communication.

The initiative sets out to capitalize on the ubiquity of digital imagery, manipulating pixel representation and color values to conceal sensitive information. By leveraging the Least Significant Bit (LSB) Substitution method, the project demonstrates the feasibility of transmitting hidden messages imperceptible to both human observers and unsophisticated algorithms. Our results indicate a high degree of visual fidelity between original and steganographically modified images, affirming the technique's viability. The project's success lays the groundwork for future exploration into more advanced steganographic strategies and the development of intuitive interfaces, potentially elevating the standard of secure communication in the digital age.

Objective

The overarching goal of this project is to develop a robust and secure data communication system through the adept application of steganography principles. Employing Python as the primary programming language and leveraging the capabilities of the PIL (Pillow) library, our team is dedicated to crafting an innovative solution that discreetly conceals sensitive information within digital images. The primary aim is to render the presence of this hidden data virtually undetectable to unauthorized users, ensuring a high level of confidentiality and security.

Problem Statement

Cybercrime has evolved into a formidable adversary in the domain of data security, with adept adversaries employing a plethora of methods to compromise valuable and confidential data. The urgency to bolster data security measures against such insidious attacks has never been greater. Conventional systems have employed a variety of techniques to secure electronic communications and safeguard them from unauthorized entities. Among these, steganography has emerged as a nuanced strategy that embeds messages within other non-critical data, maintaining the appearance of the carrier data unaltered post the embedding process. However, the efficacy of steganography is contingent upon its ability to remain undetected; some existing techniques may falter under close scrutiny, thereby compromising their concealing role. This project confronts this challenge head-on by introducing an innovative and secure image steganography technique that resists detection while facilitating reliable data communication. This paper proposes a novel approach named "Secure Image Steganography Framework" which undergirds a robust communication system, insulating it from unauthorized interception and access. The discussion extends to the delineation of the strengths and limitations of image steganography, as well as the spectrum of cyber threats it can adeptly mitigate, emphasizing the criticality of a fortified steganography technique in the vanguard of data security.

Introduction

In today's digital landscape, marked by frequent data breaches and unauthorized access, relying solely on conventional encryption techniques for data protection may be inadequate. This report investigates the project "Secure Data Communication using Image Steganography," addressing the pressing need for more sophisticated data security measures. It focuses on steganography, a technique that not only encrypts but also conceals the existence of data, thereby providing an extra layer of security. This study underscores the importance of steganography, which embeds information within other non-secret data or text, as a critical tool in safeguarding data communications in an age where information security is crucial.

Steganography Definition and Principles

Steganography, derived from the Greek words "steganos" (covered) and "graphein" (writing), involves concealing sensitive information within another medium to prevent its detection. Unlike encryption, which makes the content of a message unreadable, steganography hides the existence of the message itself. The underlying principle of computer steganography is that files, particularly those containing digitised images or sounds, can be slightly altered without losing their functionality. Human perception is generally unable to discern minor changes in image colour or sound quality, which steganography exploits.

Steganography Vs Cryptography

1. Concealment Vs. Encryption:

Steganography:

- Objective: The primary goal of steganography is to hide the very existence of the message.
- Method: It achieves this by embedding the message within other seemingly innocuous media, such as images, audio files, or even text, in a way that is imperceptible to human senses.
- Focus: The emphasis is on covert communication, making the information blend seamlessly with its surroundings, ensuring that unauthorized individuals are not even aware that a hidden message exists.

Cryptography:

- Objective: Cryptography, in contrast, is focused on securing the content of the message itself.
- Method: It achieves this by encrypting the original message (plaintext) using a mathematical algorithm, transforming it into an unreadable format known as ciphertext.
- Focus: The primary emphasis is on content protection through the use of encryption, ensuring that even if the encrypted message is intercepted, it remains unintelligible without the proper decryption key.

2. Detection Vs. Decryption:

Steganography:

- Detection Process: The detection process in steganography involves identifying subtle alterations or patterns within the carrier medium that may indicate the presence of hidden information. It aims to reveal the fact that covert communication is taking place.
- Objective: The goal is to uncover the hidden message and expose the use of steganographic techniques.

Cryptography:

- Decryption Process: Cryptography's decryption process involves using a specific decryption key to transform ciphertext back into plaintext.
- Objective: The primary objective is to prevent unauthorized access to the content of the message. Knowledge of the decryption key is essential for transforming the ciphertext into a readable format.

Types of Steganography

Steganography, the art of hiding information within plain sight, has evolved significantly with advancements in digital technology. This practice of concealing data within another medium ensures that the existence of the hidden information is unknown to a casual observer. Various forms of steganography use different media as carriers, with each type having its unique methods and applications suited to specific security needs and contexts.

1. Text Steganography

- Method: Concealing information within text by manipulating spaces, font size, or altering the format of letters and characters.
- Applications: Useful in areas where text is the primary mode of communication. Employed in emails, forums, or documents to transfer sensitive information discreetly.

2. Image Steganography

- Method: Involves embedding data within an image, commonly using techniques like the Least Significant Bit (LSB) modification, where bits of the image are replaced with bits of the secret message.
- Applications: Widely used due to the prevalence of digital images. Ideal for secure communication over the internet and social media.

3. Audio Steganography

- Method: Hides information within audio files by manipulating sound waves or embedding low-volume sounds within a louder audio signal.
- Applications: Useful in secure communication and watermarking audio files. Can be used in music, podcasts, or any audio medium.

4. Video Steganography

- Method: Involves embedding data into digital video files. Given the complexity of video files, this can be achieved by manipulating individual frames or integrating signals within the audio track.
- Applications: Beneficial for embedding large amounts of data. Used in film, multimedia, and television industries for secure data transfer and copyright protection.

5. Network Steganography

- Method: Conceals data within network control protocols and communication channels, like TCP/IP headers or VOIP.
- Applications: Used for covert communication over the internet. Helps in bypassing network security systems and firewalls.

Each of these types offers unique advantages and challenges. Text steganography, while simple, may offer less capacity for data. Image and audio steganography balance capacity with accessibility, as images and sounds are ubiquitous in digital communication. Video steganography offers the largest capacity but requires more processing power and sophistication. Network steganography, being the most technical, provides robust concealment but requires in-depth knowledge of network protocols.

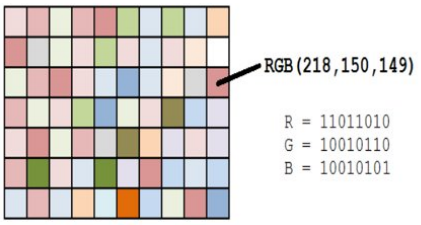
In essence, the choice of steganography type depends on factors like the amount of data to be concealed, the level of security required, the intended recipients, and the ease of transmission. The effectiveness of steganography lies in its discretion; the best-hidden message is the one whose existence is entirely unsuspected.

Image Steganography

Image steganography, the focus of this project, involves the process of hiding data within images. This technique is particularly potent because images are ubiquitous and typically do not arouse suspicion. The project explores ways to manipulate the pixel representation and colour values of an image to embed data. One of the most significant advantages of image steganography is its ability to transmit hidden messages or data without revealing the existence of the message, which is crucial for maintaining confidentiality in sensitive communications.

Applications of image steganography range from securing private files and documents to more complex scenarios like hiding passwords, encryption keys, or even transmitting highly confidential documents between governments. The effectiveness of this method lies in its ability to make changes to the image that are imperceptible to the human eye, thus maintaining the image's appearance while securely hiding data within it.

Techniques Used



In the project's application of image steganography, the predominant method employed is the Least Significant Bit (LSB) Substitution technique. This approach entails the modification of the least significant bits within the pixel values of an image to incorporate data. Given that each pixel is represented by bytes determining its color (red, green, and blue values), adjusting the least significant bit of these bytes induces a subtle color change that is typically imperceptible to human vision.

Advantages

Least Significant Bit (LSB) steganography is a crucial algorithm in the field of information security, renowned for its simplicity and effectiveness in embedding data within digital media, particularly images. The advantage of the LSB technique lies in its simplicity and the high degree of imperceptibility it offers. By making minor adjustments to pixel values, it becomes nearly impossible to detect the presence of hidden data without prior knowledge or specific analysis tools. This method strikes a balance between maintaining the original image's appearance and maximizing the amount of data that can be hidden.

Its significance stems from several key advantages and specific applications that make it preferable over other algorithms in certain scenarios.

Importance of LSB Steganography

Stealth and Camouflage: LSB steganography is proficient at discreetly integrating information. Through modifying the least significant bits of an image, it guarantees that the alterations remain undetectable to the human eye, preserving the image's initial look.

Ample Storage Capacity: In comparison to alternative steganographic methods, LSB can incorporate more extensive data volumes without causing noticeable changes to the host medium. This feature makes it well-suited for scenarios demanding the concealment of substantial information.

Ease and Availability: LSB is relatively uncomplicated to deploy and doesn't necessitate intricate computations, rendering it accessible to a diverse range of users and applications.

Versatility: It can be applied to different digital formats, including images, audio, and video, but it is most effective with lossless image formats like PNG or BMP, where quality and data integrity are preserved.

Where LSB is Preferable

Covert Communications: Intelligence and defense agencies may favor LSB steganography for hiding messages within images, as it allows for covert transmission of sensitive information, potentially bypassing surveillance systems.

Digital Watermarking: Content creators, media companies, and photographers find LSB steganography useful for embedding digital watermarks and copyrights. Its ability to hide these marks while maintaining the integrity of the original media is crucial for copyright protection.

Data Privacy: In industries prioritizing data privacy, such as healthcare and finance, LSB is an excellent choice for embedding confidential information within innocuous data, thereby safeguarding patient records or financial details.

Anti-Piracy Efforts: Software and game developers may use LSB steganography to incorporate unique identifiers or digital fingerprints into their products. This subtle embedding makes it a suitable tool for tracking and preventing piracy.

Cyber Forensics and Law Enforcement: Forensic experts and law enforcement agencies utilize LSB steganography to uncover hidden communications or track illegal activities in digital forensics investigations.

Robustness in Stable Environments: LSB is particularly effective in environments where the digital medium is not subject to compression or format conversion, as these processes can disrupt the steganographic data.

Comparison with Other Algorithms

Resilience to Detection: LSB is less detectable than more invasive methods, as it makes minimal changes to the carrier file.

Efficiency in Lossless Formats: While other algorithms might be more suited to lossy formats like JPEG, LSB stands out in lossless formats due to its ability to maintain data fidelity.

Ease of Implementation: Its simplicity gives it an edge over more complex algorithms, which may require more computational resources or specialized knowledge.

In summary, LSB steganography's importance lies in its ability to combine effective concealment, data capacity, simplicity, and adaptability, making it a preferred choice in various scenarios where security, privacy, and integrity are paramount. Its selection over other algorithms depends on the specific requirements of the task, particularly in contexts where subtlety and data volume are critical.

Proposed Methodology

The methodology for this project involves a step-by-step process where data is first encoded into an image using the LSB technique and then retrieved or decoded when needed. This process includes compressing the data to be hidden, thereby reducing its size and making it easier to embed without affecting the image's quality significantly.

1. User Interaction

Users are prompted to choose between encoding and decoding data.

For encoding, users must provide the source image and the text data to hide.

For decoding, users must provide the image with hidden data.

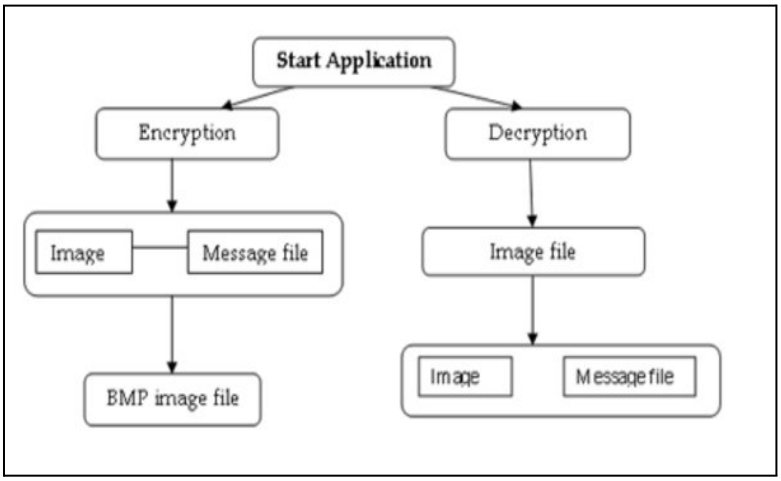
2. Execution Flow

Based on the user's choice, the corresponding function (encode or decode) is executed.

The process is interactive, with input prompts and outputs displayed in the Colab environment.

3. Result Display

Encoded and decoded images can be displayed in the notebook using IPython's Image display capabilities.



Encoding and Decoding Processes

The encoding procedure involves the selection of an image (referred to as the cover image) and the data intended for concealment within that image. Subsequently, the chosen data undergoes conversion into binary form, and this binary representation is then integrated into the pixels of the image. The decoding process reverses this procedure, extracting the concealed data from the image and thereby showcasing the efficacy of the steganographic method.

Algorithm Explanation

Encoding Process (genData, modPix, encode_enc, encode)

- ➢ Data Conversion: genData transforms the input text data into its binary format (8-bit ASCII).
- ➢ Pixel Modification: modPix adjusts the least significant bits of pixel values in the image to embed the binary data.
- ➢ Encoding Data: encode_enc utilizes the modified pixels to generate a new image containing the hidden data.
- ➢ Encode Function: The Encode function solicits user input, loads the image, invokes the encoding functions, and preserves the new image.

Decoding Process (decode)

- ➢ Extracting Data: decode reads the encoded image, extracts the least significant bits from each pixel, and reconstructs the hidden data in text form.

Code Snippets and Algorithm

The code is structured to encode (hide) data into an image and decode (retrieve) data from the image. Code snippets provided in the presentation illustrate the practical implementation of the LSB technique for both encoding and decoding data within an image.

```
from PIL import Image
import time
# Convert encoding data into 8-bit binary
# form using ASCII value of characters
def genData(data):
    # List of binary codes for each character
    newd = []
    # Iterate over each character in the input data
    for i in data:
        # Get the 8-bit binary representation of the ASCII value of the character
        newd.append(format(ord(i), '08b'))
    # Return the list of binary codes
    return newd

def modPix(pix, data):
    # Convert data to a list of binary strings
    binary_data = genData(data)
    data_len = len(binary_data)    # Get the length of the data list
    pixel_iter = iter(pix)  # Get an iterator for the image pixel data
    # Loop through each set of 3 pixels for each binary string in the data list
    for i in range(data_len):

        # Extract the next 3 pixels from the image pixel data
pixel_values = [value for value in pixel_iter.__next__()[:3] + pixel_iter.__next__()[:3] +
pixel_iter.__next__()[:3]]
        # Modify each pixel value based on the binary string
        for j in range(0, 8):
            if (binary_data[i][j] == '0' and pixel_values[j] % 2 != 0):
                pixel_values[j] -= 1
            elif (binary_data[i][j] == '1' and pixel_values[j] % 2 == 0):
                if (pixel_values[j] != 0):
                    pixel_values[j] -= 1
                else:
                    pixel_values[j] += 1



# Modify the eighth pixel of each set to indicate whether to continue reading or stop
        if (i == data_len - 1):
# If this is the last set of pixels, set the eighth pixel to 0 or 1 based on the parity of the
current value
            if (pixel_values[-1] % 2 == 0):
                if (pixel_values[-1] != 0):
                    pixel_values[-1] -= 1
                else:
                    pixel_values[-1] += 1
        else:
            # If this is not the last set of pixels, set the eighth pixel to 0 if it is odd
            if (pixel_values[-1] % 2 != 0):
                pixel_values[-1] -= 1
        # Yield the modified pixel values in sets of 3
        pixel_values = tuple(pixel_values)
        yield pixel_values[0:3]
        yield pixel_values[3:6]
        yield pixel_values[6:9]

def encode_enc(image, data):
    # Get the width of the image
    image_width = image.size[0]
    # Start at the top-left corner of the image
    x_coord = 0
    y_coord = 0
    # Loop over each modified pixel and add it to the new image
    for modified_pixel in modPix(image.getdata(), data):
        # Put the modified pixel in the new image
        image.putpixel((x_coord, y_coord), modified_pixel)
        # Move to the next row if at the end of the current row
        if x_coord == image_width - 1:
            x_coord = 0
            y_coord += 1
        else:
            x_coord += 1
```

```
# Encode data into image
def encode():
    # Get the filename of the image to be encoded
    image_filename = input("Enter image file name: ")
    # Open the image
    image = Image.open(image_filename, 'r')
    # Display the image
    image.show()
    # Get the data to be encoded
    data = input("Enter data to be encoded in the image: ")
# Raise an error if the data is empty
    if len(data) == 0:
        raise ValueError('Data is empty')
    # Create a copy of the image to modify
    new_image = image.copy()
    # Record the start time for encryption
    start_time = time.time()
    # Encode the data in the new image
    encode_enc(new_image, data)
    # Record the end time for encryption
    end_time = time.time()
    # Print the time taken for encryption
    print(f"Time taken for encryption: {end_time - start_time:.5f} seconds")
    # Get the filename of the new image
    new_image_filename = input("Enter the name of new image file to the decoded image: ")
    # Save the new image with the appropriate file format
    new_image.save(new_image_filename, str(new_image_filename.split(".")[1].upper()))


# Decode the data in the image
def decode():
    # Get the filename of the image to be decoded
    image_filename = input("Enter image name file to be decoded: ")
    # Record the start time for decryption
    start_time = time.time()
    # Open the image
    image = Image.open(image_filename, 'r')
    # Initialize an empty string to store the decoded data
    decoded_data = ''
    # Get an iterator for the image data
    image_data = iter(image.getdata())
    while True:
        # Get the RGB values of the next three pixels
        pixels = [value for value in image_data.__next__()[:3] +image_data.__next__()[:3] +
                            image_data.__next__()[:3]]
        # Convert the least significant bit of each pixel to binary and append to binstr
        binstr = ''
        for i in pixels[:8]:
            if i % 2 == 0:
                binstr += '0'
            else:
                binstr += '1'
        # Convert the binary string to a character and append to the decoded data
        decoded_data += chr(int(binstr, 2))
        # Check if the last pixel's least significant bit is 1
        if pixels[-1] % 2 != 0:
            # Record the end time for decryption
            end_time = time.time()
            # Print the time taken for decryption
            print(f"Time taken for decryption of the image: {end_time - start_time:.5f} seconds")
            return decoded_data


# Main Function
def main():
    a = int(input(":: Welcome to Steganography using Steganography:\n"
                  "1. Encode the data\n2. Decode the image\n"))
    if (a == 1):
        encode()
    elif (a == 2):
        print("Decoded Word from the image:  " + decode())
    else:
        raise Exception("Enter correct input file")
from IPython.display import Image
try:
  filename = take_photo()
  print('Saved to {}'.format(filename))

# Show the image which was just taken.
display(Image(filename))
except Exception as err:
# Errors will be thrown if the user does not have a webcam or if they do not
# grant the page permission to access it.
print(str(err))
```

Results

```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files  demo.png
• demo.png(image/png) - 1956612 bytes, last modified: 11/28/2023 - 100% done
Saving demo.png to demo.png
```

```
main()
```

```
:: Welcome to Steganography using Steganography:
1. Encode the data
2. Decode the image
1
Enter image file name: demo.png
Enter data to be encoded in the image: DataSecurityProject
Time taken for encryption: 0.00030 seconds
Enter the name of new image file to the decoded image: decdemo.png
```

```
from IPython.display import Image
Image('demo.png',width=500, height=500)
```

```
main()
```

```
:: Welcome to Steganography using Steganography:
1. Encode the data
2. Decode the image
2
Enter image name file to be decoded: decdemo.png
Time taken for decryption of the image: 0.05736 seconds
Decoded Word from the image:  DataSecurityProject
```

```
[12] from IPython.display import Image
     Image('decdemo.png',width=500, height=500)
```

Observations

Upon implementation, the results show that the changes made to the image are not discernible to the naked eye. This confirms the efficacy of the LSB technique in maintaining the visual integrity of the image while securely hiding the data.

Pixels/Words	<10 words	10-50	250	500	1000
256	0	0.0027	0.0159	0.0259	0.0754
720	0	0.0012	0.0165	0.0218	0.0349
1080	0.0011	0.0027	0.0097	0.0209	0.0340
1440	0.0011	0.0035	0.01100	0.0286	0.0606
2048	0.0015	0.0048	0.0137	0.0368	0.0914

Table 1: Time Taken to encrypt the message in Steganography

The table above shows how long it takes to hide (encrypt) a message within an image of various resolutions. The resolutions range from 256 pixels to 2048 pixels, and the message sizes range from less than 10 words to 1000 words.

For a small image of 256 pixels, the time taken to encrypt increases from 0 seconds for less than 10 words to 0.0754 seconds for 1000 words.

As the image resolution increases, the time taken to encrypt messages of the same length increases slightly. For instance, encrypting less than 10 words takes 0.0015 seconds in a 2048 pixel image.

The time taken grows with the length of the message, but not linearly. For example, encrypting a 250-word message in a 2048-pixel image takes 0.0137 seconds, while a 1000-word message takes 0.0914 seconds, which is less than seven times longer despite the message being four times longer.

Pixels/words	<10 words	10-50	250	500	1000
256	0.0096	0.0145	0.0116	0.0164	0.0194
720	0.0128	0.0126	0.0146	0.0172	0.0225
1080	0.0456	0.0468	0.0497	0.0723	0.0863
1440	0.0373	0.0646	0.0685	0.0692	0.0912
2048	0.0490	0.0753	0.0799	0.0815	0.1443

Table 2: Time Taken to decrypt the message in Steganography

This table records the time required to extract (decrypt) a hidden message from images of the same varying resolutions and message sizes as Table 1.

The time to decrypt is generally longer than to encrypt, especially as the resolution and word count increase.

For a 256-pixel image, decryption time starts at 0.0096 seconds for less than 10 words and goes up to 0.0194 seconds for 1000 words.

The time taken scales up with both image size and word count. For the largest image (2048 pixels), decrypting less than 10 words takes 0.0490 seconds, and decrypting 1000 words takes 0.1443 seconds.

Analysis

Decryption times are consistently longer than encryption times across all image resolutions and message sizes.

There is an increase in time taken with both an increase in the number of words and the size of the image. However, this increase is not directly proportional, indicating that the process might have a certain fixed overhead regardless of the message size.

Lower resolutions (256 and 720 pixels) show minimal change in encryption times as message sizes increase, suggesting that smaller images can be processed quickly even as the amount of data grows.

Higher resolutions show a more significant increase in time taken, indicating that the process becomes more resource-intensive as the amount of data and the image resolution increase.

Handling Large Message Sizes

The project also explores the impact of large message sizes on the encoding process. The time complexity of the encoding process can be considerably impacted by the size of the message. The message data is compressed during the encoding process so that it can be transmitted more quickly and with less storage space. However, when the message size grows, the compression algorithm will need more computing power to compress the data, which will make the process more time-consuming.

Moreover, problems with memory management can arise from greater message sizes. If the message size is too large, it might not fit within the memory that needs to be allocated by the encoding process to store the compressed message data. This may cause slower performance or, in some circumstances, crashes.

Technical and Practical Challenges

Additional factors to take into account include finding a balance between the quantity of data to be concealed and the image quality, along with considering the computational resources necessary for both the encoding and decoding processes.

Conclusion

This project successfully demonstrates the use of steganography, particularly image steganography, as a viable method for secure data communication. The implementation of the LSB technique shows that data can be effectively hidden within images, making it a useful tool in scenarios requiring high levels of confidentiality and data protection.

Future Work

Future research endeavors may delve into more sophisticated steganographic methods to further bolster security. One example is the Triple-A technique, which employs the LSB principle, concealing secrets in the least significant bits of pixels. However, it introduces increased randomization in selecting the number of bits and color channels, aiming to enhance system security. This method assesses pixel intensity and conceals data through a randomized pixel selection process, striving to maximize the amount of hidden data in each pixel. Additionally, there is a potential avenue for development in creating a user-friendly graphical user interface (GUI).

References

1. N. Subramanian, O. Elharrouss, S. Al-Maadeed and A. Bouridane, "Image Steganography: A Review of the Recent Advances," in IEEE Access, vol. 9, pp. 23409-23423, 2021, doi: 10.1109/ACCESS.2021.3053998.

2. W. Liu and J. Wang, "Research on image steganography information detection based on support vector machine," 2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP), Xi'an, China, 2021, pp. 631-635, doi: 10.1109/ICSP51882.2021.9408671.

3. A. A. J. Altaay, S. B. Sahib and M. Zamani, "An Introduction to Image Steganography Techniques," 2012 International Conference on Advanced Computer Science Applications and Technologies (ACSAT), Kuala Lumpur, Malaysia, 2012, pp. 122-126, doi: 10.1109/ACSAT.2012.25.

4. J. Qin, Y. Luo, X. Xiang, Y. Tan and H. Huang, "Coverless Image Steganography: A Survey," in IEEE Access, vol. 7, pp. 171372-171394, 2019, doi: 10.1109/ACCESS.2019.2955452.