AIO STEGANOGRAPHY TOOL

NADRA IBRAHIM

(Roll No.: 21MCA033**)**

DEPARTMENT OF COMPUTER SCIENCE

FACULTY OF NATURAL SCIENCES

JAMIA MILLIA ISLAMIA

May, 2023

AIO STEGANOGRAPHY TOOL

by

Nadra Ibrahim

(Enrollment No.: 20-2101210**)**

Submitted

in partial fulfilment of the requirements of the degree of

Master of Computer Applications (MCA)

to the

Department of Computer Science

Faculty of Natural Sciences

Jamia Millia Islamia

New Delhi - 110025

May, 2023

Declaration

I, **Nadra Ibrahim**, student of M.C.A hereby declare that the project report entitled **"AIO Steganography Tool"** which is submitted by me to the Department of Computer Science, Jamia Millia Islamia, New Delhi, in partial fulfilment of the requirement of the degree of **Master of Computer Applications (MCA)**, is a record of the original bonafide work carried out by me and have not been submitted in part or full to any other university or institute for the award of any degree or diploma.

(Nadra Ibrahim)

Roll No.: 21MCA033

Enrollment No.: 20-2101210

Certificate

On the basis of declaration made by the student **Nadra Ibrahim,** I/we hereby certify that the project report entitled **"AIO Steganography Tool"** submitted by **Nadra Ibrahim** to the Department of Computer Science, Jamia Millia Islamia, New Delhi, for the partial fulfilment of the requirements of the degree of **Master of Computer Applications (MCA)**, is carried out by her under my/our guidance and supervision. The report has reached the requisite standards for submission.

Prof. Suraiya Jabin

(Internal Supervisor)

Acknowledgement

I wish to express my sincere gratitude to **Mr. Himanshu Gautam, Senior Executive** , for providing me an opportunity to do my internship and project work in **"Knoldus Inc. (Part of Nashtech)"**.

I would also like to thank **Prof. Suraiya Jabin** for providing me the opportunity to embark on this project and for his continuous guidance and encouragement in carrying out this project work.

(Nadra Ibrahim)

Roll No.: 21MCA033

Enrollment No.: 20-2101210

Table of Contents

UNIT 1:

PROBLEM DEFINITION

1.1 INFORMATION HIDING TECHNIQUES

❖ **Information Hiding :**

Information/Data hiding is an act of protecting information/data from any inadvertent change. It can also be defined as the process of hiding the details of an object. A common use of information hiding is to hide the data so that if it is changed, the change is restricted to a small subset of the total data. It helps to prevent programmers from intentionally or unintentionally changing parts of a program.

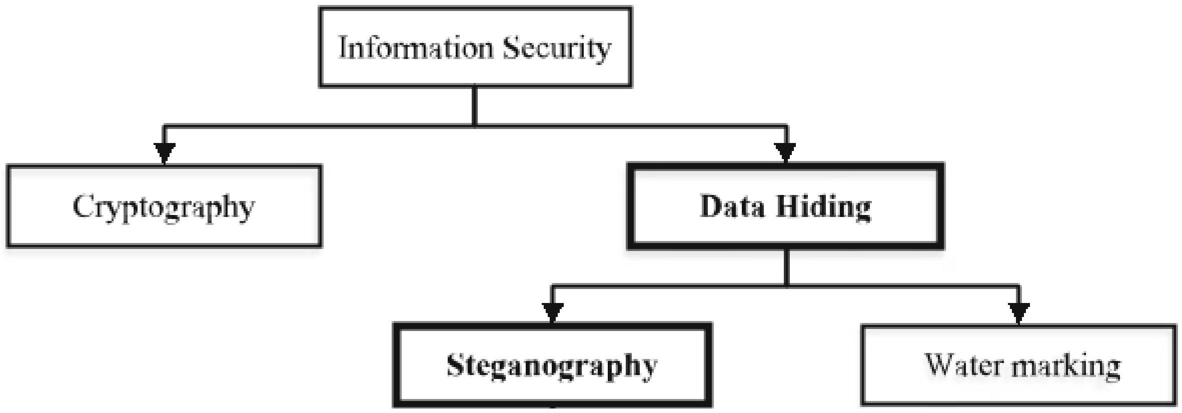


Figure 1 : Information Hiding Techniques

❖ **Steganography :**

It is a science of hiding information from unauthorized person. Steganography is a practice of concealing a file, message, image, or video within another file, message, image, or video.The secret message and the cover message are first fed to encoder using steganography techniques and then the same encryption algorithm in reverse order is implemented to recover the original secret message and the original cover. Key plays very important role for retrieving the original message.
The advantage of steganography is that the message to be transmitted is not detectable to the casual eye. In fact, unauthorized people should not even suspect that a hidden message exists. Digital steganography is the art ofhiding data within data. Goal of steganography is to hide data well enough that unintended recipients do not
suspect the steganography medium of containing hidden data . Media files are ideal for steganography transmission because of their large size.

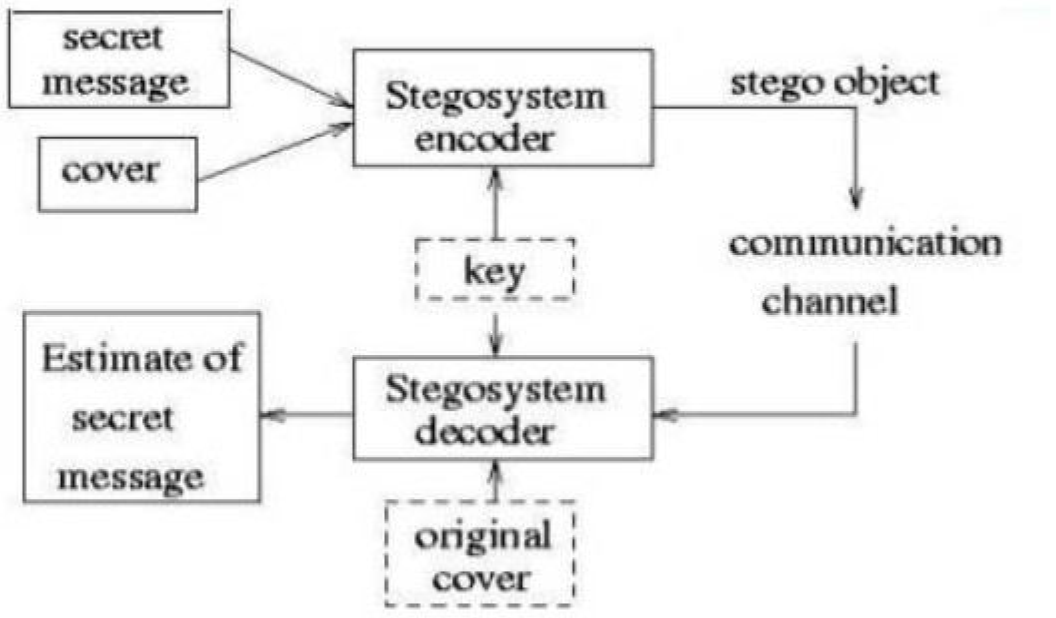


Figure 2 : Process of Steganography

❖ **Cryptography:**

Cryptography is the practice of securing information by transforming it into an unintelligible form to protect its confidentiality, integrity, and authenticity. Cryptographic techniques involve using algorithms and keys to convert plaintext data into ciphertext, making it unreadable to unauthorized parties. The recipient with the correct decryption key can then revert the ciphertext back to its original form.

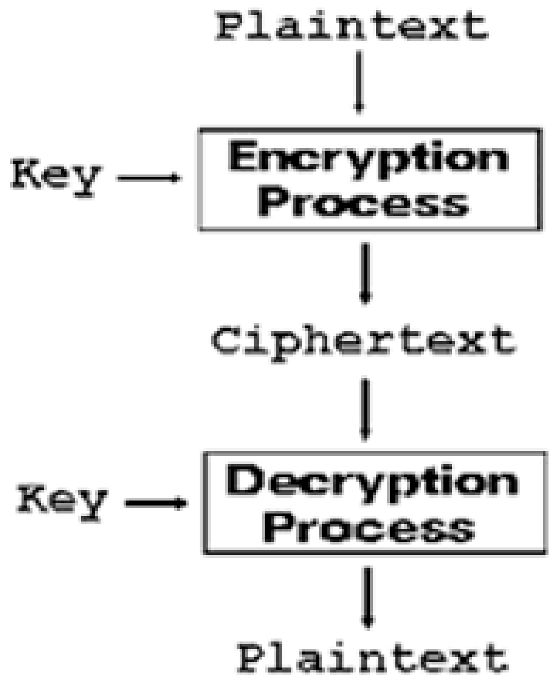


Figure 3 : Cryptography Process

❖ **Watermarking :**

Watermarking is a technique used to embed a visible or invisible mark or label into a media file, such as images, videos, or audio. The purpose of watermarking is to claim ownership, provide copyright protection, or indicate the authenticity of the media content. Watermarks are typically visible and do not attempt to hide or conceal their presence. Watermarks can be easily detected and are not designed to be secret or covert.

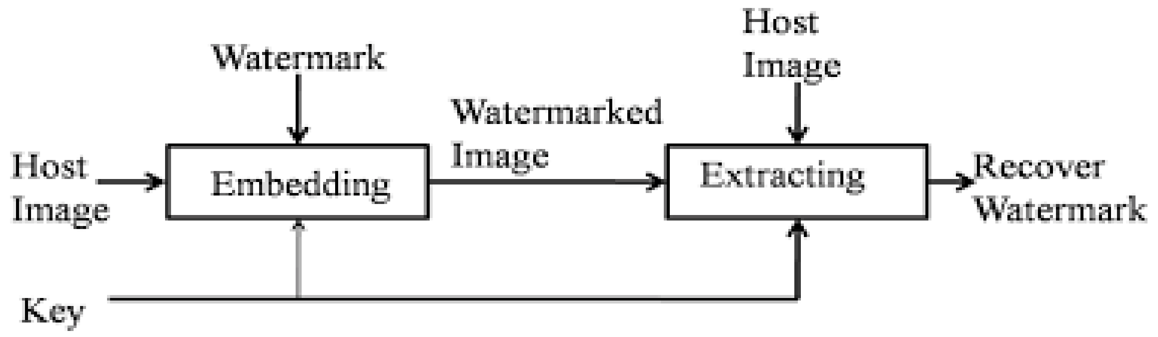


Figure 3 : Watermarking Process

❖ **Steganography VS Cryptography**

Steganography and cryptography are both techniques used to protect information, but they differ in their approaches and goals. Here are the main differences between steganography and cryptography:

1. Goal: The goal of cryptography is to encrypt data and make it unreadable to unauthorized individuals. It focuses on ensuring the confidentiality, integrity, and authenticity of the information. On the other hand, steganography aims to hide the existence of the data by concealing it within other non-secret data. The primary objective of steganography is to achieve covert communication or information hiding.

2. Concealment: Cryptography achieves concealment by transforming the original data using encryption algorithms, resulting in ciphertext that can only be decrypted with the appropriate key. It does not hide the fact that encryption is being used. In contrast, steganography hides the data within a cover medium, such as an image or audio file, making it appear unchanged and indistinguishable from non-secret data. The focus is on hiding the presence of the information rather than making it unreadable.

3. Detection: Cryptography assumes that the encrypted data is known to exist, and efforts are made to protect the encryption keys and prevent unauthorized decryption. Cryptanalysis is the process of attempting to break the encryption and uncover the original message. In steganography, the goal is to make the presence of the hidden data undetectable. It relies on the fact that the changes made to the cover medium are imperceptible to human senses, making it difficult to detect the hidden information without specialized steganalysis techniques.

4. Scope: Cryptography operates at the level of the entire message or data, securing its content as a whole. It focuses on mathematical algorithms and key management to protect data during storage or transmission. Steganography, on the other hand, operates at a more granular level, embedding data within a cover medium. It conceals the information within the file, allowing it to be hidden in plain sight.

In practice, **cryptography and steganography can be used together to provide enhanced security**. Data can be encrypted using cryptography, and then steganography can be applied to hide the encrypted data within a cover medium, adding an additional layer of concealment. This combination can provide both confidentiality and covert communication.

❖ Steganography VS Watermarking :

Watermarking and steganography are both techniques used for embedding information into media files, but they serve different purposes and have distinct characteristics.The main differences between watermarking and steganography are:

1. Purpose: Watermarking is primarily used for ownership attribution and copyright protection, while steganography focuses on hiding and concealing information.

2. Visibility: Watermarks are often visible and meant to be noticed, whereas steganographic data is typically invisible or imperceptible.

3. Detection: Watermarks can be easily detected and decoded, whereas steganography aims to make the hidden data difficult to detect without knowledge of the specific encoding technique.

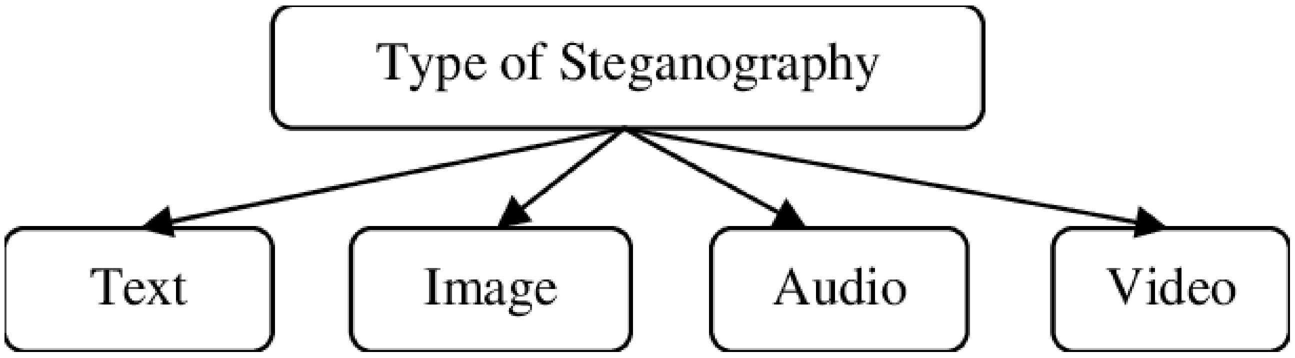
1.2 TYPES OF STEGANOGRAPHY

❖ **AIO Steganography :**

An AIO (All-in-One) steganography application is a specific type of steganography tool that encompasses various steganography techniques and features in a single application. AIO Steganography specifically refers to a comprehensive approach that encompasses various methods and algorithms for embedding and extracting hidden information in different types of files, such as images, audio, video, and even text documents. It provides a range of options and techniques for hiding data, depending on the specific file format and requirements.
Steganography can be categorized into different types based on the cover object or medium used for embedding the hidden information. Here are some common types of steganography based on the cover object:

1. Image Steganography
2. Audio Steganography
3. Video Steganography
4. Text Steganography

and many more.



a Text Steganography

Text Steganography can involve anything from changing the formatting of an existing text, to changing words within a text, to generating random character sequences or using context-free grammars to generate readable texts. Text steganography is believed to be the trickiest due to deficiency of redundant information which is present in image, audio or a video file. The structure of text documents is identical with what we observe, while in other types of documents such as in picture, the structure of document is different from what we observe. Therefore, in such documents, we can hide information by introducing changes in the structure of the document without making a notable change in the concerned output. Unperceivable changes can be made to an image or an audio file, but, in text files, even an additional letter or punctuation can be marked by a casual reader. Storing text file require less memory and its faster as well as easier communication makes it preferable to other types of steganographic methods.

b Image Steganography

Image steganography is a technique used to hide information within digital images. It involves embedding secret data into the pixels of an image in a way that is imperceptible to the human eye. The primary goal of image steganography is to conceal the presence of the hidden data within the image while maintaining the visual quality and integrity of the image. Here are some common techniques used in image steganography:

1 **Least Significant Bit (LSB) substitution:** This is one of the simplest and most widely used methods. It involves replacing the least significant bits of the pixel values with the bits of the hidden message. Since the LSBs have less impact on the overall pixel value, the changes are often visually undetectable.

2 **Pixel value differencing:** This technique takes advantage of the variations between adjacent pixel values in an image. The differences between pixel values are modified to encode the hidden data. By analyzing the differences in pixel values, the hidden information can be extracted during decoding.

3 **Spread Spectrum**: This method spreads the hidden information across the image using techniques similar to those used in digital communications. The data is distributed by modifying the pixel values across multiple spatial locations or frequencies within the image. The changes are carefully controlled to minimize visual artifacts.

4 **Transform domain steganography:** Transform domain steganography involves applying mathematical transformations, such as Discrete Cosine Transform (DCT) or Discrete Wavelet Transform (DWT), to the image. The hidden data is embedded by modifying the transform coefficients. Transform domain techniques offer a higher capacity for hiding information while preserving image quality.

5 **Palette-based steganography:** This method is applicable to indexed color images with a limited palette. It involves modifying the color lookup table or palette values to embed the hidden data. The changes in the palette can be used to encode the information.

c Audio Steganography

Audio steganography is a method of hiding information within an audio file without altering its perceptual quality. It allows for the covert transmission or storage of data within audio signals. The hidden information can be in the form of text, images, or other types of data. Here are a few common techniques used in audio steganography :

1 **Least Significant Bit (LSB) insertion:** This is one of the simplest and most widely used techniques. It involves replacing the least significant bits of audio samples with the bits of the hidden message. Since the least significant bits have less impact on the audio quality, the changes are often imperceptible to the human ear.

2 **Spread Spectrum:** This technique spreads the hidden message across a wide frequency range in the audio signal. The information is embedded by adding a low-power signal to the original audio, using techniques like frequency hopping or direct sequence spread spectrum. The added signal is chosen to be masked by the original audio, making it difficult to detect.

3 **Phase coding:** This method exploits the phase information of audio signals to encode hidden data. By manipulating the phase of selected audio samples, the hidden information is embedded. Since human perception is less sensitive to phase changes, this technique can achieve good concealment.

4 **Echo hiding:** Echo hiding involves embedding information in the echoes of an audio signal. Echoes are created when sound waves reflect off surfaces in an environment. By manipulating the delay and amplitude of these echoes, hidden data can be encoded within the audio.

5 **Steganography in audio compression algorithms:** Audio compression algorithms like MP3 or AAC can be exploited to embed hidden data. By taking advantage of the way these algorithms compress audio, additional information can be inserted into the compressed file without noticeable degradation in audio quality.

d Video Steganography

Video steganography is a technique used to hide information within a video file. It allows for the covert transmission or storage of data within the frames or components of a video, making it difficult to detect without prior knowledge or the use of specific decoding algorithms. Similar to other forms of steganography, video steganography aims to conceal the presence of the hidden data while maintaining the perceptual quality and visual integrity of the video. Various techniques can be employed for video steganography, including:

1 **Frame-level steganography:** This method involves hiding information within individual frames of the video. It can be achieved by modifying specific pixels or regions in the frames to encode the hidden data. Techniques like LSB substitution or modification of least significant bits are commonly used for frame-level steganography.

2 **Temporal steganography:** Temporal steganography exploits the temporal relationship between video frames to embed hidden data. Information can be encoded by subtly modifying the timing or sequence of frames, such as altering frame order or adjusting timestamps.

3 **Transform domain steganography:** This technique operates in the frequency or transform domain of the video. It involves applying transformations like Discrete Cosine Transform (DCT) or Discrete Wavelet Transform (DWT) to video frames and modifying the transform coefficients to hide the information. Transform domain steganography can provide a higher capacity for embedding data while maintaining visual quality.

4 **Audio-video synchronization:** Video files often include audio tracks. Steganographic techniques can be applied to the audio component of the video to hide information, while the visual component remains unaffected. The synchronization between audio and video is crucial to ensure the concealed data can be properly extracted during decoding.

1.3. PROBLEM STATEMENT

In the digital age, the widespread use of digital media and the Internet has created a need for covert communication and secure data transmission. Steganography came into the picture as a response to these needs, driven by the following factors:

1. Encryption detection: While encryption provides a secure way to protect sensitive data, its use can raise suspicion and draw attention from authorities or adversaries. Steganography offers an alternative approach by hiding the existence of the communication itself, making it difficult to detect or even recognize that a hidden message is being transmitted.

2. Covert operations and espionage: Governments, intelligence agencies, and law enforcement organizations require covert means of communication to carry out their operations effectively. Steganography provides a way to exchange information discreetly, minimizing the risk of interception or exposure.

3. Privacy and confidentiality: With the increasing prevalence of surveillance and data breaches, individuals and organizations seek methods to protect their privacy and ensure the confidentiality of their sensitive information. Steganography allows for the covert transmission or storage of data within innocuous digital files, making it challenging for unauthorized individuals to access or interpret the hidden information.

4. Anti-censorship and circumvention: In regions where internet censorship or surveillance is prevalent, individuals or groups may face restrictions on their ability to communicate or access certain information. Steganography can be used to bypass such restrictions and transmit information undetected, ensuring the free flow of information and preserving privacy.

5. Digital watermarking and copyright protection: With the ease of copying and distributing digital media, content creators and rights holders face challenges in protecting their intellectual property. Steganography techniques are utilized in digital watermarking to embed identifiers within media files, helping to deter unauthorized use and track copyright infringements.

UNIT 2: APPLICATION OVERVIEW

2.1. OBJECTIVE

The objective of an All-in-One (AIO) steganography tool is to provide a comprehensive and versatile solution that combines multiple steganography techniques and functionalities in a single platform. Here are the key objectives of an AIO steganography tool:

1. Multi-Technique Integration: The AIO steganography tool aims to integrate various steganography techniques, such as image steganography, audio steganography, video steganography, text steganography, network steganography, and file steganography. It offers a wide range of options for users to choose from based on their specific needs and the type of cover object they want to use.

2. Seamless Workflow: The tool ensures a seamless and consistent user experience across different steganography techniques. It provides a unified interface that allows users to easily switch between techniques, embed and extract hidden information, and customize settings without the need for separate tools or applications.

3. Cross-Media Support: An AIO steganography tool supports embedding and extracting hidden information across various types of media, including images, audio files, videos, text documents, network protocols, and more. It provides flexibility and convenience for users to work with different media formats within a single tool.

4. Enhanced Security Features: The tool may incorporate additional security features to enhance the protection of hidden information. This can include encryption capabilities for the embedded data, password protection for accessing the hidden information, or digital signatures for ensuring data integrity and authenticity.

5. Steganalysis Resistance: An AIO steganography tool employs advanced techniques and algorithms to minimize the chances of steganalysis and detection. It utilizes methods that counter common steganalysis techniques, making it challenging for adversaries to discover the presence of hidden information.

6. User-Friendly Interface: The AIO steganography tool aims to provide a user-friendly and intuitive interface that simplifies the process of working with different steganography techniques. It offers clear instructions, guides users through the necessary steps, and presents options in an organized and understandable manner.

7. Compatibility and Performance: The AIO steganography tool should be compatible with different operating systems and exhibit efficient performance. It should support various file formats, provide fast processing speeds, and deliver reliable results to ensure a smooth and efficient user experience.

2.2. APPLICATION DESCRIPTION

AIO Steganography Tool is a desktop-based application that provides a comprehensive set of features for hiding data within various media files, such as images, audio, or video.
The Application starts with a home window, having three buttons :

1 Encode

2 Decode

3 Info

BUTTON 1 : ENCODE :

The encode window opens up a new frame of window having two labels :

1 **Select cover media** : It shows three radio buttons, out of which one is to be selected by the user for the cover media in which the the secret information will be embeded.

- Image
- Video
- Audio

Then, the file is to be selected by pressing the **Open** button.
If file is selected, the path will be shown below the open button.

2 **Select input media** : This label specifies two radio buttons, which remains disable as long as the cover media is not selected.

If Image is selected as the cover media, there are two options of selecting the input media, that are :

- Image : The dimension of the image is checked to be compatible with cover image. If there is no compatibility issue, then the input secret image will be encoded into the cover image and the stego-image will be stored in the same path as **“image_image_out.png”.**
- Text : The text button enables the message area and on pressing encode button, the secret key is required to encrypt the text, then the cypher text is encoded into the cover image and the stego-image is saved as **“image_text_out.png”**

Else if Audio or Video is selected as the cover media, there are only one option of selecting the input media, that is :

- Text : For embedding text into audio, random lsb subsitution algorithm is used and the file is saved as **“audio_text_out.wav”.** and for encoding text into video, rc4 encryption is performed on the text and then the cipher text is embeded into the video frames and the finally stego-video file is saved as **“video_text_out.flv” .**

BUTTON 2 : DECODE :

The decode window opens up a new window having four radio buttons :

1. Text from Image : The Stego-image is selected by the user and on pressing decode button, the secret text will be decoded from the image and will be displayed.

2. Image from Image : The Stego-image is selected by the user and on pressing decode button, the secret mage will be decoded from the image. The secret image will be saved in the current working directory as “Imageoutput.png”. The user can open the image from the file browser.

1. Text from Audio : The Stego-audio file of format (.wav) is selected by the user and on pressing decode button, the secret text will be decoded and displayed.

1. Text from Video : The Stego-video file of format (.flv) is selected by the user and on pressing decode button, the secret text will be decoded and displayed.

BUTTON 3 : INFO:

The info frame will display the necessary information of the application :

1. Developer Info : Name, Version, Contact info, etc.

2. Steganography Significance

2. Application Working : Data flow, Algorithm, etc.

2.3. JUSTIFICATION AND NEED FOR THE APPLICATION

An "all-in-one" steganography application can provide several benefits and address various needs for individuals or organizations involved in secure communication or data protection. Here are some justifications and needs for such an application:

1. Convenience and Ease of Use: Having a single application that integrates multiple steganography techniques simplifies the process for users. They don't need to switch between different tools or learn multiple software interfaces, making it more convenient and user-friendly.

2. Enhanced Security: By incorporating multiple steganography techniques, an all-in-one application can provide stronger security and resistance to detection. Users can combine different methods to achieve a higher level of secrecy, making it more challenging for unauthorized individuals to discover and extract hidden information.

3. Protecting Sensitive Data: Individuals or organizations may need to protect sensitive or confidential information during transmission or storage. An all-in-one steganography application can serve as a reliable tool for embedding sensitive data within innocuous media files, adding an extra layer of security to prevent unauthorized access.

4. Communication and Collaboration: Steganography applications can enable secure communication and collaboration among individuals or teams. With the ability to hide messages within media files, users can exchange information discreetly, especially in situations where privacy is crucial, such as in journalism, activism, or sensitive research.

5. Educational and Research Purposes: An all-in-one steganography application can serve as a valuable tool for educational purposes or research in the field of information security. It allows students, researchers, or enthusiasts to explore different steganography techniques, understand their strengths and weaknesses, and contribute to the advancement of the field.

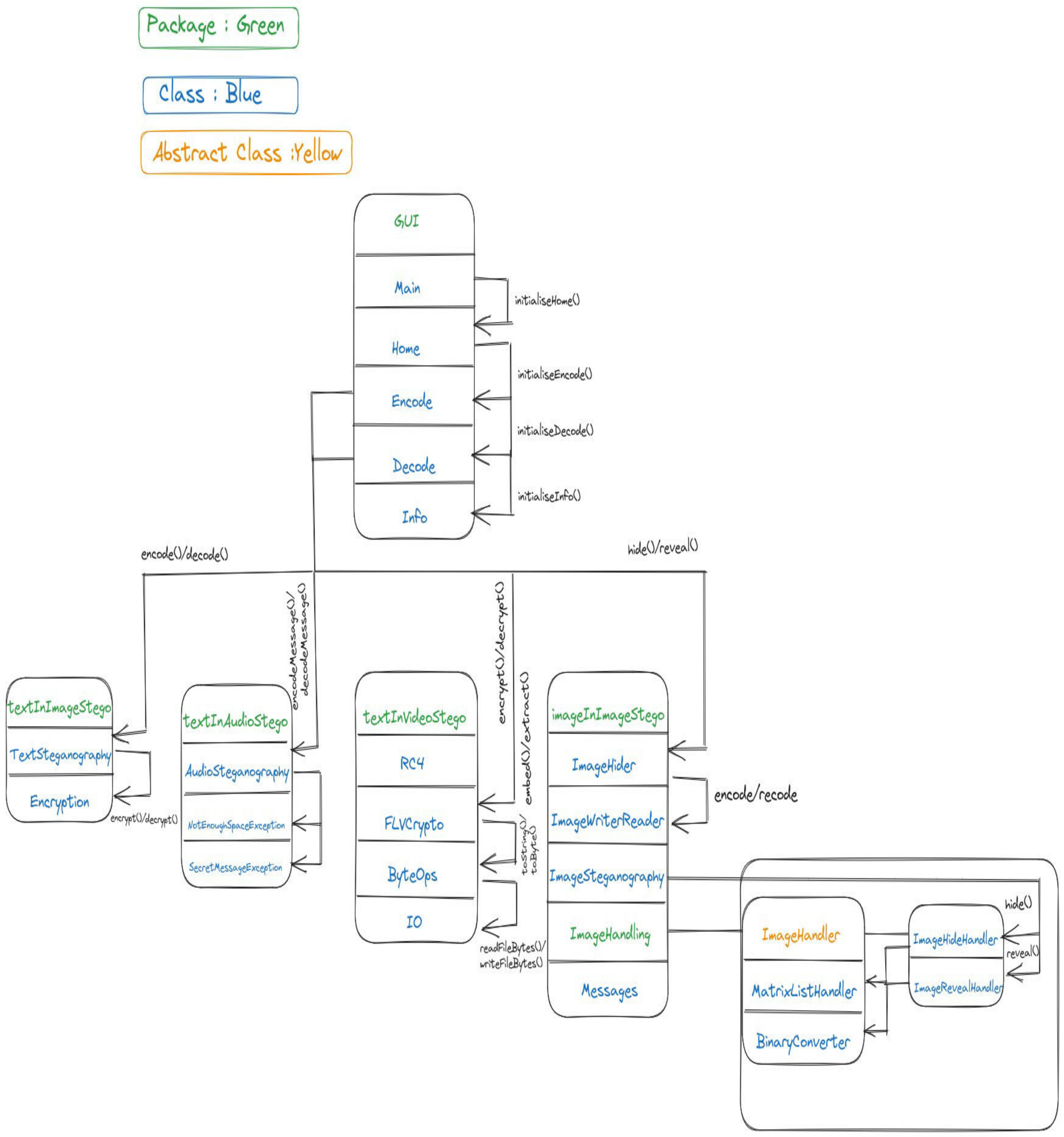
2.4. LITERATURE SURVEY

2015: G. Prashanti and K. Sandhyarani have done survey on recent achievements of LSB based image steganography. In this survey authors discuss the improvements that enhance the steganographic results such as high robustness, high embedding capacity and un-detectability of hidden information. First technique is used to embed data or secret messages into the cover image and in the second technique a secret gray scale image is embedded into another gray scale image.
2014-2015: Savita Goel proposed a new method of embedding secret messages in cover image using LSB method using different progressions such as Peak Signal to Noise Ratio (PSNR), Mean Square Error (MSE), histograms and CPU time, Structure Similarity (SSIM) index and Feature Similarity Index Measure (FSIM). Their study and experimental results shows that their proposed method is fast and highly efficient as compared to basic LSB methods.
2015: Bingwen Feng, Wei Lu, and Wei Sun in their paper "Secure Binary Image Steganography Based on Minimizing the Distortion on the Texture" purposed a state of-the-art approach of binary image steganography. This technique is proposed to minimize the distortion on the texture. In this method of steganography firstly the rotation, complement and mirroring invariant texture patterns are extracted from the binary image. They also proposed a measurement and based on this proposed measurement this approach is practically implemented.
2014: On the based on Huffman Coding, Amitava Nag present a novel steganographic technique of LSB substitution. Their technique basically focuses on high security, larger embedding capacity and acceptable level of stego image quality. Firstly Huffman tree is produced to encode every 8 bits of secret image. After encoding, they divide the encoded bits into four parts and have 0 to 3 decimal values. Location of embedding a message in cover image is determined by these decimal values. Experimental results show that it is very difficult for attacker to extract the secret information because Huffman table decrease the size of the cover image. This method reduces the leap in colour scale because only blue
components are used to embed the secret information.
2010: On the bases of Human visual system (HVS) X. Qing proposed a new technique in which sensitive information is embedded in all planes of RGB components of an image. In this technique multiple plan-bit is used with adaptive nature of information hiding algorithm. This proposed method has high embedding capacity than traditional LSB method and low computational complexity. Proposed system also has good quality of stego image.

UNIT 3:

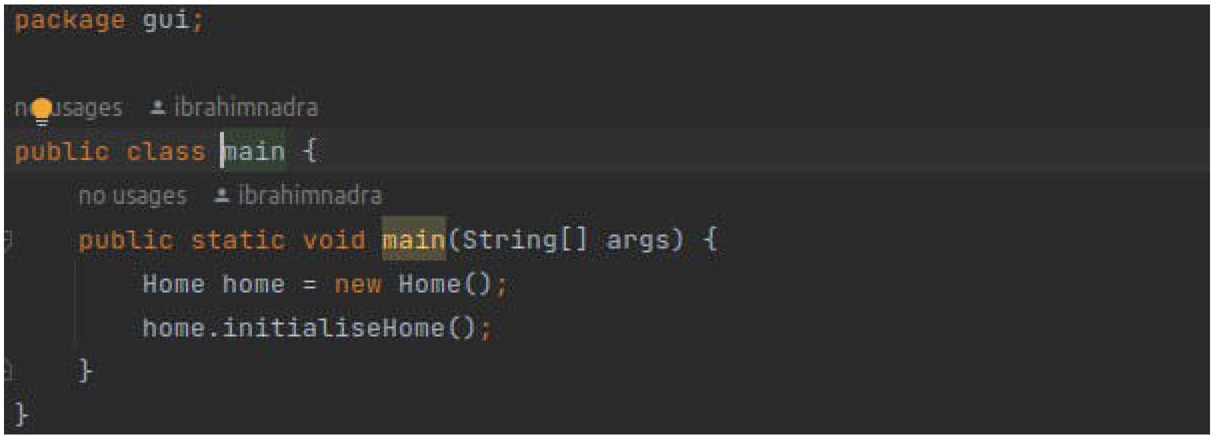
IMPLEMENTATION

3.1. APPLICATION'S ARCHITECTURE



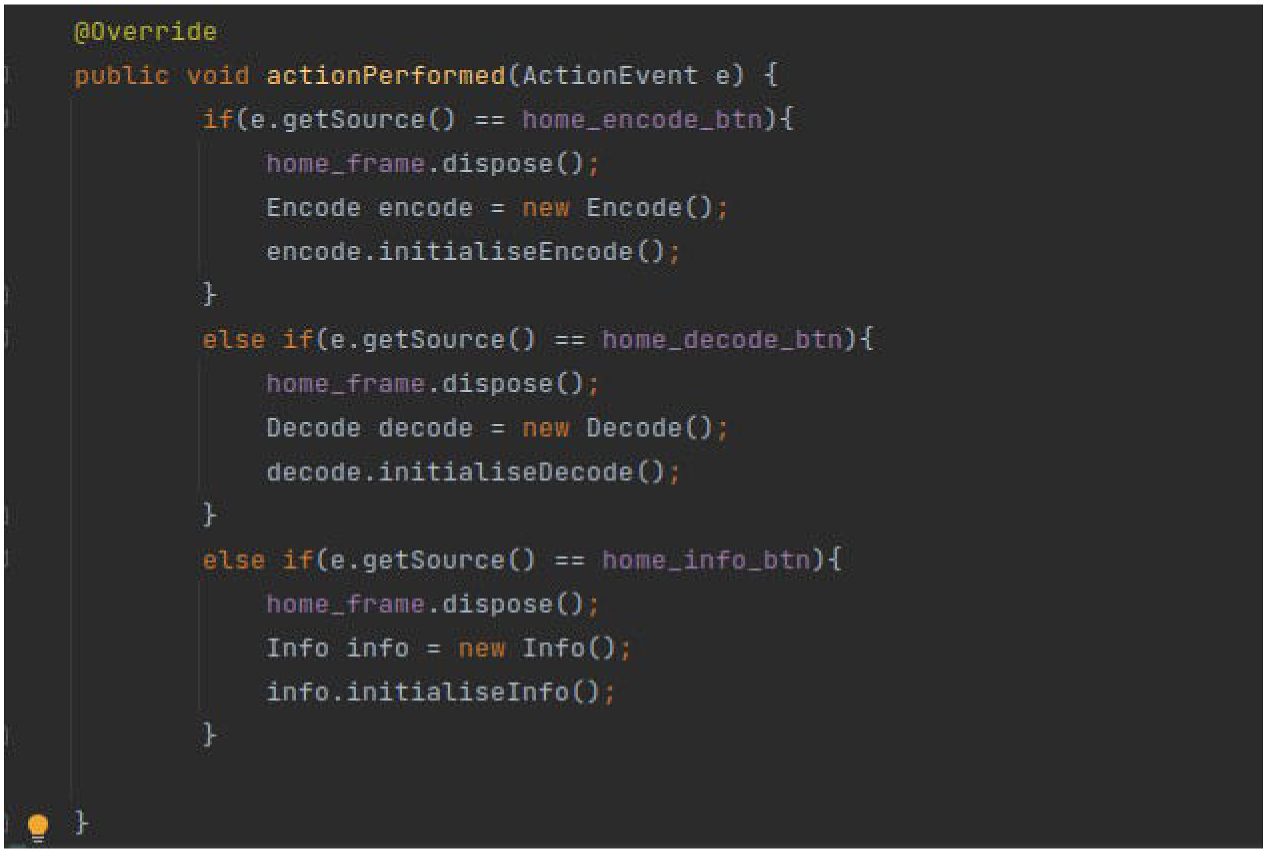
3.2. GUI CODE

CLASS 1 : MAIN



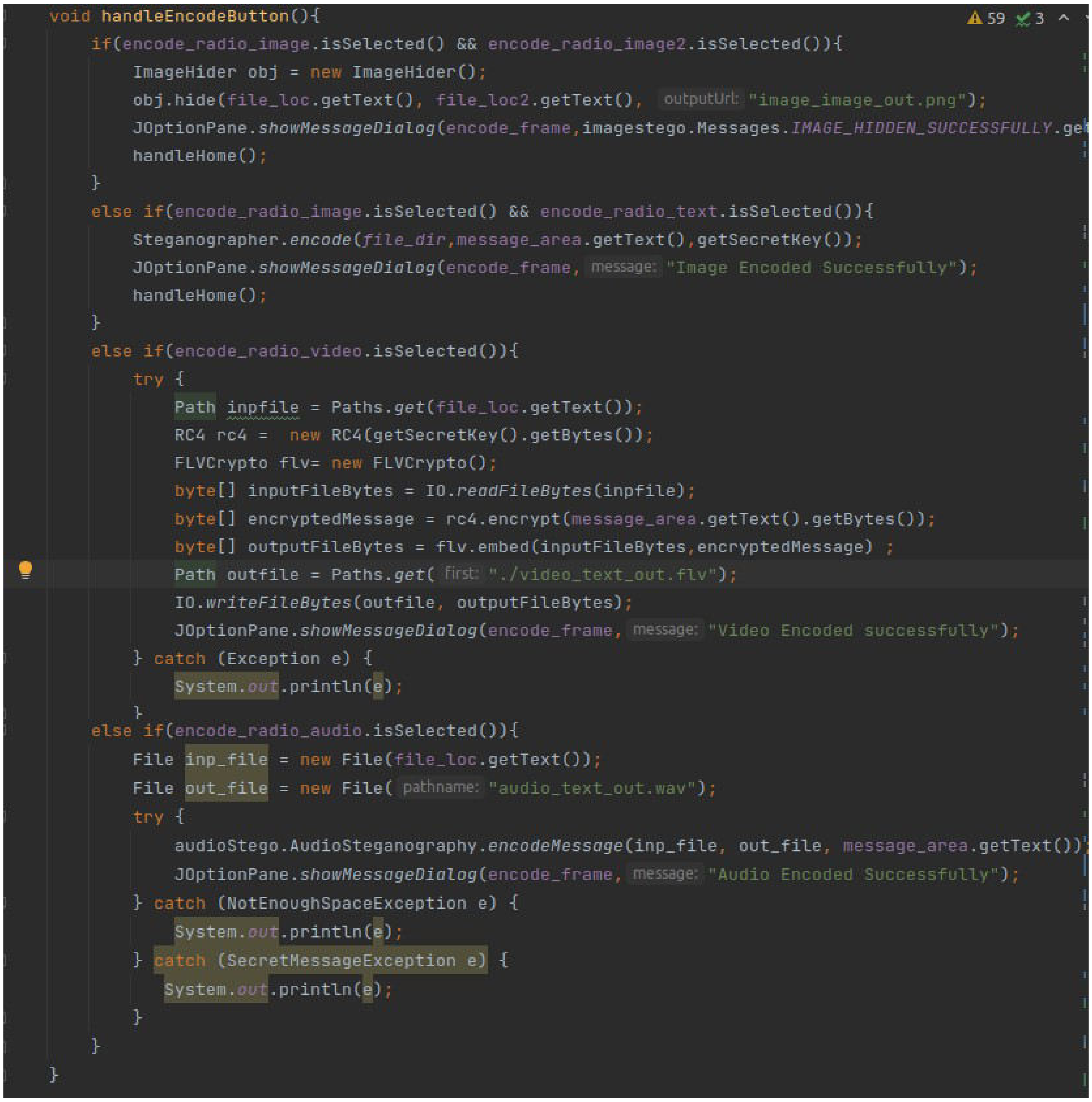
```
package gui;

no usages  ibrahimnadra
public class main {
    no usages  ibrahimnadra
    public static void main(String[] args) {
        Home home = new Home();
        home.initialiseHome();
    }
}
```

CLASS 2 : HOME : actionPerformed()



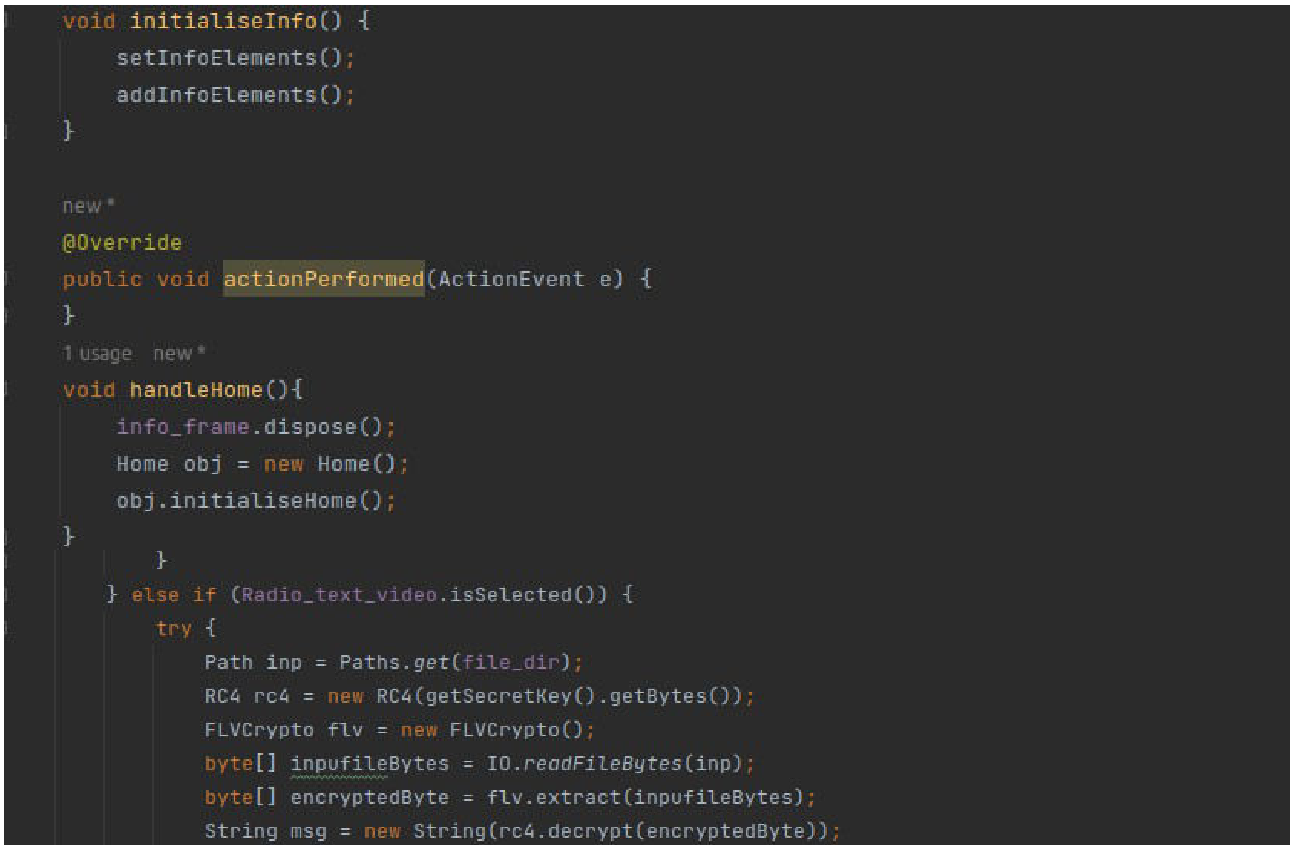
```
    @Override
    public void actionPerformed(ActionEvent e) {
            if(e.getSource() == home_encode_btn){
                home_frame.dispose();
                Encode encode = new Encode();
                encode.initialiseEncode();
            }
            else if(e.getSource() == home_decode_btn){
                home_frame.dispose();
                Decode decode = new Decode();
                decode.initialiseDecode();
            }
            else if(e.getSource() == home_info_btn){
                home_frame.dispose();
                Info info = new Info();
                info.initialiseInfo();
            }

    }
```

CLASS 3 : ENCODE : handleEncodeButton()



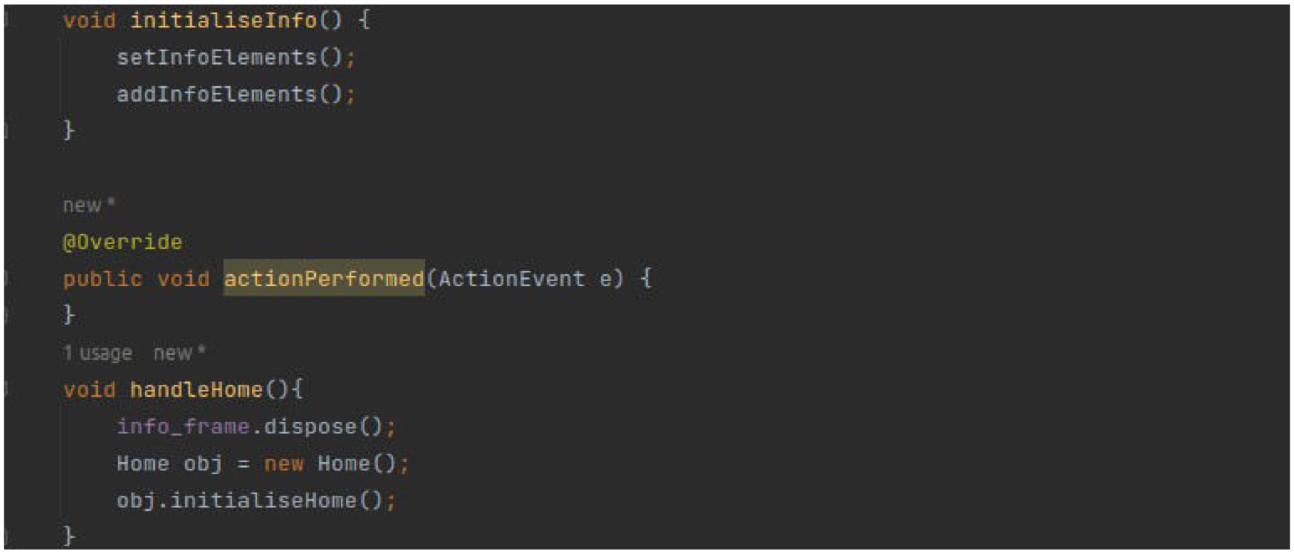
```
void handleEncodeButton(){
    if(encode_radio_image.isSelected() && encode_radio_image2.isSelected()){
        ImageHider obj = new ImageHider();
        obj.hide(file_loc.getText(), file_loc2.getText(),  outputUrl: "image_image_out.png");
        JOptionPane.showMessageDialog(encode_frame,imagestego.Messages.IMAGE_HIDDEN_SUCCESSFULLY.ge
        handleHome();
    }
    else if(encode_radio_image.isSelected() && encode_radio_text.isSelected()){
        Steganographer.encode(file_dir,message_area.getText(),getSecretKey());
        JOptionPane.showMessageDialog(encode_frame,  message: "Image Encoded Successfully");
        handleHome();
    }
    else if(encode_radio_video.isSelected()){
        try {
            Path inpfile = Paths.get(file_loc.getText());
            RC4 rc4 =  new RC4(getSecretKey().getBytes());
            FLVCrypto flv= new FLVCrypto();
            byte[] inputFileBytes = IO.readFileBytes(inpfile);
            byte[] encryptedMessage = rc4.encrypt(message_area.getText().getBytes());
            byte[] outputFileBytes = flv.embed(inputFileBytes,encryptedMessage) ;
            Path outfile = Paths.get( first: "./video_text_out.flv");
            IO.writeFileBytes(outfile, outputFileBytes);
            JOptionPane.showMessageDialog(encode_frame,  message: "Video Encoded successfully");
        } catch (Exception e) {
            System.out.println(e);
        }
    else if(encode_radio_audio.isSelected()){
        File inp_file = new File(file_loc.getText());
        File out_file = new File( pathname: "audio_text_out.wav");
        try {
            audioStego.AudioSteganography.encodeMessage(inp_file, out_file, message_area.getText())
            JOptionPane.showMessageDialog(encode_frame,  message: "Audio Encoded Successfully");
        } catch (NotEnoughSpaceException e) {
            System.out.println(e);
        } catch (SecretMessageException e) {
           System.out.println(e);
        }
    }
}
```

CLASS 4 : DECODE : actionPerformed()



```
    void initialiseInfo() {
        setInfoElements();
        addInfoElements();
    }

    new *
    @Override
    public void actionPerformed(ActionEvent e) {
    }
    1 usage   new *
    void handleHome(){
        info_frame.dispose();
        Home obj = new Home();
        obj.initialiseHome();
    }
            }
        } else if (Radio_text_video.isSelected()) {
            try {
                Path inp = Paths.get(file_dir);
                RC4 rc4 = new RC4(getSecretKey().getBytes());
                FLVCrypto flv = new FLVCrypto();
                byte[] inpufileBytes = IO.readFileBytes(inp);
                byte[] encryptedByte = flv.extract(inpufileBytes);
                String msg = new String(rc4.decrypt(encryptedByte));
```

CLASS 5 : INFO : actionPerformed()



```
    void initialiseInfo() {
        setInfoElements();
        addInfoElements();
    }

    new *
    @Override
    public void actionPerformed(ActionEvent e) {
    }
    1 usage   new *
    void handleHome(){
        info_frame.dispose();
        Home obj = new Home();
        obj.initialiseHome();
    }
```

2.3.

3.3. STEGANOGRAPHY TECHNIQUES USED

1. Image Steganography :

i. Encoding Image into Image : Technique : LSB substitution

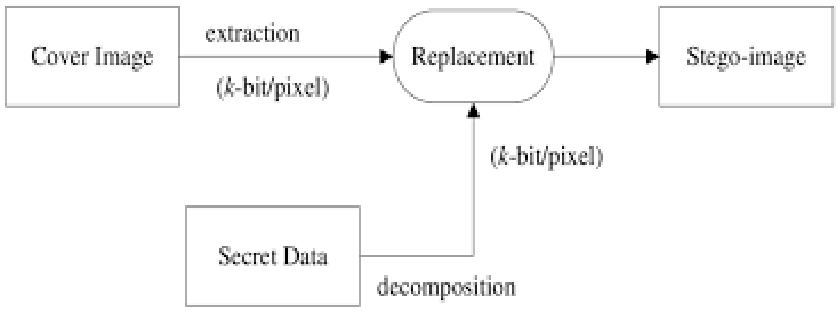
Image steganography is the practice of hiding secret information within an image while maintaining its visual integrity. One common technique for image steganography is LSB (Least Significant Bit) substitution.

Least Significant Bit Substitution :

LSB is the most popular techniques of steganography. To hide a message inside an image without changing its visible properties, the cover source can be altered in ”noisy”.

A simple approach for embedding information in cover image is using Least Significant Bits. The message directly embed into lsb plane of the cover image in a deterministic sequence. Modulating the least significant bit does not result in human perceptible difference because the amplitute of the change is small.

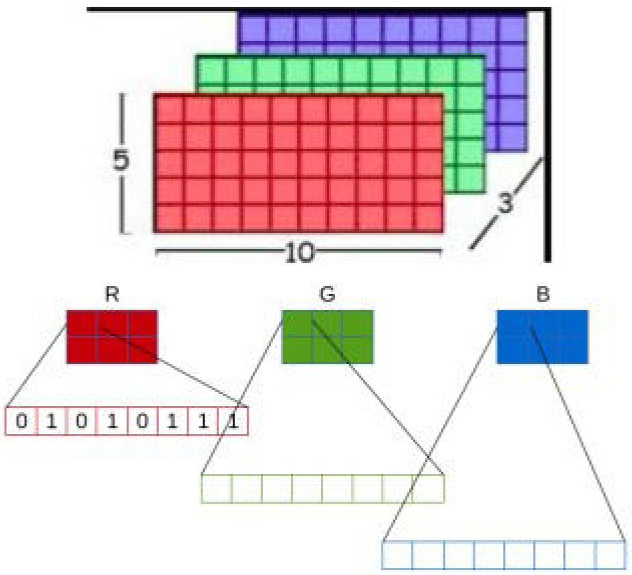
An image is a picture that has been created or copied and stored in electronic form. An image can be described in terms of vector. The numeric represantation form a grid and the individual points are refered to as pixel.



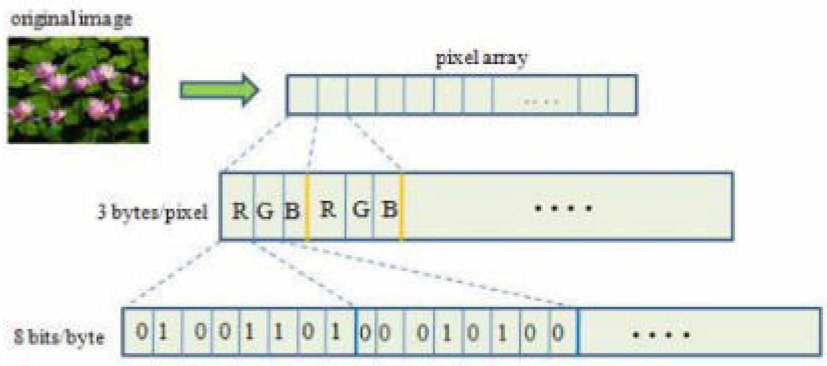
RGB representation of a pixel of an image- In digital image(raster form) 2-dimensional rectangular array of static data elements called pixels. In a pixel the three 8-bits parts red-R, blue-B,and green-G constitute 24-bits.

Approach :

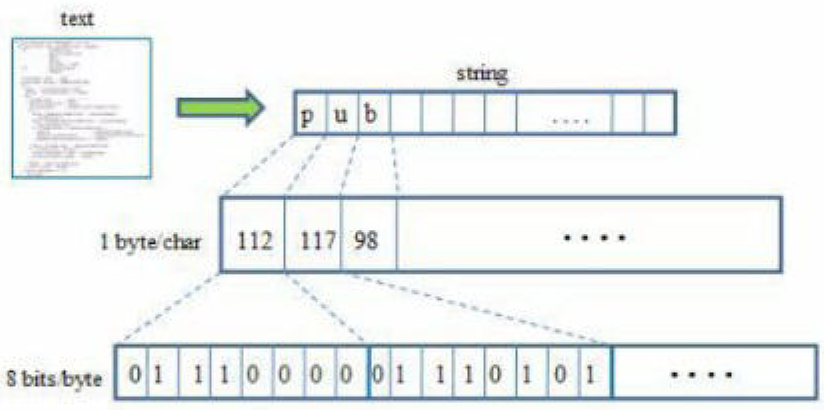
Step 1 : Accessing the bits of an image file



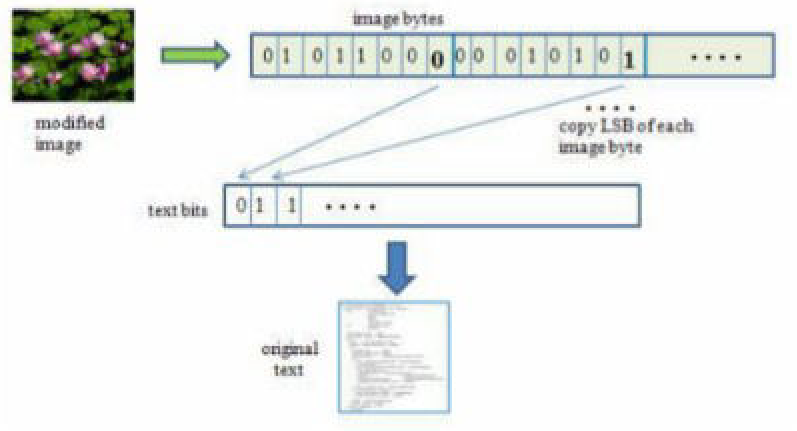
Step2 : Accessing the bits of the text.



Step 3 : Embedding text with image



Step 4 : Extracting text from received image



<u>Encryption-decryption process:</u>

1. Preprocessing:

- Choose the cover image: This is the image that will hide the secret image. It should be visually similar to the secret image to minimize suspicion.
- Convert the cover image and secret image to binary format: Each pixel in an image consists of three color channels (red, green, and blue). Convert the pixel values of both images to binary representation.

2. Embedding:

- Divide the cover image and secret image into corresponding pixel blocks: Each block should contain the same number of pixels (e.g., 8x8 blocks).
- For each pixel block in the cover image:
  - Extract the corresponding pixel block from the secret image.
  - Take the least significant bits of the cover image block and replace them with the most significant bits of the secret image block.
  - Obtain the modified cover image block.
- Combine all the modified cover image blocks to create the stego image.

3. Extraction:

- Divide the stego image into pixel blocks.
- For each pixel block:
  - Take the least significant bits of the block to obtain the embedded bits.
  - Construct the secret image block using the extracted bits.
- Combine all the secret image blocks to reconstruct the hidden image.

4. Postprocessing:

- Convert the reconstructed secret image back to its original format (e.g., from binary to RGB values).
- Save or display the extracted secret image.

ii. Encoding text into Image : Technique : DES + LSB

LSB substitution and DES (Data Encryption Standard) is combined to encode text into an image with an additional layer of encryption and security. First the message is encrypted using DES algorithm and then the cypher text is hidden in the image.

The DES (Data Encryption Standard) algorithm is a symmetric-key block cipher created in the early 1970s by an IBM team and adopted by the National Institute of Standards and Technology (NIST). The algorithm takes the plain text in 64-bit blocks and converts them into ciphertext using 48-bit keys.

Since it's a symmetric-key algorithm, it employs the same key in both encrypting and decrypting the data. If it were an asymmetrical algorithm, it would use different keys for encryption and decryption.

<u>Encryption Process :</u>

1. The `Encrypt` method takes a user password and text to be encryptedas input.
2. The user password is converted to bytes using the UTF-8 encoding.
3. The `DESedeKeySpec` is created using the password bytes.
4. The `SecretKeyFactory` is initialized with the "DESede" encryption scheme, and the key specification is generated.
5. The `Cipher` is initialized in encryption mode using the generated secret key.
6. The user text is converted to bytes using the UTF-8 encoding.
7. The text bytes are encrypted using the initialized cipher.
8. The encrypted bytes are encoded as a Base64 string using the `Base64` class.
9. The encrypted string is returned from `Encrypt` method.
10. The `encodeImage` method is called and takes three parameters: the image byte array, the additional data byte array, and the starting position in the image byte array where the data should be encoded.
11. It first checks if the combined length of the additional data and the offset exceeds the length of the image byte array. If it does, it throws an `IllegalArgumentException` .
12. It then iterates over each byte in the `addition` byte array.

13. For each byte, it extracts the individual bits from most significant to least significant using bitwise operations.

14.It then modifies the corresponding bit in the image byte at the current offset position. The bit is extracted from the image byte using bitwise AND and OR operations.

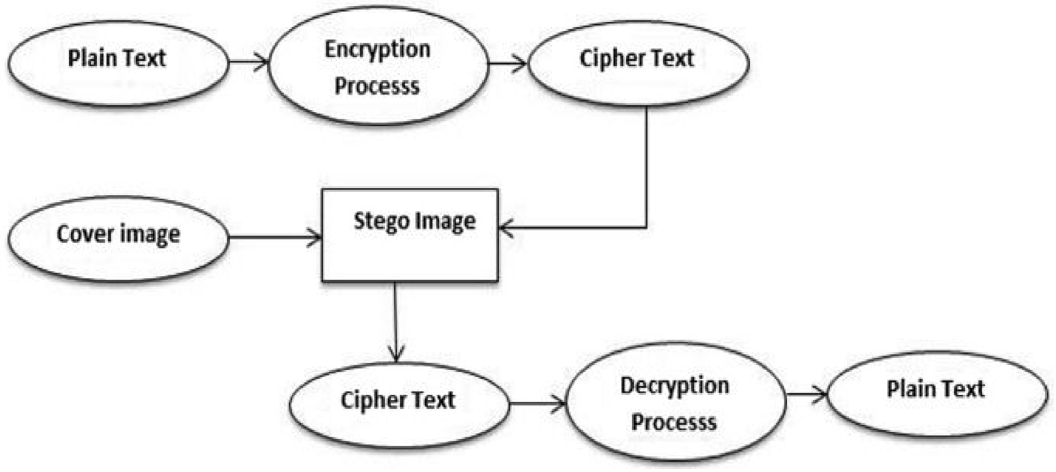
· image[offset] & 0xFE clears the least significant bit of the image byte by ANDing it with a byte mask 0xFE (which has all bits set to 1 except the least significant bit, which is 0).

| b sets the least significant bit of the image byte by ORing it with the extracted bit value b (0 or 1).

15. The offset is incremented after each bit is encoded.

16. Finally, it returns the modified image byte array with the additional data encoded.

For **decryption**, the process is reversed.



2. Audio Steganography : Technique : Randomized LSB substitution.

Randomized Least Significant Bit (LSB) substitution is a modification of the basic LSB steganography algorithm.

In Randomized LSB steganography, instead of sequentially embedding the secret message in the LSBs of consecutive pixels, random data locations within the cover image are selected for storing the message bits. This adds an extra layer of randomness and makes it harder for someone to detect the hidden message by analyzing the LSB patterns.

<u>Encryption-decryption process:</u>

The **encoding** algorithm works as follows:

1. The input audio file is copied to an output file.

2. The length of the secret message is calculated and checked to ensure it is within a valid range.

3. Random access is performed on the output file, and positions within the file are determined where the characters of the secret message will be stored.

4. The length of the message is written to a specific position in the output file.

5. Each character of the message is written to its corresponding position in the output file.

6. The output file is closed, and the encoding process is complete.

The **decoding** algorithm works as follows:

1. The input audio file is opened for random access.

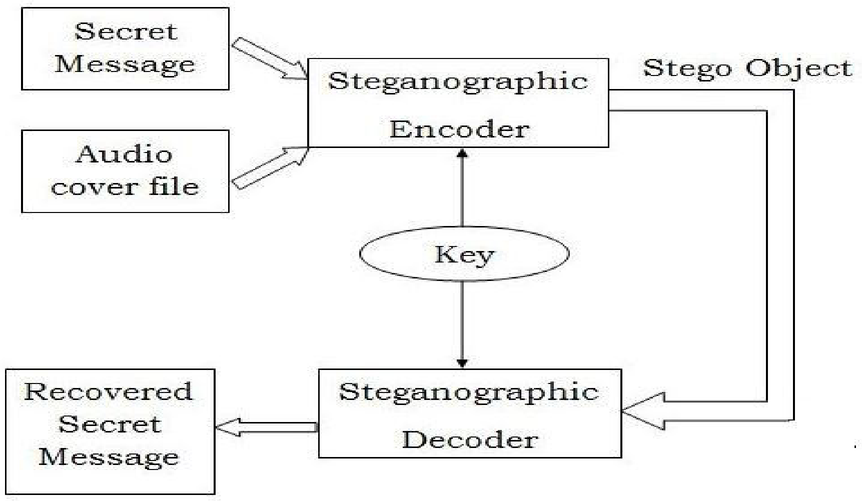
2. The position in the file is set to the start position where the message length is stored.

3. The message length is read from the file.

4. The characters of the message are read from their respective positions in the file.

5. The characters are combined to form the secret message.

6. The secret message is returned.



3. Video Steganography : Technique : RC4 + Frame-level steganography.

RC4 stands for Rivest Cipher 4. Ron Rivest invented RC4 in 1987, and it is a stream cipher. RC4 creates a pseudo-random bit stream (a keystream). These, like any other stream cipher, can be used for encryption by utilizing bit-wise exclusive or to combine it with the plaintext. The same procedure is used for decryption (since exclusive-OR is a symmetric operation).The cipher uses a secret internal state that is divided into two sections to generate the keystream −

- Each of the 256 available bytes is permuted.
- Two index pointers (8 bits each).

Encryption process:

1. Receive a video file and the secret message to be embedded and a secret key.

2. For the secret key entered, the encryption engine creates the keystream.

3. Plaintext is XORed with the generated keystream. Because RC4 is a stream cipher, byte-by-byte XORing is used to generate the encrypted text.

4. This encrypted text is now sent in encrypted form to the calling method.

5. Split the video file into individual frames.

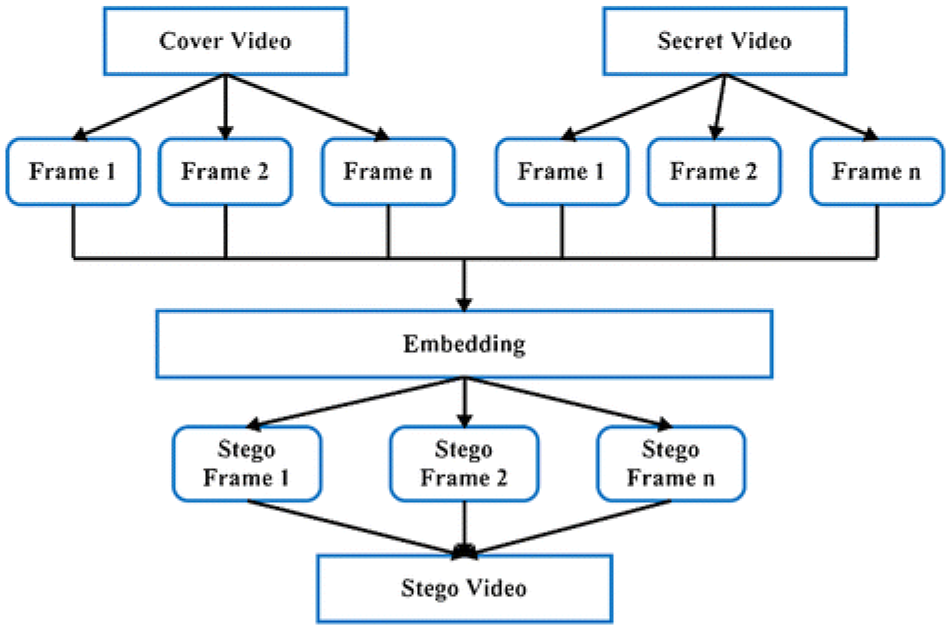
7. Choose specific frames within the video where the encrypted message will be embedded.

8. Modify the Least Significant Bits (LSBs) of pixel values in the frame to represent the bits of the encrypted message. The changes to the LSBs are typically imperceptible to human observers.

6. Combine the modified frames with the unmodified frames to reconstruct the video sequence.

7. Save the steganographic video file, which contains the embedded secret message.

To **extract** the hidden message from the steganographic video, follow a similar process but in reverse.



3.4. HARDWARE & SOFTWARE REQUIREMENT

Hardware Requirements

● For Client

o Processor: Intel Core to Duo or above

o Ram: Min 2 GB

o Hard disk: Min 50 GB

o Network card

● Developer

o Processor: Intel Core i7 8th gen processor or advance

o Ram: 16 GB

o Hard disk: 1TB

o SDD: 250 GB

Software Requirements

● Client:

o Operating System : Java Development Kit (JDK)

o Application's JAR(Java ARchive) file.

● Developer:

o Operating System : Ubuntu 20.04.6 LTS

o Developing Language : Java 8

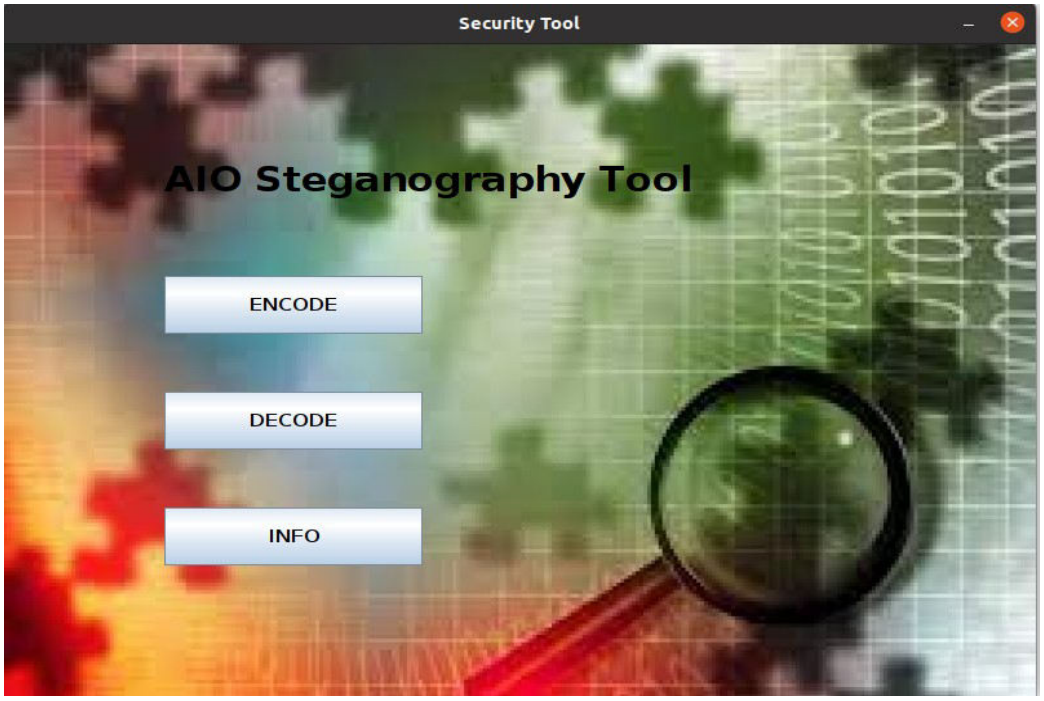
o Development Environment : IntelliJ IDEA

o Framework: Swing

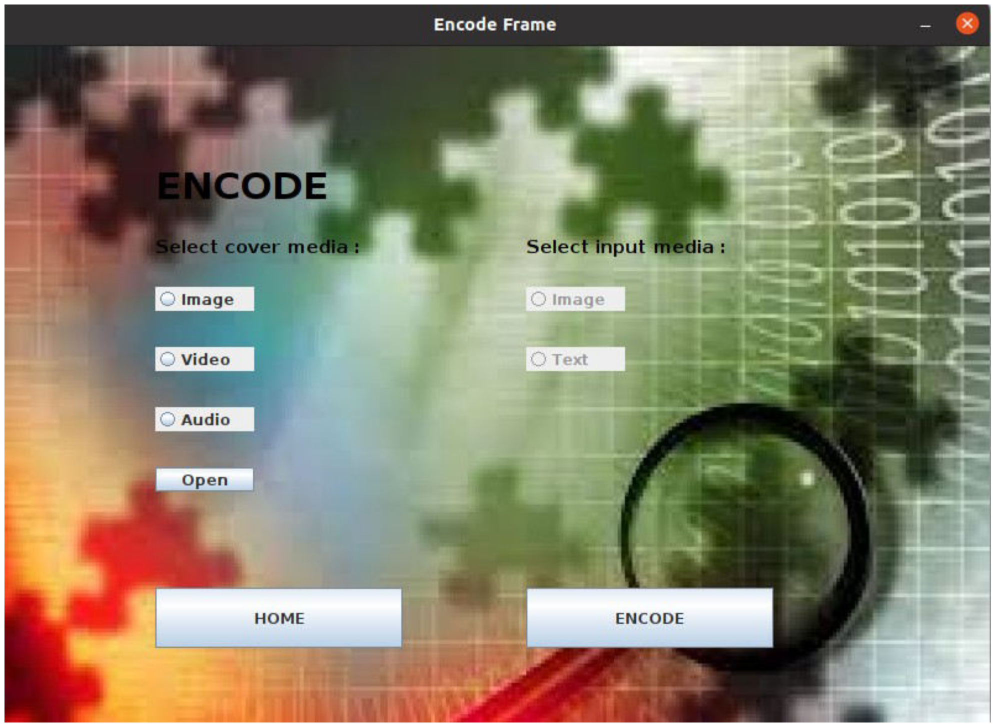
o Version Control: GitHub

UNIT 4: DEMONSTRATION

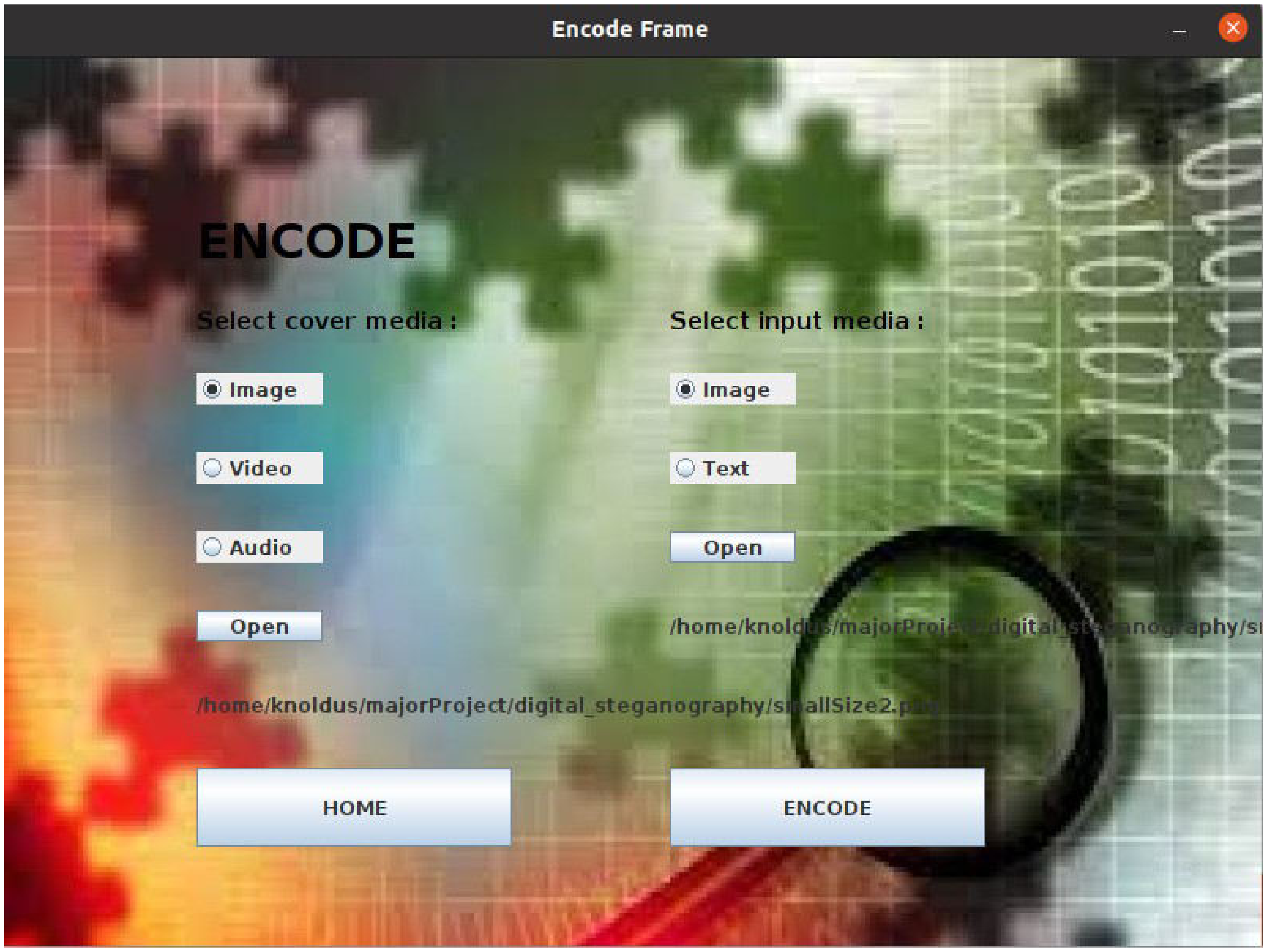
HOME FRAME



FRAME 1 : ENCODE



1 Encoding image into image :



/home/knoldus/majorProject/digital_steganography/s

/home/knoldus/majorProject/digital_steganography/smallSize2.png

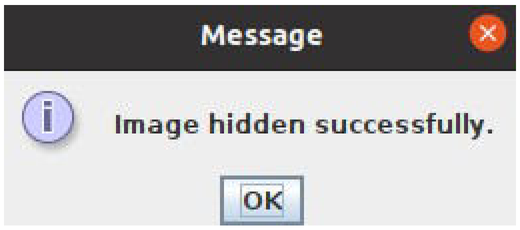
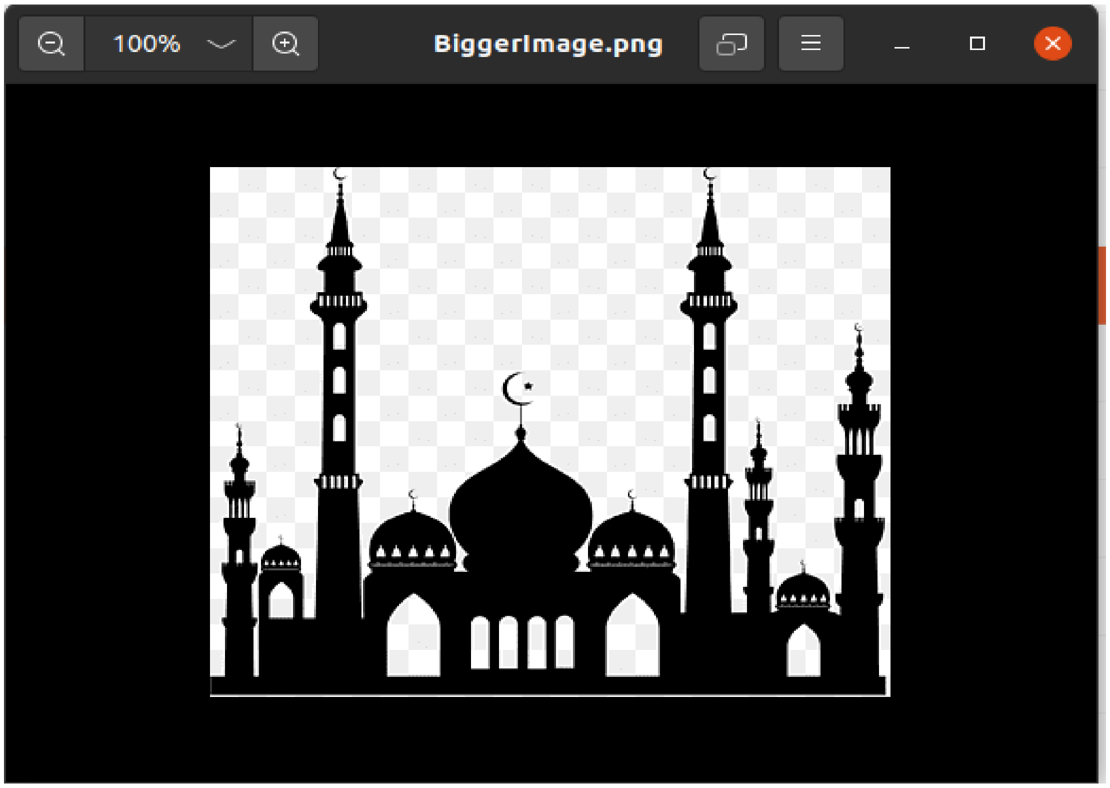


Image hidden successfully.

COVER IMAGE :



SECRET IMAGE :

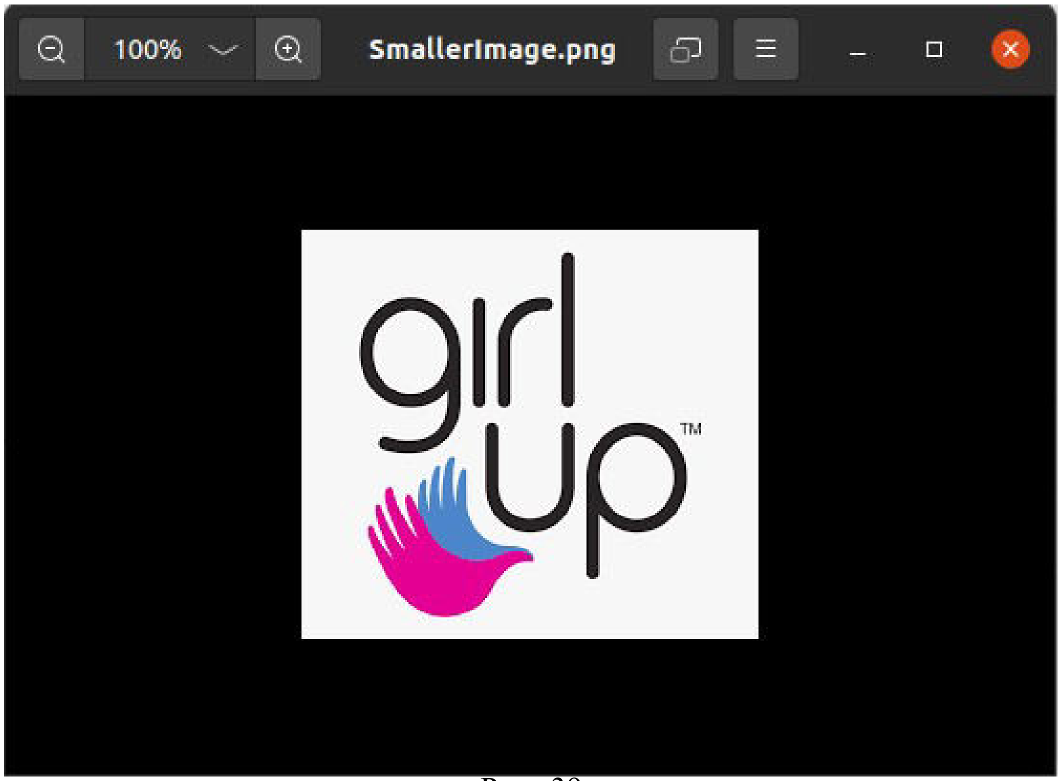
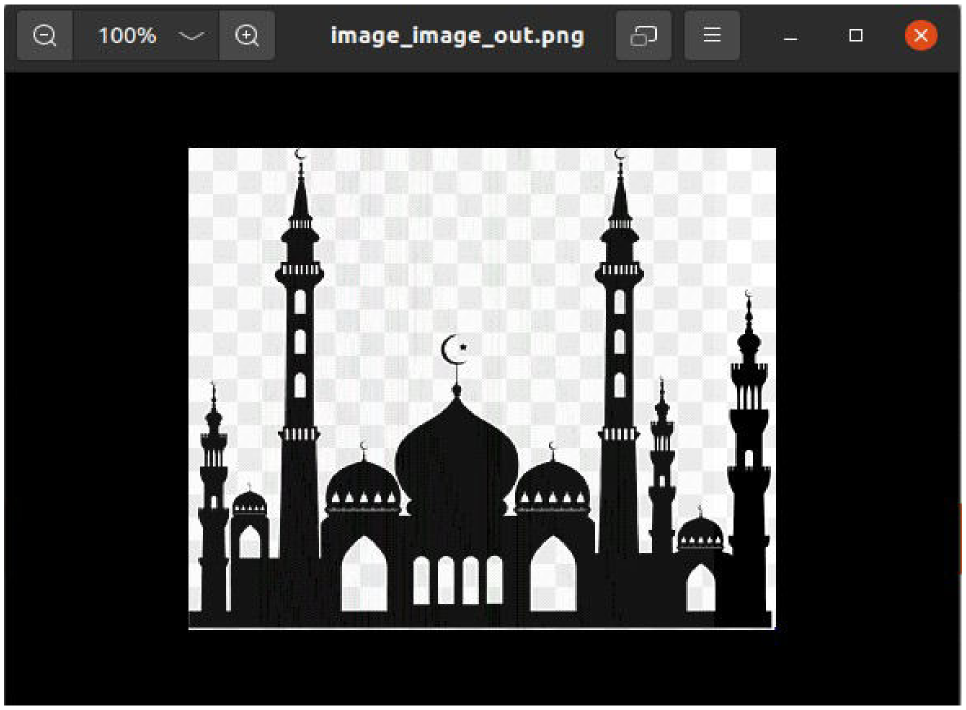
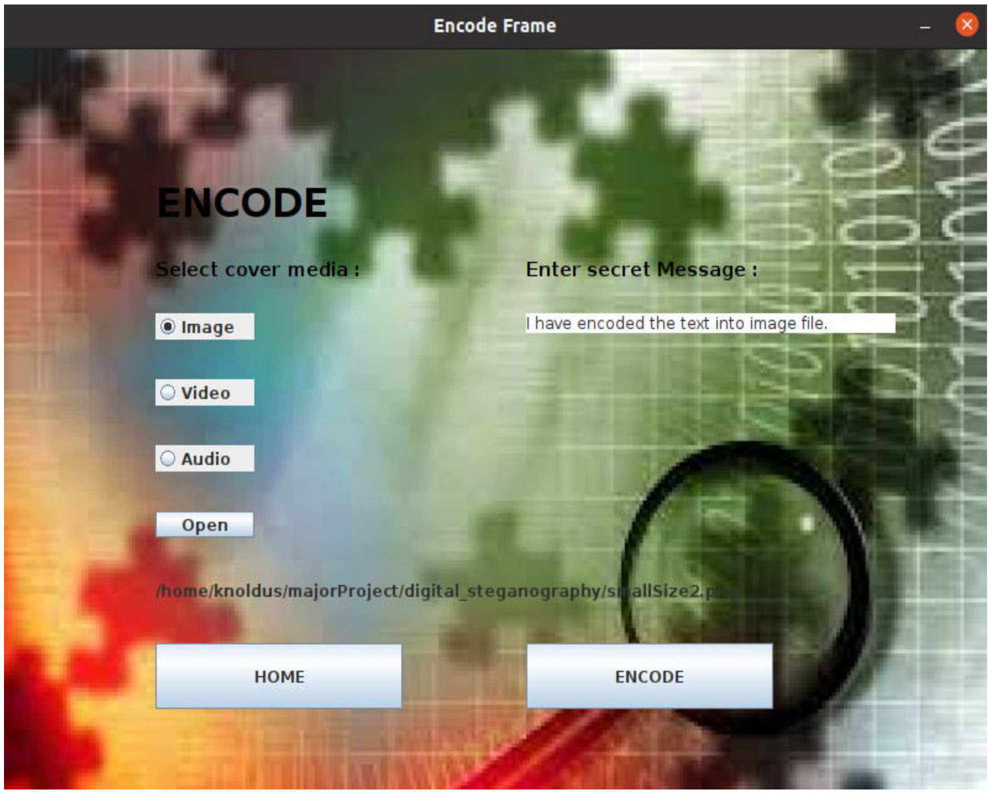


IMAGE AFTER ENCODING :



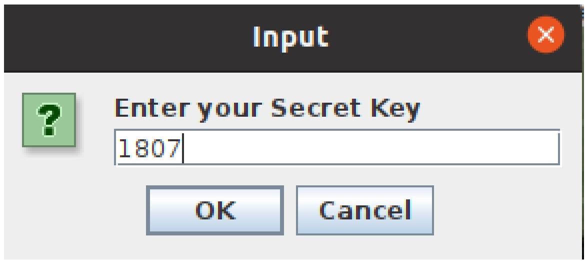
2 Encoding text into image :



Enter secret Message :

I have encoded the text into image file.

/home/knoldus/majorProject/digital_steganography/smallSize2.p


Input

Enter your Secret Key

1807

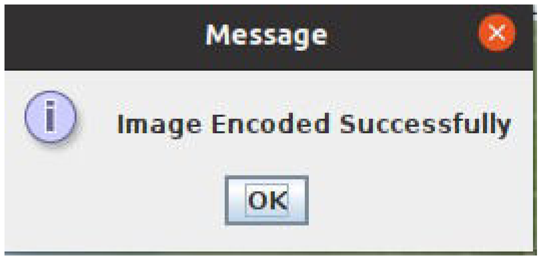
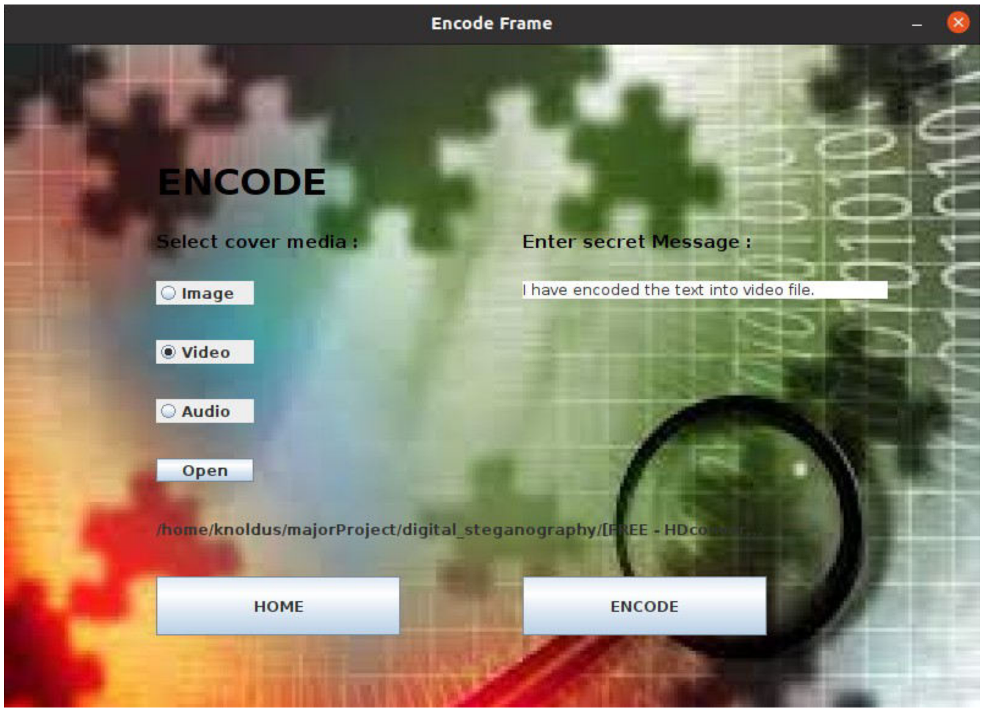

Message

Image Encoded Successfully

3 Encoding text into video :


Encode Frame

ENCODE

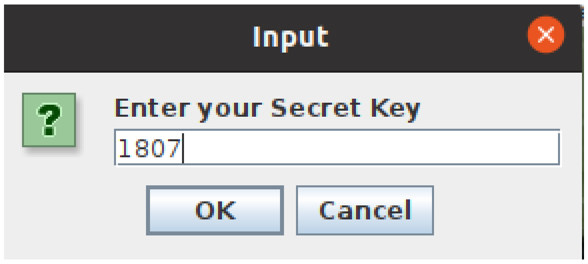
Select cover media :

Enter secret Message :

I have encoded the text into video file.

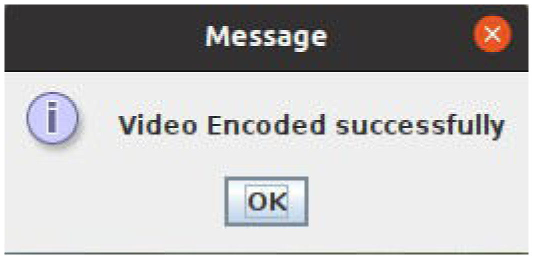
( ) Image
(●) Video
( ) Audio

/home/knoldus/majorProject/digital_steganography/[FREE - HDco


Input

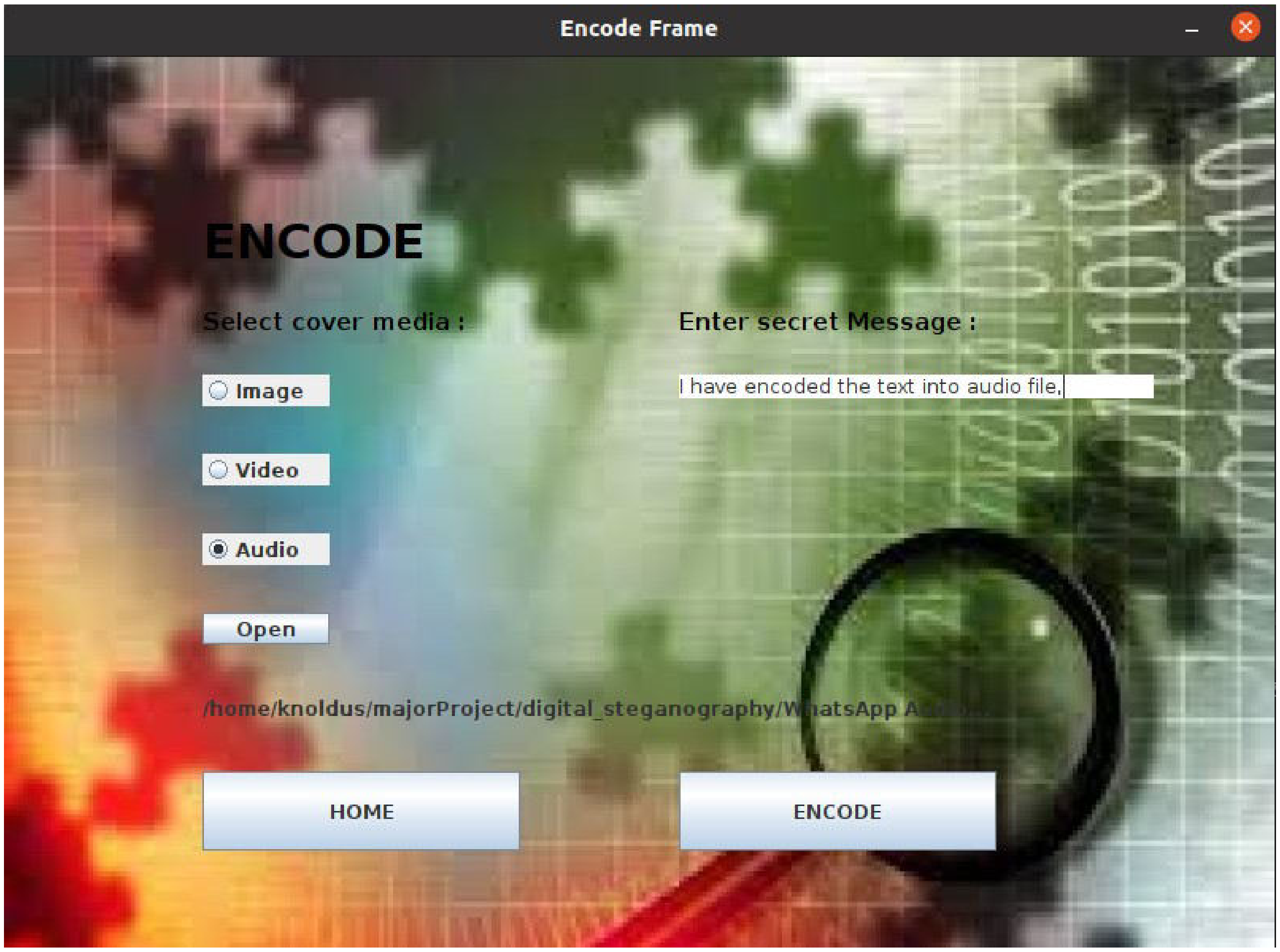
Enter your Secret Key

1807


Message

Video Encoded successfully

4. Encoding text into audio :

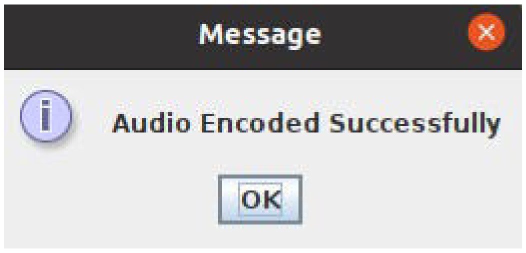


Enter secret Message :

I have encoded the text into audio file,

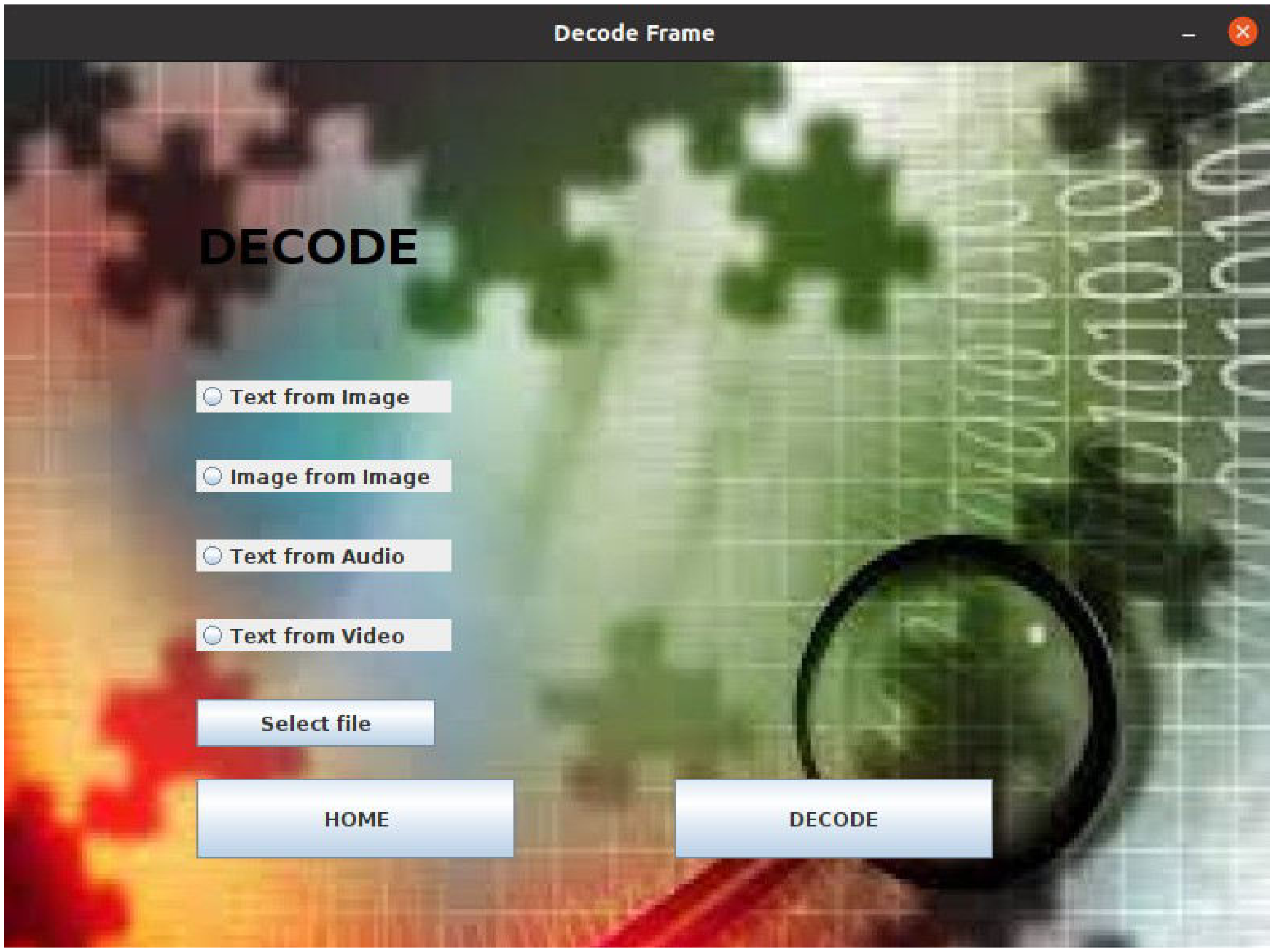
Image [✓ Audio] Video

/home/knoldus/majorProject/digital_steganography/WhatsApp A

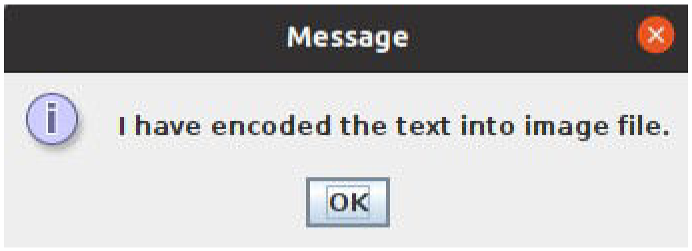

Message

Audio Encoded Successfully

FRAME 2 : DECODE :

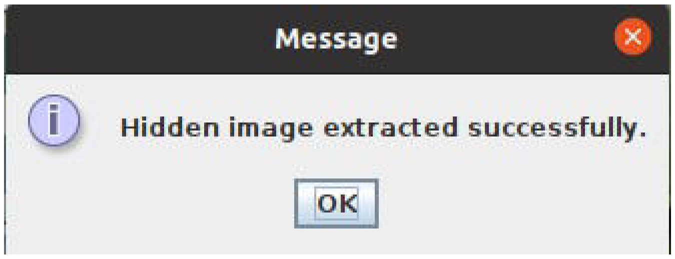


1. Text from Image :

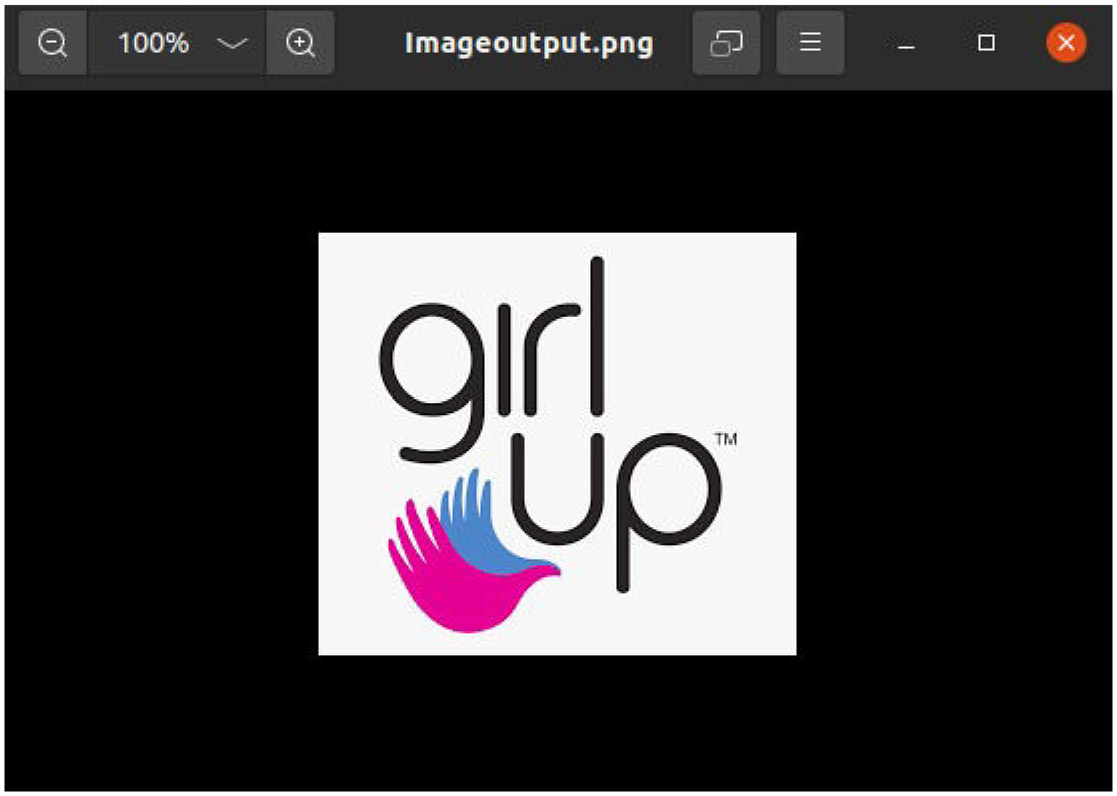


I have encoded the text into image file.

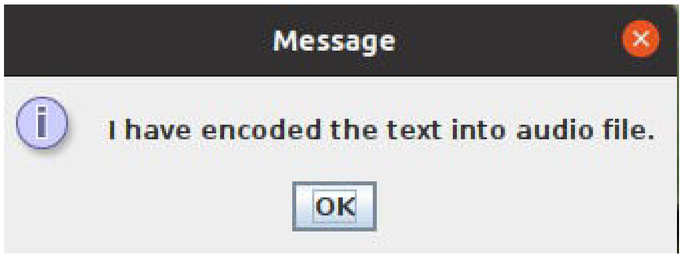
2. Image from Image :



Hidden image extracted successfully.

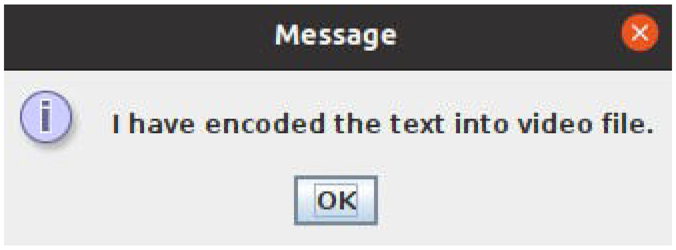


3. Text from Audio :



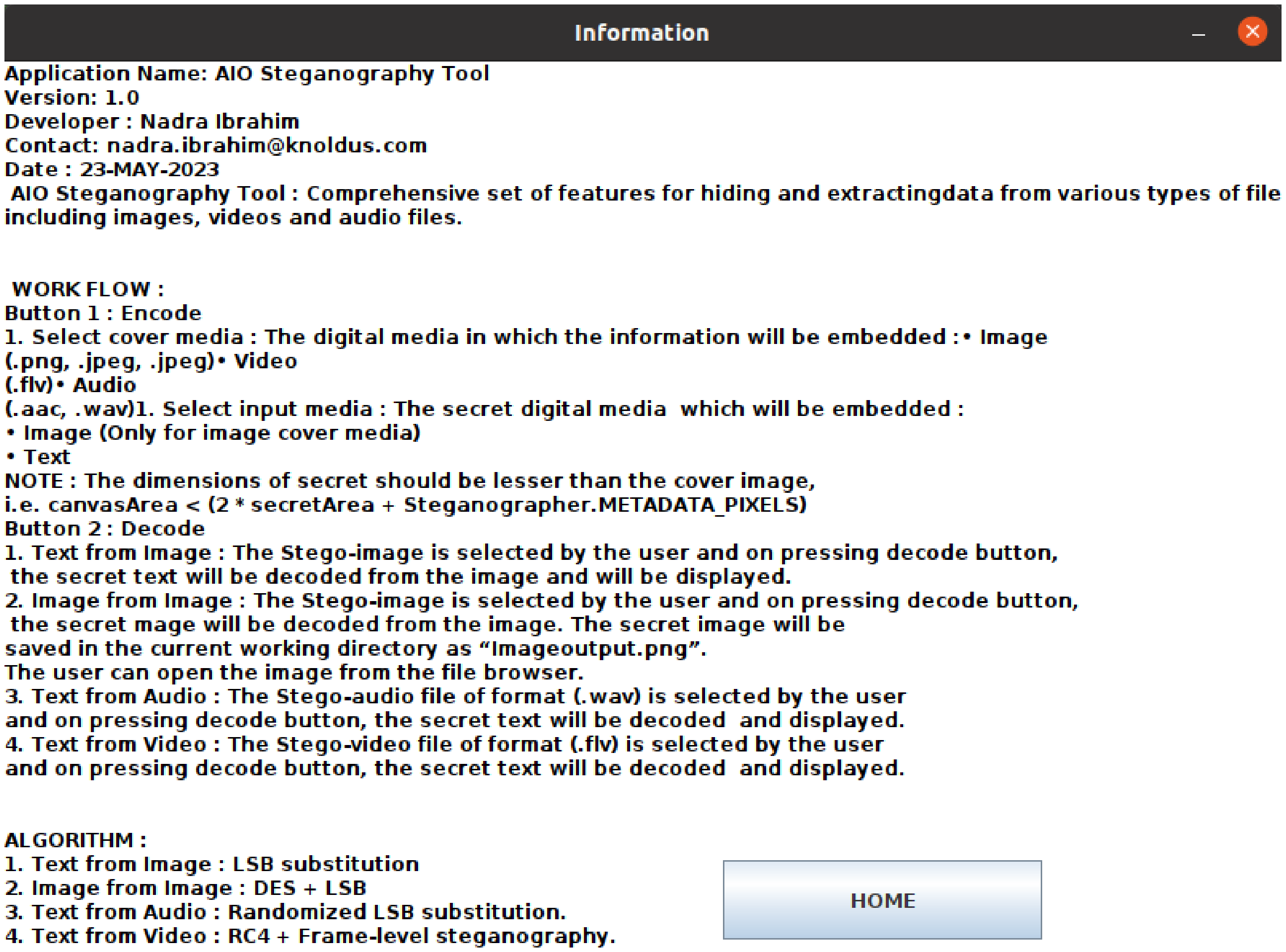
I have encoded the text into audio file.

3. Text from Video :



I have encoded the text into video file.

FRAME 3 : INFO



Application Name: AIO Steganography Tool
Version: 1.0
Developer : Nadra Ibrahim
Contact: nadra.ibrahim@knoldus.com
Date : 23-MAY-2023
AIO Steganography Tool : Comprehensive set of features for hiding and extractingdata from various types of file including images, videos and audio files.

WORK FLOW :
Button 1 : Encode
1. Select cover media : The digital media in which the information will be embedded :• Image (.png, .jpeg, .jpeg)• Video
(.flv)• Audio
(.aac, .wav)1. Select input media : The secret digital media which will be embedded :
• Image (Only for image cover media)
• Text
NOTE : The dimensions of secret should be lesser than the cover image,
i.e. canvasArea < (2 * secretArea + Steganographer.METADATA_PIXELS)
Button 2 : Decode
1. Text from Image : The Stego-image is selected by the user and on pressing decode button,
the secret text will be decoded from the image and will be displayed.
2. Image from Image : The Stego-image is selected by the user and on pressing decode button,
the secret mage will be decoded from the image. The secret image will be
saved in the current working directory as "Imageoutput.png".
The user can open the image from the file browser.
3. Text from Audio : The Stego-audio file of format (.wav) is selected by the user
and on pressing decode button, the secret text will be decoded and displayed.
4. Text from Video : The Stego-video file of format (.flv) is selected by the user
and on pressing decode button, the secret text will be decoded and displayed.

ALGORITHM :
1. Text from Image : LSB substitution
2. Image from Image : DES + LSB
3. Text from Audio : Randomized LSB substitution.
4. Text from Video : RC4 + Frame-level steganography.

UNIT 5:

CONCLUSION &

SCOPE

4.1. CONCLUSION

In conclusion, the All-in-One (AIO) Steganography Tool is a versatile and powerful application for hiding and extracting information within digital media. With its comprehensive set of features and functionalities, it provides users with a convenient and secure means of implementing steganography techniques.
The AIO Steganography Tool offers support for various file formats, including images, audio files, and videos, allowing users to embed and extract hidden information across different types of media. Its user-friendly interface makes it accessible to users with varying levels of technical expertise, enabling both beginners and advanced users to utilize its capabilities effectively.
One of the key strengths of the AIO Steganography Tool is its emphasis on security. The tool employs encryption algorithms and techniques to ensure the confidentiality and integrity of the hidden information. By incorporating cryptographic methods, it provides users with a reliable means of protecting sensitive data from unauthorized access.
Furthermore, the tool includes functionality for detecting and extracting hidden information from digital media. This capability is useful for uncovering concealed data and analyzing steganographic content within files.
Overall, the AIO Steganography Tool is a comprehensive solution for implementing steganography techniques. Its ease of use, support for various file formats, security features, and detection capabilities make it a valuable tool for individuals and organizations seeking to protect sensitive information, engage in covert communication, or apply digital watermarking techniques.

4.2. CONTRIBUTORY SOURCE CODE

Complete souce code is available at :

HTTP Link :https://github.com/ibrahimnadra/aio_steganography_tool.git

SSH Link : git@github.com:ibrahimnadra/aio_steganography_tool.git

or clone through Github CLI :

gh repo clone ibrahimnadra/aio_steganography_tool

4.3. FUTURE SCOPE

The future scope of the AIO Steganography application is promising, with several potential areas of development and enhancement. Here are some possibilities for future advancements:

1. Advanced Steganographic Techniques: As technology evolves, new and more sophisticated steganographic techniques are likely to emerge. The AIO Steganography application can explore incorporating these advanced techniques to provide users with even stronger and more secure ways to hide information within digital media.

2. Deep Learning and AI Integration: Deep learning and artificial intelligence (AI) can significantly contribute to steganalysis and steganography. By leveraging AI algorithms, the AIO Steganography application can improve its detection capabilities, making it more effective in identifying hidden information and detecting advanced steganographic methods.

3. Mobile and Web Integration: Expanding the AIO Steganography application to mobile platforms and web-based versions can increase its accessibility and reach a wider user base. Mobile integration would enable users to perform steganography tasks on their smartphones, while web integration would allow users to access the application directly from their browsers, eliminating the need for software installations.

4. Cross-Media Steganography: Currently, the AIO Steganography application focuses on image, audio, and document steganography. In the future, it can be extended to support other types of media, such as video, 3D models, and virtual reality content. This would provide users with more diverse options for hiding information across various media formats.

5. Improved User Experience: Enhancing the user interface and user experience of the AIO Steganography application can make it more intuitive and user-friendly. Implementing interactive tutorials, tooltips, and visual aids can help users understand and utilize the application's features more effectively.

6. Integration with Cloud Services: Integrating the AIO Steganography application with cloud services can offer users the ability to store and retrieve their steganographic files securely. It can also enable collaboration features, allowing multiple users to work on steganography projects simultaneously.

7. Steganography in Emerging Technologies: As new technologies like augmented reality (AR), blockchain, and Internet of Things (IoT) continue to evolve, exploring steganography applications within these domains can open up new possibilities. For example, embedding hidden information within AR experiences or securing data in IoT devices using steganographic techniques could be areas for future development.

8. Enhanced Security Features: Strengthening the security aspects of the AIO Steganography application, such as implementing advanced encryption algorithms and multi-factor authentication, can further safeguard the hidden information from unauthorized access.

It's important to note that as steganography can have both legitimate and malicious uses, the future scope of the AIO Steganography application should also consider ethical considerations and potential misuse. Implementing responsible usage guidelines and promoting awareness about the responsible use of steganography can be valuable aspects of its future development.

4.4. **REFERENCES**

[1] B. Saha and S. Sharma, “Steganographic Techniques of Data Hiding using Digital Images”, in Defence Science Journal, vol. 62, no. 1, 2012 January, pp. 11-18.

[2] Silman, J., “Steganography and Steganalysis: An Overview”, SANS Institute, 2001.

[3] Jamil, T., “Steganography: The art of hiding information in plain sight”, IEEE Potentials, 18:01, 1999.

[4] Johnson, N.F. & Jajodia, S., “Exploring Steganography: Seeing the Unseen”, Computer Journal, February 1998.

[5] Sei-ichiro Kamata, et al: Depth-First Coding for Multi-Valued Pictures Using Bit-Plane Decomposition, IEEE Trans. on CT, Vol.43, No.5, pp.1961-1969, May, 1995.

[6] https://www.w3.org/TR/PNG-Chunks.html