Aum Amriteshwaryai Namah

Steganography - encode and decode a message in an image

Team Members:
Deepak Yadav AM.EN.U4AIE19024
Gourav Singh Bajeli AM.EN.U4AIE19031
Indraraj Biswas AM.EN.U4AIE19034
Shashank Priyadarshi AM.EN.U4AIE19060

Abstract

Steganography is the study of encoding hidden data in a suitable multimedia carrier, such as an image, audio, or video file. It is based on the concept that if the feature is visible, the point of attack is obvious, hence the goal is always to hide the fact that the embedded data exists. We will cover several principles in Steganography, as well as a brief history of Steganography and a few types of techniques accessible today in Steganography, in this project. Other subjects covered include Steganography security and mobile texting. We intend to create an executable that requires the user to just upload an image and provides them with options for the various tools we have created.

Objectives

The Aim of this project is to implement some of the basic concepts of steganography and introduce the reader to all the concepts of steganography ,advantages and disadvantages.

- We plan to create a EXE in which the user can upload the cover image(image which is used to be insert data) and the secret
- Then with click of one button can receive the encoded image which he/she can use to send secret to some other user
- Then the other user can use the same application and retrieve the secret by giving the encoded image
- We plan to implement the following concepts of steganography:
    - Least significant bit insertion
    - Hiding in Metadata
    - Hiding image in another image
    - DCT
    - LSB insertion in video
- We plan to make this open source on github so that anyone can use this tool

Assumptions

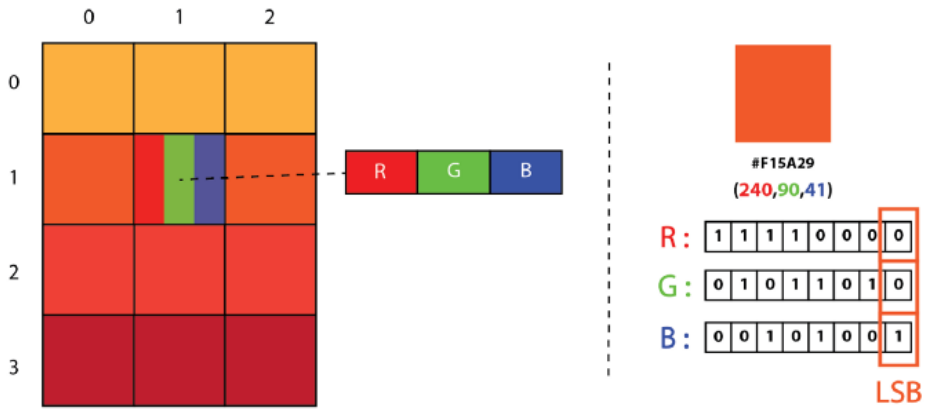
- The DCT steganography is mostly suitable for jpeg/jpg files
- for the hiding image inside another image works when both the images are of almost similar size
- The metadata stego works with any image but can be easily extracted using exiftools so it is not a strong steganography method

- LSB steganography is the most used method as we can hide the data and the image quality also does not decrease by much

System Architecture

We developed a Python based desktop app using PyQt5 library which provides features such as you can select LSB encoding, Hiding in metadata, image hide, etc for encoding and similar decoding techniques. It provides you an option to insert an image and message to start the following encodings. So you need to first insert the image and then the message if your encoding requires based on that you need to press Generate button.

1. **LSB encoding** :The least significant bit is abbreviated as LSB. The rationale behind LSB embedding is that if we change a pixel's final bit value, the colour won't change significantly. 0 is black, for example. Changing the value to 1 won't make much of a change because the colour remains black, but in a lighter tint.



The encoding is done using the following steps:
   - Convert the image to grayscale and resize it if needed
   - Convert the message to be inserted to its binary format
   - Then we traverse through each pixel of the image and do the following:
     - We take the binary of the current pixel and compare its LSB with the current bit of the message
     - if they are same nothing changes
     - else then the LSB changes to current bit of the msg
   - We do this until all the bits of the message are inserted into the pixels
   - Then we save the output pixels into an image

2. **Video-LSB:** We know that video is made up of frames, each of which is a picture. We can save data using LSB steganography and then stitch those encoded frames back together into a video with a secret message if we extract all the frames from a video.We'd start by deciding on the inputs.

- First, the user must choose whether they wish to encode or decode a video. Then they'd provide us a video file with an extension, which we'd read with OpenCV.
- Now that we have the video, the first thing we should extract frames from the movie as photos.
- We can break the strings into little chunks and hide each chunk of the message inside a frame now that we have all of the extracted frames.
- We'll now concentrate on extracting the audio from the video, which will then be stitched together with frames. We can use FFmpeg to stitch all of our frames with a concealed message together to make a film, and then lay out the audio.

Note: we have not included this in the app but we have implemented a simple code of it.

3. **Hiding in Metadata** :The secret message is hidden in the image metadata.The Exif format stores image metadata (date, time, format, camera tags, photo editing software tags, and so on) and can be accessed by any photo viewer software.
   Here we will be using piexif module
   The list of possible tags is quite large (see EXIF Tags) and piexif doesn't manage all of them (piexif supported tags) but enough for you to be creative and hide data in unusual places like GPS coordinates or even image thumbnails…
   But just keep in mind that Exif metadata can be restricted in size (64 kB in JPEG).

4. **Hiding image in another image** : In this method we will be hiding one image inside another image.To hide a picture within another, the image to be hidden must be at least the same size as the image to be hidden.
   - We must create two loops to go through all rows and columns (actually each pixel) from the images.
   - we get the RGB from image 1 and image 2 as binary values.
   - We merge the most significant bits from image 1 with the most significant bits from image 2.
   - Note that the **merge_rgb** function is using the 4 most significant bits from each image, but it could be changed. Keep in mind that using fewer bits from the hidden image will result in low quality of the recovery image.

- And finally we convert the new binary val to int And set it to a new pixel position from the resulted image.Now we have an image hidden inside another image.

5. **DCT steganography** :The DCT is a mathematical transformation that takes a signal and transforms it from spatial domain into frequency domain.
    - DCT coefficient are used for jpeg compression.It separates the image into parts for differing importance .it transforms a signal or image from the spatial domain to the frequency domain.it can separate the image into high, middle and low frequency components the general equation for a 1d DCT is :

$$C(u) = \alpha(u)\sum_{x=0}^{N-1} f(x)\cos\left[\frac{(2x+1)u\pi}{2N}\right]$$

for $u = 0, 1, 2, \ldots, N-1$.

    - The general equation for a 2D DCT is defined by:

$$C(u,v) = \alpha(u)\alpha(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cos\left[\frac{(2x+1)u\pi}{2N}\right]\cos\left[\frac{(2y+1)v\pi}{2N}\right] \quad (2)$$

for $u,v = 0, 1, 2, \ldots, N-1$

Contribution of each member of the team

Deepak Yadav(AM.EN.U4AIE19024)- I researched about the different methods of steganography and implemented the meta data steganography where the secret is hidden in the metadata of the image and also helped in building the app

Gourav Singh Bajeli (AM.EN.U4AIE19031)- I implemented the LSB Encoding of the image as well as video LSB and also helped shashank in implementing the DCT encoding of the image, i researched many pages and came up with the methods of steganography we have implemented.i have added the link to those papers in the reference column at the end of this paper

Indraraj Biswas (AM.EN.U4AIE19034)- I researched for the datasets and implemented the Hiding image in another image method and was also researching on the various methods of steganography

Shashank Priyadarshi(AM.EN.U4AIE19060)- I researched and learnt the DCT concept and implemented it. I was also looking into DWT and was not able to implement it properly as it was causing some errors.I also built the app that uses all the methods we have used above and performs steganography.

Input to the system

We have included images with which we have tested these methods along with code

- **LSB:** for LSB encoding we have tried with jpg and png,it doesn't work for some of the jpg files mainly because most pixels of those images must be having some set of pixels which are not fit for this method tried with PNGs and JPEGs.
- **DCT:** for DCT all the images we tried worked, it might cause error in some conditions ,that we are not aware of.tried with PNGs and JPEGs.
- **Image_hide:** in this the image2 should be greater than image 1 in size ,also it was causing an error for a image as during decoding it was unable to retrieve the MSB but it worked on some other examples we tried.tried with PNGs and JPEGs.
- **Metadata:** it works with every image we tried which were PNGs and JPEGs.
- **LSB Video:** this we have tested with only one video due to some time constraints ,but we expect it to work with any MP4 for now.Also, we prefer you use a short video for this

Examples:
DCT_encode:
msg= “hello_world”

Metadata_encode:
msg=”yo_world”

Image_hide:

LSB_Encode:
msg=”very_secret_message”

LSB_Video:
msg=”hello_world”
https://drive.google.com/file/d/1SNn7LVHYYVICdwDFM2-y7HupB2Z_bL7m/view?usp=sharing

(for decode functions we would be using the output of these photos)

Output

- **LSB:** the output generated from the encoded function would be the image with the secret in it and in decryption you can expect the secret message.there wont be much noticeable difference in the output and input image
- **DCT:** The output image would have a light blue tint layer on it .the output of decode would be the secret message.the only difference we noticed was the blue tint on the output of encoded image
- **Image_hide:** The output would be the 1st image but it can loose some quality as we would be replacing the 4 LSBs.for the decoding expect the img2 but the quality might be low as we set the last 4LSB as 0000

- **Metadata:** The output would be the image with the secret message inside the metadata of the image and the decrypt function extracts the metadata and get the secret and prints it.you can also use exiftool to extract the secret.

Examples:

DCT:

Decrypt function output:

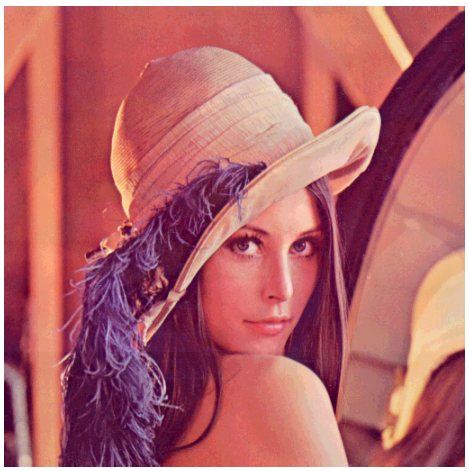
```
fugitiv3@Fugitive:/mnt/g/SEM5/Project/SIP$ python Project.py
babylon.png
hello_world
```

LSB:

Decrypt function output:

```
fugitiv3@Fugitive:/mnt/g/SEM5/Project/SIP$ python Project.py
Secret: very_secret_message
```

Image hide:



Decrypt function output:

Metadata:

Decrypt function output:

```
fugitiv3@Fugitive:/mnt/g/SEM5/Project/SIP$ python Project.py
yo_world
```

LSB_VIDEO:
https://drive.google.com/file/d/1n64LWtyY-yhn5-By1sxL1J_aM69LwlxJ/view?usp=sharing

```
fugitiv3@Fugitive:/mnt/g/SEM5/Project/SIP$ python video_project.py
1.Hide a message in video
2.Reveal the secret from the video

Any other value to exit

Enter your choice :1
Enter the name of video file with extension:input_video.mp4
Enter the message :hello_world
[INFO] temp directory is created
[INFO] frame ./temp/0.png holds he
[INFO] frame ./temp/1.png holds ll
[INFO] frame ./temp/2.png holds o_
[INFO] frame ./temp/3.png holds wo
[INFO] frame ./temp/4.png holds rl
[INFO] frame ./temp/5.png holds d
The message is stored in the Embedded_Video.mp4 file
[INFO] temp files are cleaned up
1.Hide a message in video
2.Reveal the secret from the video

Any other value to exit

Enter your choice :3
```

Decrypt function output:

```
fugitiv3@Fugitive:/mnt/g/SEM5/Project/SIP$ python video_project.py
1.Hide a message in video
2.Reveal the secret from the video

Any other value to exit

Enter your choice :2
Enter the name of video with extension :Embedded_Video.mp4
[INFO] temp directory is created
hello_world
[INFO] temp files are cleaned up
1.Hide a message in video
2.Reveal the secret from the video

Any other value to exit

Enter your choice :3
fugitiv3@Fugitive:/mnt/g/SEM5/Project/SIP$
```

References:

- https://en.wikipedia.org/wiki/Discrete_cosine_transform
- https://link.springer.com/chapter/10.1007/978-3-642-15766-0_102
- https://www.ijert.org/multiple-image-steganography-using-lsb-dct-technique
- https://www.hindawi.com/journals/scn/2021/4179340/
- https://www.ukessays.com/essays/computer-science/the-types-and-techniques-of-steganography-computer-science-essay.php
- https://core.ac.uk/download/pdf/234644722.pdf