IMAGE STEGANOGRAPHY

Minal H R
Dept of Computer Science
PES University
Bangalore, India
minalhr55@gmail.com

Rishika Satheesh
Dept of Computer Science
PES University
Bangalore, India
rishikaprs@gmail.com

I. Abstract—Data hiding is one of the best topics in secret communication. A lossless data hiding technique is done using LSB . Steganography is the art and science of hiding the fact that communication is taking place. Secrets can be hidden in all types of medium: text, audio, video and images. Steganography is an important area of research in recent years involving a number of applications. It is the science of embedding information into the cover image viz., text, video, and image without causing statistically significant modification to the cover image. The modern secure image steganography presents a challenging task of transferring the embedded information to the destination without being detected.

II. Introduction

In transmitting vast volumes of data over open networks and dangerous platforms, the Internet and cloud systems are commonly used, exposing private and confidential data to severe situations. It has thus become one of the most critical issues in the field of data protection to ensure that data transmission over these media is safe and secure. A variety of methods have been developed to keep unauthorized people away from the data transmitted, and steganography is one of them.
Steganography is a type of security technique that hides the presence of a message between the sender and the intended recipient through obscurity, science and art.
Imperceptibility, payload and robustness are the three most significant criteria for audio steganography. Different implementations of the steganography technique used have different specifications. It attempts to conceal hidden messages in a regular message as a strategy for clandestine communication. On a number of data types such as text, image, video and audio data, steganography techniques can be applied.

III. PROBLEM STATEMENT

A. Goal

A model that is used to hide and retrieve data from an image.

B. Methodology

We have used the LSB algorithm on a kaggle dataset of images from Star Wars to work on image steganography . Data from 3 different text files in star-wars-movie-scripts folder is read, cleaned and stored in string, All_SW_Scripts.Image is read from star-wars-steganography-images. The rows, columns and color channels are read. The size of message in bytes is compared to the total number of pixels in the image using

CheckbitSize() function:

if size of message < size of image:

return True

else:

return False

if CheckBitSize() returns True:

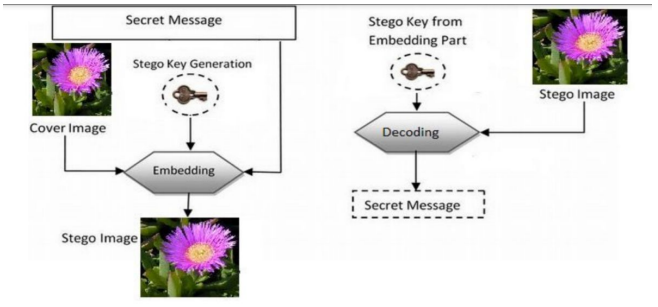
encode message, i.e string All_SW_Scripts into image using LSB Algorithm Compare the original image with the cover image. Decode the message from the cover image.

if decoded message == string, All_SW_Scripts:

successful encryption and decryption.

Cover image is saved. We connect it to a login portal after the job is completed, with a particular password that gives us access to this project. Later on a simple interface could be added.

C. Data Flow Diagram .



.

IV. APPROACH

We use the star wars movie script as covertext (the one which we embed into an image). We embed it into the images from the star wars dataset that we found on kaggle we are using the algorithm of least significant bit as mentioned above and with the help of PIL python library we can extract the pixels from the images and the text and encode the data.similarly we decode it. the receiver receives the encoded covertext and hence he will be able to decode it using the login security portal that will be added into the system. key can be added too.

C. Least Significant Bit

Why LSB?

LSB data hiding technique does not affect the visible properties of the image. This project deals with hiding text in an image file using Least Significant Bit (LSB) technique. The LSB algorithm is implemented in a spatial domain in which the payload bits are embedded into the least significant bits of cover image to derive the stego-image.

D. Algorithm

All images can be represented in the form of pixels(picture elements). Each byte consists of 8 bits. The last bit in a byte is called the Least Significant Bit as its value will affect the pixel value only by “1”(or max 3) . So, this property is used to hide the data in the image. This helps in storing extra data.

The Least Significant Bit (LSB) steganography is one such technique in which the least significant bit of the image is replaced with a data bit.

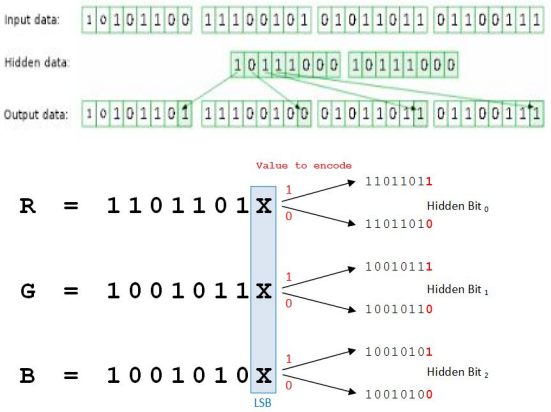
In this method the least significant bits of some or all of the bytes inside an image is replaced with bits of the secret message.

Each pixel consists of red(R), green(G) and blue(B) values, where each takes a value between 0 and 255.

Binary representation of 255= 11111111.

In message to be encoded, each character’s ASCII value is converted to its binary form .

About 3 consecutive pixels(9 bytes) are required to decode 1 letter, so that the least significant bits may be replaced by the bits of the character.



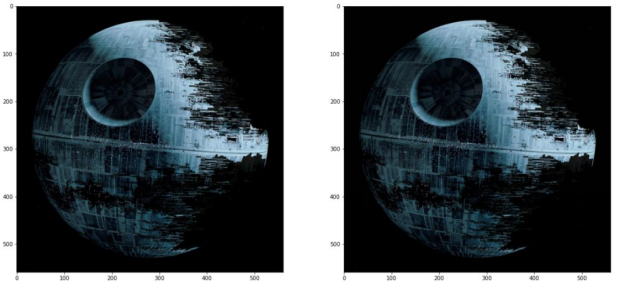
V. IMPLEMENTATION

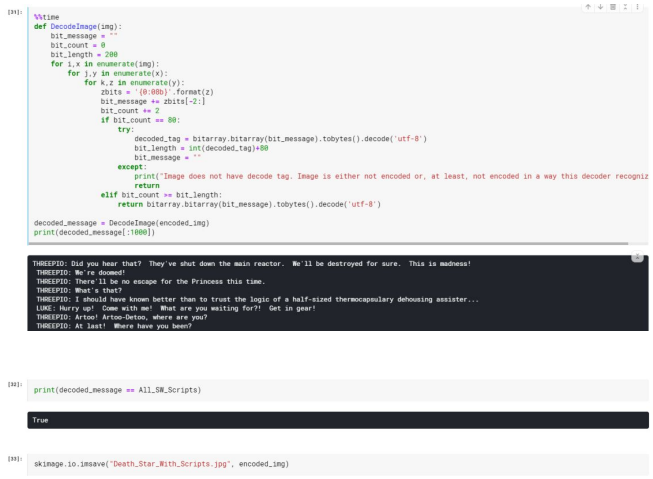
We have implemented our project notebook here, the following are the screenshots of the output:
Dataset used: Starwars images and movie scripts (link is mentioned in the reference)

```
[24]:
All_SW_Scripts = ""

def TextToString(txt):
    with open (txt, 'r') as file:
        data=file.readlines()
        script = ""
        for x in data[1:]:
            x = x.replace('"','').replace("\n"," \n ").split(' ')
            x[1] += ":"
            script += " ".join(x[1:-1]).replace("\n"," \n ")
        return script
All_SW_Scripts += TextToString("../input/star-wars-movie-scripts/SW_EpisodeIV.txt")
All_SW_Scripts += TextToString("../input/star-wars-movie-scripts/SW_EpisodeV.txt")
All_SW_Scripts += TextToString("../input/star-wars-movie-scripts/SW_EpisodeVI.txt")
print(All_SW_Scripts[:1000])
```

```
THREEPIO: Did you hear that?  They've shut down the main reactor.  We'll be destroyed for sure.  This is madness!
 THREEPIO: We're doomed!
 THREEPIO: There'll be no escape for the Princess this time.
 THREEPIO: What's that?
 THREEPIO: I should have known better than to trust the logic of a half-sized thermocapsulary dehousing assister...
 LUKE: Hurry up!  Come with me!  What are you waiting for?!  Get in gear!
 THREEPIO: Artoo! Artoo-Detoo, where are you?
 THREEPIO: At last!  Where have you been?
 THREEPIO: They're heading in this direction. What are we going to do?  We'll be sent to the spice mines of Kessel or smashed into who
knows what!
 THREEPIO: Wait a minute, where are you going?
```

```
[29]:
def CompareTwoImages(img1,img2):
    fig=plt.figure(figsize=[20, 20])

    fig.add_subplot(2, 2, 1)
    plt.imshow(img1)
    fig.add_subplot(2, 2, 2)
    plt.imshow(img2)

    plt.show()
CompareTwoImages(deathstar_img, encoded_img)
```





```
%%time
def DecodeImage(img):
    bit_message = ""
    bit_count = 0
    bit_length = 200
    for i,x in enumerate(img):
        for j,y in enumerate(x):
            for k,z in enumerate(y):
                zbits = '{0:08b}'.format(z)
                bit_message += zbits[-2:]
                bit_count += 2
                if bit_count == 80:
                    try:
                        decoded_tag = bitarray.bitarray(bit_message).tobytes().decode('utf-8')
                        bit_length = int(decoded_tag)+80
                        bit_message = ""
                    except:
                        print("Image does not have decode tag. Image is either not encoded or, at least, not encoded in a way this decoder recogniz
                        return
                elif bit_count >= bit_length:
                    return bitarray.bitarray(bit_message).tobytes().decode('utf-8')

decoded_message = DecodeImage(encoded_img)
print(decoded_message[:1000])
```

```
THREEPIO: Did you hear that?  They've shut down the main reactor.  We'll be destroyed for sure.  This is madness!
 THREEPIO: We're doomed!
 THREEPIO: There'll be no escape for the Princess this time.
 THREEPIO: What's that?
 THREEPIO: I should have known better than to trust the logic of a half-sized thermocapsulary dehousing assister...
 LUKE: Hurry up!  Come with me!  What are you waiting for?!  Get in gear!
 THREEPIO: Artoo! Artoo-Detoo, where are you?
 THREEPIO: At last!  Where have you been?
```



```
print(decoded_message == All_SW_Scripts)
```

```
True
```



```
skimage.io.imsave("Death_Star_With_Scripts.jpg", encoded_img)
```

VI. References

[1] Dataset

The dataset we have used here is the movie script of star wars. We found it on kaggle.The link is given below

https://www.kaggle.com/xvivancos/star-wars-movie-scripts

[2] We used this dataset to get the images for encoding and decoding:

https://www.kaggle.com/valkling/star-wars-steganography-images

VII. Expected Result

This system will be able to encode and decode the secret movie script message that is embedded into the star wars images with the help of a useful algorithm called LSB. Image steganography is majorly used in the field of cybersecurity.

Below is the link for our project notebook :

https://www.kaggle.com/rishikasatheesh/image-steganography

VIII. CONCLUSION

In this paper, we have given a brief explanation of our project based on image steganography, that is widely used for network security purposes . The project should be able to fulfill the expected end result.

IX. ACKNOWLEDGEMENT

We are sincerely thankful for our course instructors who gave us a correct picture on how to take this project forward. We are thankful to each other for displaying teamwork and cooperating in the time of the making of this project.