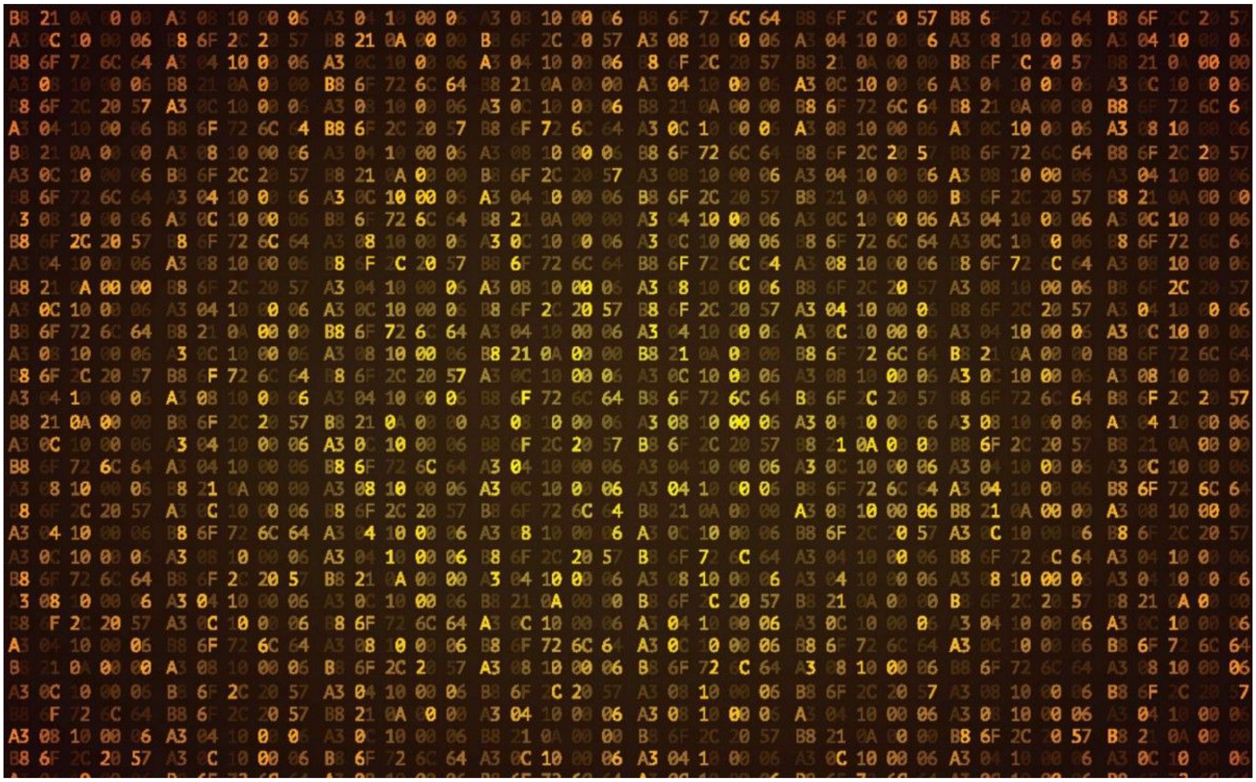


Image Steganography Project First Review

Cryptography and Network Security (MC - 407)

10th October 2020

Anish Sachdeva
Delhi Technological University (DTU)
DTU/2K16/MC/013

Least Significant Bit (LSB) Steganography

Steganography is the art of hiding messages so that they can only be read or interpreted by the receiving party for whom it was meant for. It is different from Cryptography, as Cryptography relies on encrypting (changing) the data using an encryption function such that only the intended receiving party can decrypt it and see the plaintext message.

Steganography on the other hand relies on the principle of security by obscurity, hiding the message in plain sight. There are many different methods of hiding data inside Images, one of the simplest and very efficient ways to do that is by using the LSB (Least Significant Bit) Steganography technique.

In this technique we store the data bits in each of the least significant bits of the Red, Blue and green channel. In this project I implement the LSB technique and further enhance it by adding an additional Blue channel layer for pixel storage.

I have implemented the LSB Steganography technique and it can be viewed on GitHub here.

Tasks Accomplished

- Implemented a Python Function that takes in an image and a plaintext message and encrypts that image with the message and returns a new image.
- Implemented a Python Function that Takes in an image and decrypts it to retrieve the hidden message.
- Implemented a Python function that can save an image of any format with a PNG extension so that the data is stored in a lossless format and can be retrieved again with accuracy. Other formats such as JPG, JPEG, JPEG-2000 are lossy formats and messages hidden in them can’t be retrieved again.
- Implemented a web server which provides a visual interface to encrypt images with the plaintext message and decrypt them again to retrieve the message.

Project in Action

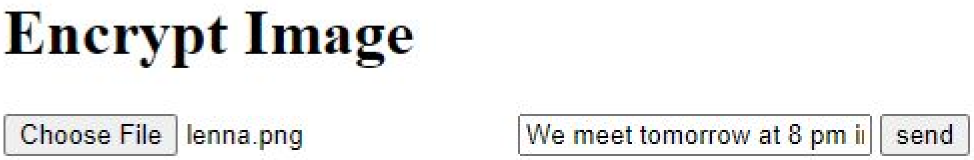
Taking an image and a message. Let the Image be standard Lenna Image

Lenna: Standard Image Used in Digital Image Processing

We now consider the following message:

Message: We meet tomorrow at 8 pm in the park on the bridge to exchange the package.

We enter this information into our web application.



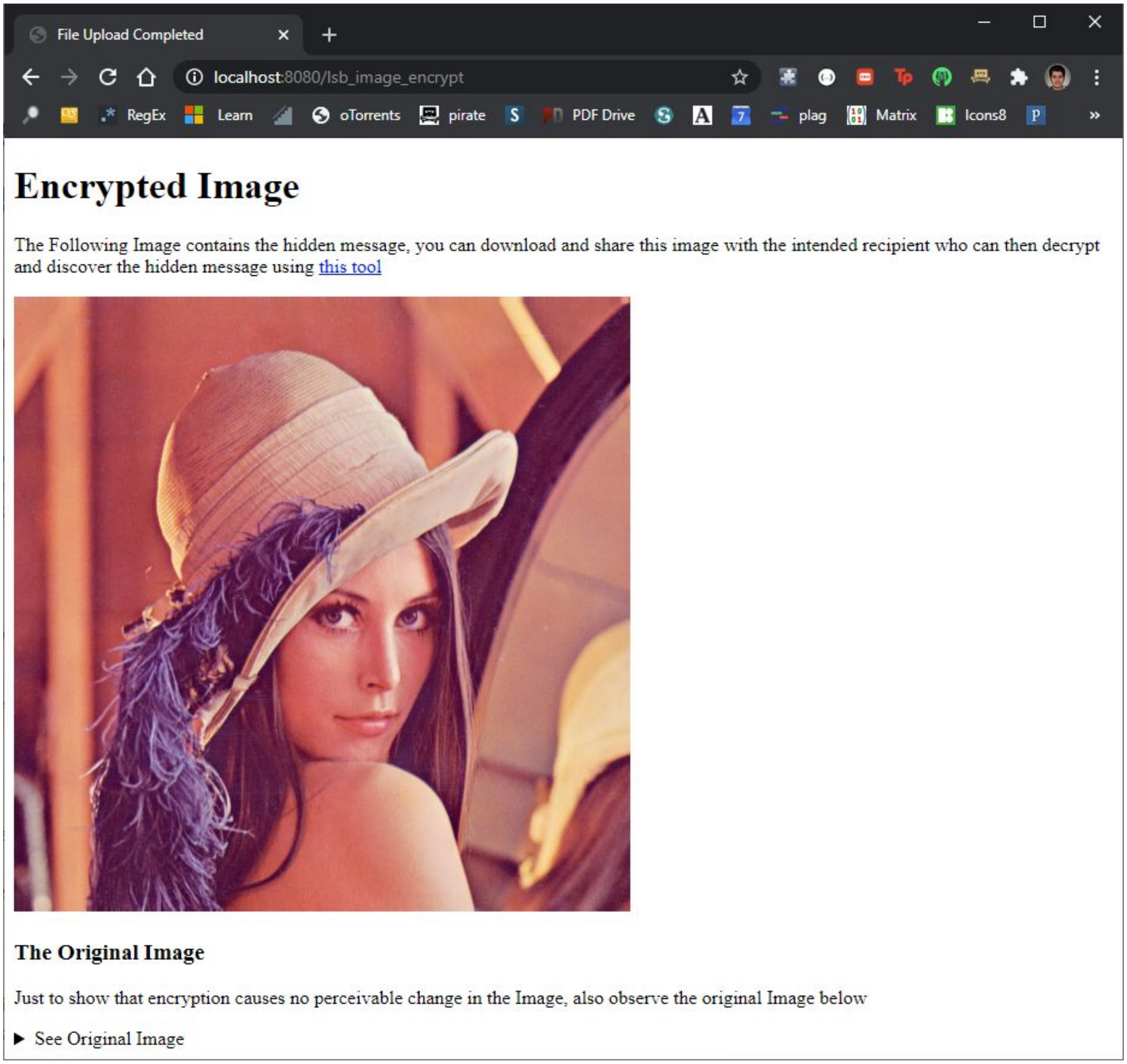
Encrypt Image

lenna.png

We meet tomorrow at 8 pm i

Encrypting an Image with message in the application

We receive the following result:



Encrypted Image

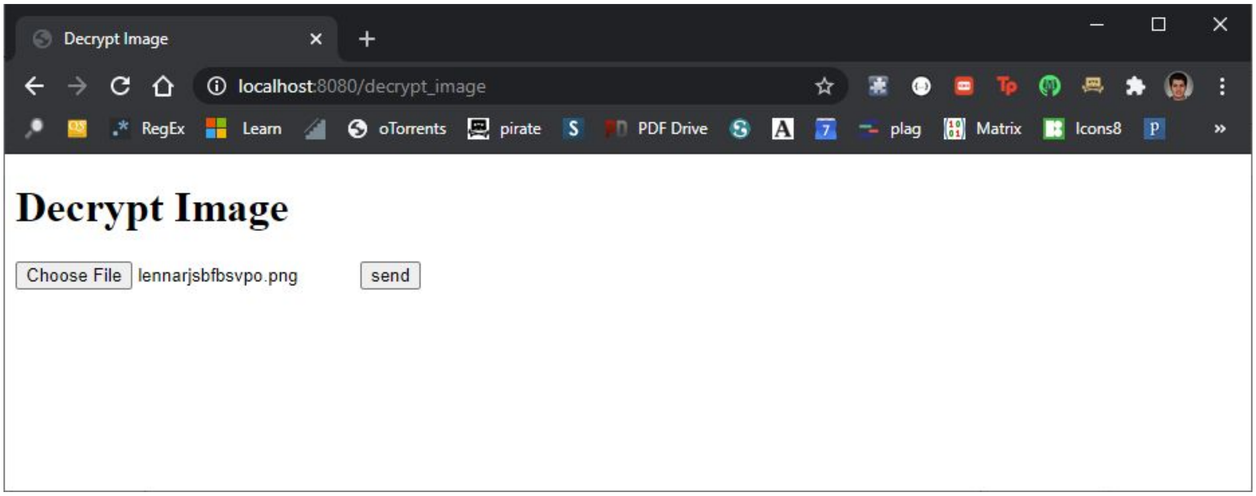
The Following Image contains the hidden message, you can download and share this image with the intended recipient who can then decrypt and discover the hidden message using this tool

The Original Image

Just to show that encryption causes no perceivable change in the Image, also observe the original Image below

▶ See Original Image

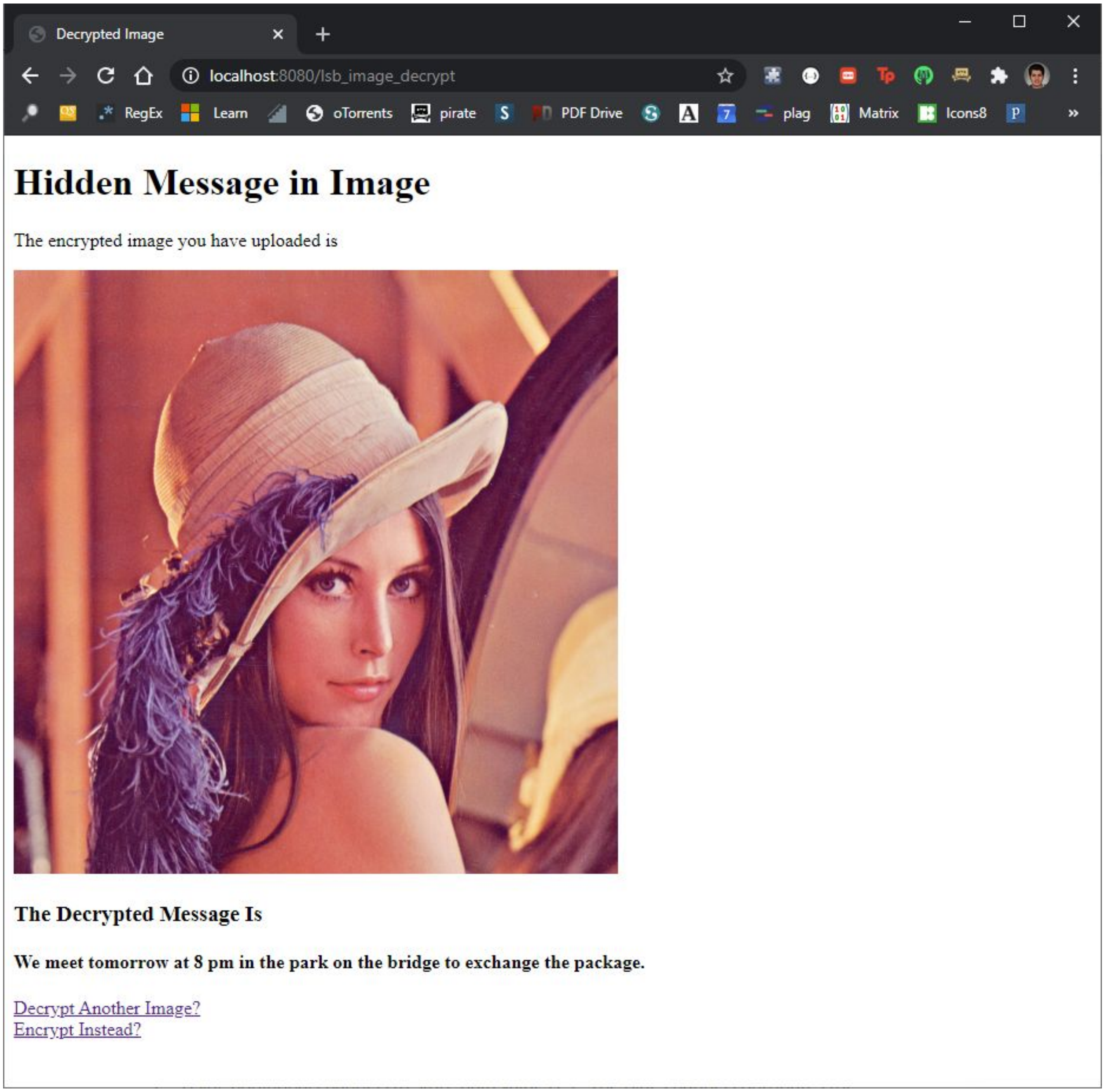
The encrypted Image is given above and as we can see, there is no perceivable difference at all by the human eye, but the message is present. We can now download and retrieve the message from this image using the Decryption Tool.



Decrypt Image

lennarjsbfbsvpo.png

We now retrieve the message from the image and obtain the following result. See in next page.



Hidden Message in Image

The encrypted image you have uploaded is

The Decrypted Message Is

We meet tomorrow at 8 pm in the park on the bridge to exchange the package.

Decrypt Another Image?
Encrypt Instead?

Future Tasks

So far what I have implemented has been the standard LSB Steganography algorithm. The amount of data I can hide currently is directly proportional to the number of pixels i.e. $height \times width$ of the image. To be precise we require 8 pixels for a byte of data (basically a single character) and we are using 3 channels to store the data, so all in all in an Image with height h and width w we can store a message of length .

I wish to further enhance this using the following techniques:

1. Using the concept of super-resolution or simply resolution enhancement to increase the amount of pixel data.
2. Using additional channels to store data values e.g. The Blue channel contributes the least when it comes to visual perception of the image, and hence instead of using a single channel from Red, Blue and Green, we can use 2 from blue and a single one from green.
3. Adding additional cryptographic protection to the data, we can encrypt the message we wish to send using a standard encryption algorithm and store the encrypted message inside the image. The intended receiver will now need to first use steganography to retrieve ciphertext and then use the decryption algorithm to retrieve the plaintext, so now the recipient will need to have the knowledge that the image actually contains a message and also the key for decryption.

I wish to implement some or all of the above for my Cryptography and Computer Security Project for this semester. To view the project at its current state please see it on GitHub here.

Bibliography

1. Hiding Data in Images by Simple LSB Substitution ~Elsevier
2. Image Steganography - Least Significant Bit with Multiple Progressions ~ Springer
3. A Methodology for Using Data Hiding using Images ~IEEE Xplore
4. Forouzan A. - Cryptography and Network Security
5. Digital image steganography Survey and analysis of current methods ~Elsevier
6. Image Pixel Values
7. NumPy
8. OpenCV
9. Python
10. Pip
11. flask
12. flask-cors