Delhi Technological University

19^{th} November 2020

Modifying The Least Significant Bit (LSB) Steganography in Images Method to Increase The Data Volume Transfer Capacity

Cryptography and Network Security (MC-407)

Anish Sachdeva
DTU / 2K16 / MC / 13

Contents

1 Acknowledgements **1**

2 Introduction **2**
2.1 Kirchhoff's Law 2

3 Least Significant Bit (LSB) Stenography **3**

4 Higher Data Density in Image Stenography **7**

5 Algorithm **8**

6 Entropy of Data Hidden In Image **9**
6.1 Image To Grayscale 10
6.1.1 Average of Channels 10
6.1.2 Weighted Averaging Grayscale 11

7 Other Projects **13**
7.1 Annotated Cryptography & Network Security by Forouzan 13
7.2 DES Algorithm 13

8 Conclusion **14**

Bibliography **15**

1 Acknowledgements

I would like to express my special thanks of gratitude to my teacher Prof Dr. Rohit Kumar of the Mathematics Department (DTU) who gave me an amazing opportunity to work on this wonderful project on Cryptography and Network Security (CNS) and enhance the existing Least Significant Bit (LSB) Steganography Algorithm to be able to hide more data in a given image.

In this project I got to read up a lot on the current techniques we have of hiding Data in Images and discovering the basics of Stenography. I further understood in depth the Least Significant Bit (LSB) Algorithm of hiding messages in data by modifying the least significant bits of the pixel RGB Values. I have also gotten a chance to study a little on Information Theory (7) by Shannon and using the weighted perception of Images (8) enhance this method a little so as to be able to encrypt more data into the Image.

I also had the immense pleasure of learning so many various tools and libraries in the Python language (1), namely I have now become highly proficient with the Numpy (3) Library and the OpenCV (2) Library for Computer Vision Operations. I have also used parallel computing in this project and learnt how to accomplish Parallelization sing Python 3.

Secondly I would also like to thank my parents and friends who helped me a lot in finalizing this project (4) within the limited time frame. View project at github.com/anishLearnsToCode/lsb-steganography. (4)

2 Introduction

View the project at github.com/anishLearnsToCode/lsb-steganography.

Steganography (10) is the practice of concealing a file, message, image, or video within another file, message, image, or video. The word steganography comes from Greek steganographia, which combines the words steganós, meaning "covered or concealed", and -graphia meaning "writing".

Cryptography (9), or cryptology is the practice and study of techniques for secure communication in the presence of third parties called adversaries. More generally, cryptography is about constructing and analyzing protocols that prevent third parties or the public from reading private messages; various aspects in information security such as data confidentiality, data integrity, authentication, and non-repudiation are central to modern cryptography.

2.1 Kirchhoff's Law

1. The system must be practically, if not mathematically, indecipherable.
2. It should not require secrecy, and it should not be a problem if it falls into enemy hands.
3. It must be possible to communicate and remember the key without using written notes, and correspondents must be able to change or modify it at will.
4. It must be applicable to telegraph communications.
5. It must be portable, and should not require several persons to handle or operate.
6. Lastly, given the circumstances in which it is to be used, the system must be easy to use and should not be stressful to use or require its users to know and comply with a long list of rules.

In this project we see how we can conceal messages inside standard RGB Images using the Least Significant Bit (LSB) Data Hiding Method, which can also be use in images which have more or less channels than an RGB Image.

We then propose a small change to this method and showcase that higher volumes of data can be concealed using More bits in the image.

3 Least Significant Bit (LSB) Stenography

Least Significant Steganography takes a message that we wish to hide and also takes an Image in which we wish to hide the message. Let the plaintext message be as follows:

$$\text{plaintext} = hello$$

We extract the characters from our plaintext message and convert each character into its corresponding ASCII (11) Values.

$$h, e, l, l, o = 104, 101, 108, 108, 111$$

We will now convert each decimal number into a Binary bit stream to add to the Image

$$h, e, l, l, o = 01101000, 01100101, 01101100, 01101100, 01101111$$

The bit stream is denoted by $[0, 1, 1, 0 \cdots]$. Let the Image be a 3×3 image with 3 channels; RGB. Let us take the Blue channel and denote the Blue level values in a 3×3 matrix as:

$$B = \begin{bmatrix} 100 & 234 & 234 \\ 110 & 234 & 90 \\ 95 & 240 & 92 \end{bmatrix}$$

We now tale the binary of this Blue channel:

$$(B)_2 = \begin{bmatrix} 01100100 & 11101010 & 11101010 \\ 01101110 & 11101010 & 01011010 \\ 01011111 & 11110000 & 01011100 \end{bmatrix}$$

We will now add our bit stream $(0, 1, 1, 0, 1, 0, 0, 0, 0 \cdots)$ that we derived from our plain-text in the last bit of the Blue channel :

$$(B')_2 = \begin{bmatrix} 01100100 & 11101011 & 11101011 \\ 01101110 & 11101011 & 01011010 \\ 01011110 & 11110000 & 01011100 \end{bmatrix}$$

$$(B')_{10} = \begin{bmatrix} 100 & 234 & 235 \\ 110 & 235 & 90 \\ 94 & 240 & 92 \end{bmatrix}$$

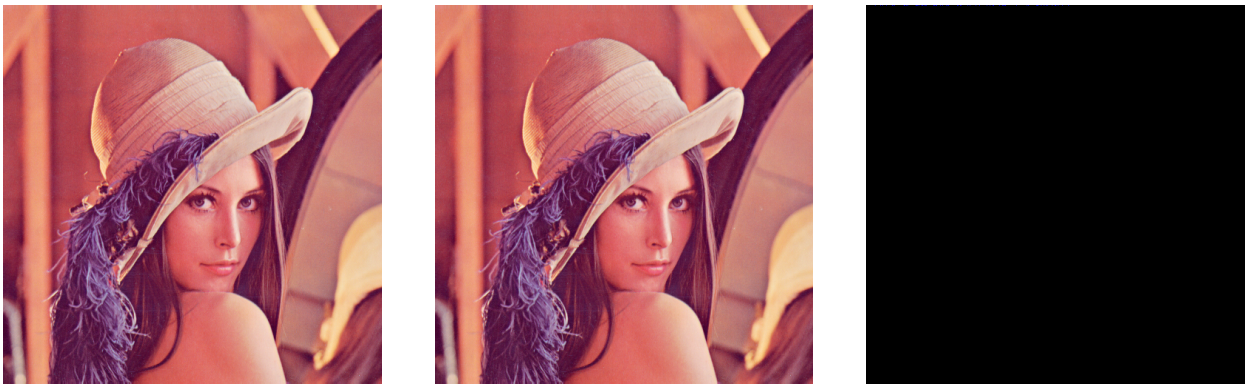
The difference between the original Image Blue channel and the channel after we added the data can be measured as $B' - B$ which will be

$$B' - B = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

The pixel values for a single channel (R, G or B) vary between $[0, 255]$ or 8 bits, and changing the least significant bit changes the pixel values by a very small amount. We see that it remains constant in many cases i.e. $p = 0$ and the maximum change can be ± 1.

We can see from the above image that we can only add 9 bits in the given 3×3 Image if we only hide data in the Blue channel, but if we also perform a similar operation on the Red and Green channels we will be able to hide $3 \times 3 \times 3$ bits of data and store a total of $\lfloor 27/8 \rfloor = 3$ characters of plain-text.

We can similarly extract the hidden message from the Image by creating a bit stream from the least significant bits and then extracting the message from that bit stream. Given below is an example of a standard Digital Signal Processing Image Lenna and the corresponding image with data hidden in the Blue Layer with the difference between them being imperceptible to the Naked eye, but visible on further analysis.



(a) Standard Lenna Image

(b) Lenna Image with the plaintext hidden inside the Image

(c) Difference Between the 2 Images

Figure 1: LSB Steganography applied to an image and the difference as we can see is imperceptible

If we add more data e.g. 10000 characters stored in the Image the result will be as follows:

(a) Standard Lenna Image

(b) Lenna Image with the plaintext of length 10,000 characters hidden inside the Image

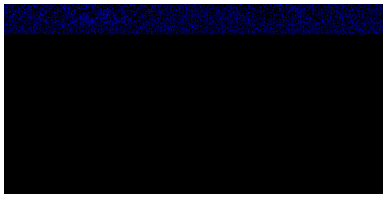
(c) Difference Between the 2 Images

Figure 2: LSB Steganography applied to an image and the difference as we can see is in the blue channel pixel with plaintext data of length 10,000

We see that all the data of the 10,000 characters is successfully stored in the Blue channel B and we did not need to hide any in the Green G or Red R channels at all. We can see similar results with other images as well.
We observe the same thing with LSB Steganography with another image as well. We can increase the number of characters being stored in our image and see that it will need more space to store the data and will eventually occupy other channels and not just Blue.

(a) Sample Image

(b) Image with the plaintext of length 10,000 characters hidden inside the Image

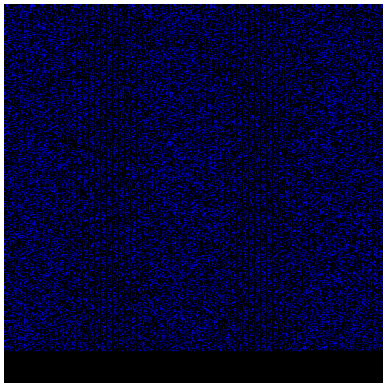


(c) Difference Between the 2 Images

Figure 3: LSB Steganography applied to an image and the difference as we can see is in the blue channel pixel with plaintext data of length 10,000

(a) Standard Lenna Image

(b) Lenna Image with the plaintext of length 30,000 characters hidden inside the Image

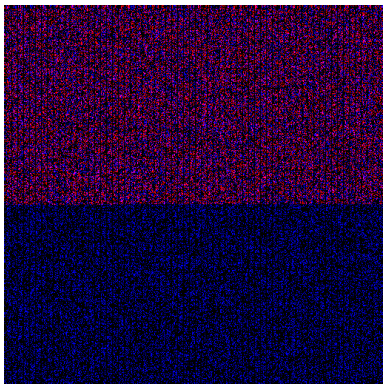


(c) Difference Between the 2 Images

Figure 4: LSB Steganography applied to an image and the difference as we can see is in the blue channel pixel with plaintext data of length 30,000

(a) Standard Lenna Image

(b) Lenna Image with the plaintext of length 50,000 characters hidden inside the Image

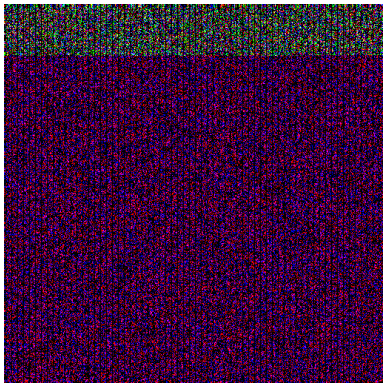


(c) Difference Between the 2 Images

Figure 5: LSB Steganography applied to an image and the difference as we can see is in the blue channel pixel with plaintext data of length 50,000

(a) Standard Lenna Image

(b) Lenna Image with the plaintext of length 70,000 characters hidden inside the Image

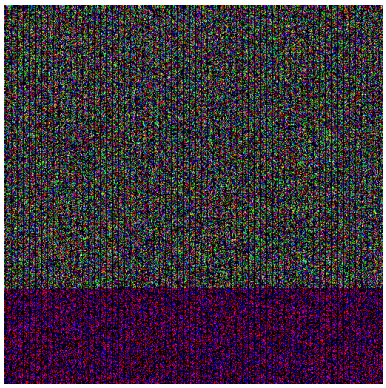


(c) Difference Between the 2 Images

Figure 6: LSB Steganography applied to an image and the difference as we can see is in the blue channel pixel with plaintext data of length 70,000

(a) Standard Lenna Image

(b) Lenna Image with the plaintext of length 90,000 characters hidden inside the Image



(c) Difference Between the 2 Images

Figure 7: LSB Steganography applied to an image and the difference as we can see is in the blue channel pixel with plaintext data of length 90,000

4 Higher Data Density in Image Stenography

If we continue in the manner above we will eventually exhaust the Data Capacity offered by the LSB Selenography technique which is $\lfloor 3wh/8 \rfloor$ characters. We propose that instead of just using the Least Significant Bits for storing data, we instead use the last 2 significant bits to store data hence effectively increasing the Data storage capacity by a factor of 2. The next question arises in what order to add the pixels and how as to store the length of the data in the Image.

The maximum data that can be stored in an Image I of dimensions (w, h) with 3 channels, namely RGB is denoted by:

$$D = \left\lfloor \frac{6wh}{8} \right\rfloor \text{ characters and}$$
$$B = \frac{6wh}{8} \text{ Bits}$$

The number of bits required for storing the length of the plaintext message in the Image will be:

$$N_b(I) = \left\lceil \lg \frac{6wh}{8} \right\rceil \ Bits$$

Hence the total number of bits required for the plaintext message will be:

$$N_b(|M|) = n_b(|P|) + B$$
$$N_b(|M|) = \left\lceil \lg \frac{6wh}{8} \right\rceil + \frac{6wh}{8}$$

So, we will first receive the plaintext message P from the user along with the image I. We will convert the plaintext message to a bit stream from the characters using the ASCII table. We will then compute the maximum storage of the Image i.e. $N_b(I)$ and we will obtain the length of the plaintext message $|P|$ in binary of length $N_b(I)$ and add that as the first bits in our previously obtained bit stream.

We will then need to add our length and plaintext stream, which we will now call the message M stream (which contains the length as the head of this stream) into the 2 least significant bit streams of our Image I.

5 Algorithm

We will now describe the algorithms for hiding the plaintext message P in the Image I and also extracting the message from the Image.

Algorithm 1 Hiding the Plaintext data P in the Image I

Require: Image I
Require: Plaintext P
Determine the maximum data volume of the Image I
$V \leftarrow \lceil \lg \frac{6wh}{8} \rceil$
$l_{10} \leftarrow |P|$
$l_2 \leftarrow l_{10}$ as binary
Pad extra 0's in front of l_2 such that $|l_2| = V$
$S \leftarrow$ Binary stream from the Plaintext Message P using ASCII Code
$M \leftarrow l_2||S$
$R, G, B \leftarrow$ Extract color channels from Image I
$C \leftarrow B||R||G$
Add message stream M to the channel stream C. First add bits in the least significant position for every member of channel stream C and once all bits at position 8 have been used then start adding at position 7
$R, G, B \leftarrow C$ (Extracting the Color channels back from the updated channel stream)
return New Image from the updated R, G, B Channel

Algorithm 2 Extracting the Hidden Plaintext Message from an Image I

Require: Image I
$R, G, B \leftarrow I$ (Extract the color channels from the Image)
$S_B \leftarrow$ Blue channel B as binary stream of bits
$S_R \leftarrow$ Red channel R as binary stream of bits
$S_G \leftarrow$ Green channel G as binary stream of bits
$S \leftarrow S_B||S_R||S_G$ (Creating a stream for the Image where the data is hidden)
$N_b(I) = \lceil \lg(6wh/8) \rceil$
$S_{|P|} \leftarrow$ Bit stream from $S_1 \cdots S_{N_b(I)}$
$|P| \leftarrow (S_{|P|})_{10}$
$S_P \leftarrow$ bit stream from bit $S_{N_b(I)+1} \cdots S_{N_b(I)+1+8|P|}$
for all Bytes b in S_P **do**
  $P \leftarrow P||$ character(b)
end for
return P

6 Entropy of Data Hidden In Image

The bits that we sue for hiding data do not hold the same weight for every bit in the Image. Least significant Bits for any pixel and for any particular channel; Red, Green or Blue will have less weight for a higher bit for the same pixel. e.g. for some arbitrary pixel p let the pixel values for GB be $(101, 100, 204)$ and in binary these are represented as $(01100101, 01100100, 11001100)$.

If we modify the last bits of each of the values by flipping from $0 \to 1$ and $1 \to 0$, we get $(01100100, 01100101, 11001101)$. This represented in base 10 will be $(100, 101, 205)$. We see that this is negligible difference and will not cause any perceivable difference in the images.

(a) Standard Lenna Image

(b) Lenna Image with the Least Significant Bits in RBG Channel flipped

Figure 8: Result of Least Significant Bit Flipping in Images

We can apply this to other images as well, and see that no perceivable difference is caused:

(a) Standard Image

(b) Image with the Least Significant Bits in RGB Channel flipped

Figure 9: Result of Least Significant Bit Flipping in Images

We now introduce Shannon's entropy as defined as:

$$H = -\sum p_i \log p_i \tag{1}$$

Now, the probability of the value of a it changing when we modify the least significant bit (bit at 8^{th} position) is:

$$P_b(8) = 2^0/(2^8 - 1) = 1/255 = 3.92 \times 10^{-3}$$

Currently the weights for each channel in the LSB method are assumed equal i.e. changing the blue channel's least significant bit will cause just as much of a difference as the Red channel. This assumption is incorrect and doesn't take into account that human perception of each color channel isn't equal and we in fact do associate different weights to

different color channels.

6.1 Image To Grayscale

When we wish to convert an image to grayscale, we have 2 methods:

1. Average of RGB Channels
2. Weighted Average of Channels

6.1.1 Average of Channels

In normal average grayscale method we obtain our gray image as

$$G = \left\lfloor \frac{1}{3}(R + G + B) \right\rfloor$$

(a) Standard Image

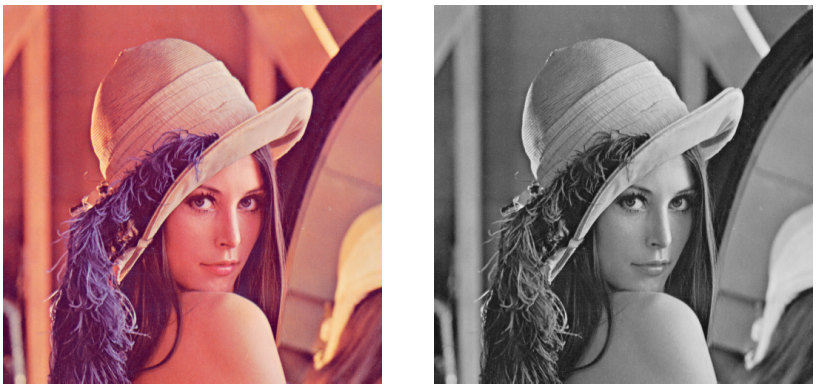
(b) Image with the Least Significant Bits in RGB Channel flipped

Figure 10: Result of Averaging Channels Grayscale

6.1.2 Weighted Averaging Grayscale

In reality the human eye perceives different channels with different weights and hence in standard Computer Vision / Computer Graphics the following formula is used:

$$G = 0.2126 \cdot R + 0.7152 \cdot G + 0.0722 \cdot B \tag{2}$$



(a) Standard Image

(b) Image with the Least Significant Bits in RGB Channel flipped

Figure 11: Result of Weighted Average Channels Grayscale

We can clearly see that the weighted average results are considerable better. From the above conversation we state that humans do not perceived each color channel equally and in fact there is a different perception weight attached to each color bit. We now take these perception weights into account to define the probability of image modification that the least significant bit of a color may hold:

$$\begin{aligned} P_B &= 0.0722 \cdot P_b \\ P_R &= 0.2126 \cdot P_b \\ P_G &= 0.7152 \cdot P_b \end{aligned}$$

The entropies of the 3 different color channels for the same pixel are denoted as:

$$\begin{aligned} H_B &= -P_B \lg P_B = 3.33 \times 10^{-3} \\ H_R &= -P_R \lg P_R = 8.52 \times 10^{-3} \\ H_G &= -P_G \lg P_G = 2.37 \times 10^{-2} \end{aligned}$$

If we use every pixel in our Image of dimensions $(w, h, 3)$ the total entropy of our Image will be:

$$\begin{aligned} H_B(I) &= wh3.33 \times 10^{-3} \\ H_R(I) &= wh8.52 \times 10^{-3} \\ H_G(I) &= wh2.37 \times 10^{-2} \end{aligned}$$

Total Entropy for the LSB Steganography for the entire Image will be:

$$H(I) = H_B(I) + H_R(I) + H_G(I) = 3.533 \times 10^{-2} \ \ wh \tag{3}$$

We can also utilise more pixels in our image, namely the second least significant bits and obtain even more data volume and if we follow the order of leas entropy i.e. Blue, Red and Green we will obtain Images which are least perceivable different than the Original Image.

If we store data in the second least significant bit the change in probability obtained is (the 7^{th} bit in the number):

$$P_b(7) = 2^1/(2^8 - 1) = 2/255 = 7.84 \times 10^{-3} \tag{4}$$

The Respective channel probabilities will be :

$$\begin{aligned} P_B(7) &= 0.0722 \cdot P_b(7) = 5.66 \times 10^{-4} \\ P_R(7) &= 0.2126 \cdot P_b(7) = 1.66 \times 10^{-3} \\ P_G(7) &= 0.7152 \cdot P_b(7) = 5.60 \times 10^{-3} \end{aligned}$$

The corresponding entropies for single pixel will be:

$$\begin{aligned} H_{B,7} &= -P_B(7) \lg P_B(7) = 6.10 \times 10^{-3} \\ H_{R,7} &= -P_R(7) \lg P_R(7) = 1.53 \times 10^{-2} \\ H_{G,7} &= -P_G(7) \lg P_G(7) = 4.19 \times 10^{-2} \end{aligned}$$

The total entropy of the Image if we modify every single second least significant bits of all color channels in every pixel of the Image with dimensions $(w, h, 3)$:

$$H_7(I) = (H_{B,7} + H_{R,7} + H_{G,7}) \cdot wh = 6.33 \times 10^{-2} \ \ wh \tag{5}$$

If we use both; the least significant bit and the second least significant bit, the total entropy of the Image Difference will be:

$$H_{7,8}(I) = H_8(I) + H_7(I) = 9.863 \times 10^{-2} \ \ wh \tag{6}$$

7 Other Projects

7.1 Annotated Cryptography & Network Security by Forouzan

I have created a repository (12) that covers every aspect of the book on Cryptography and Network Security by Behrouz A. Forouzan and covers all mathematical and cipher constructs with explanation, code and running examples along with a live Jupyter notebook in Python.

7.2 DES Algorithm

I have implemented the DES Algorithm (13) with proper explanation into the algorithm along with the mathematical constructs such as the P-Box, S-Box etc. This is also covered like the previous project with full explanation, code and examples in a Jupyter Notebook.

8 Conclusion

Steganography works on the principles of security by obscurity whereas Cryptography is more firmly based in Mathematics and doesn't rely on obscurity for security. In a cryptographic system by following Kirchhoff's Law, the entire system and how the data will be transferred, enciphered and deciphered and the full inner workings of the Cryptographic system are put up on display.

So, the entire security of the system lies on the solid mathematical foundations and proofs by countless people trying to penetrate the system. The algorithm is published beforehand and then people try to come up with ways on how to break it. This gives the organization a sense of how strong the algorithm is and how they can improve and work on any vulnerabilities.

Steganography on the other hand tries to hide the fact that data is being hidden in some manner as it is relying on people not looking at a particular part of the infrastructure at all. In the above application the method of extracting the data from the image can't be published openly as then anyone will be able to extract any message from the Image, hence we need to treat the algorithm as the key in this program. For 2 parties communicating amongst themselves they need to understand that data shared using LSB Image Steganography is not guaranteed as being safe or authenticated and a random picture between 2 parties will be considered normal, but 2 people or more persons communicating solely in pictures will arouse a lot of suspicion, hence this method although very good when used sparsely can't be relied upon for extra safety when used in abundance.

To guarantee that this method is as secure as the Diffie-Hellman Key Exchange or the RSA method, we need to add additionally in this method the Use of a key. First we will use standard Encryption to encrypt our message using the secret key and then hide this encrypted message in the Image, adding an additional security parameter. We can then convert our image into Binary fragments for transport over a network and encrypt each binary fragment individually for transportation over a public network.

The attacker Eve will first need to collect all Binary blobs that represent the Image. Then the attacker will need to decrypt these blocks and will require a key for that. If the attacker does accomplish that she can combine these blobs and then see the Image we were sending. The attacker will think that the Image was the item of import and the message we were trying to send and will not try to imagine that there is in fact additional information in the image.

If the attacker indeed does try to extract additional information from the Image, she will need the algorithm used to hide data in the Image. In this case we are using LSB Steganography, and too a modified version but there are several other methods available. The attacker will then have to try several methods to obtain a text that the attacker feels is valid, but wait!

We had encrypted the message before hiding it in the Image hence the correct extraction of the message will be incoherent and will eventually require a key to decrypt. This key is something that Eve doesn't possess. Hence by adding 2 additional keys in Image Steganography we can make this a very secure algorithm for high volume data interchange with extra security and secrecy.

Bibliography

[1] Python 3

[2] Python: OpenCV

[3] Python: Numpy

[4] **Project**: LSB Image Steganography with High Density Message Transfer

[5] **Introduction to Applied Cryptography** by University of Colorado, Specialization on Coursera

[6] **Applied Cryptography** by University of Colorado, Specialization on Coursera

[7] **A Mathematical Theory Of Communication** by Claude E. Shannon *The Bell System Technical Journal*, Vol. 27, pp. 379–423, 623–656, July, October, 1948.

[8] **How To Convert an Image To Grayscale?** by *K. D. Nuggets*

[9] Wikipedia: Cryptography

[10] Wikipedia: Steganography

[11] **ASCII**: American Standard for Information Interchange

[12] Annotated Cryptography, Running Code with Explanations of the Cryptography Network Security Book Forouzan

[13] An explanation with Code of the complete DES Algorithm