

Programme Name: \_\_\_\_\_\_\_\_**BCS HONS**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Course Code: \_\_**CSC 2730**\_\_\_\_\_\_\_

Course Name: \_\_\_\_\_\_\_\_**Network And Data Security**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Individual Project No. \_**1**\_\_\_

Date of Submission: \_\_\_\_\_\_**2/1/2020**\_\_\_\_\_\_\_\_\_\_\_\_\_

**Submitted By: Submitted To:**

Student Name**: Dipesh Tha Shrestha** Faculty Name**: Manoj Gautam**

IUKL ID: **041902900028** Department**: LMS**

Semester**: Third Semester**

Intake**: September 2019**

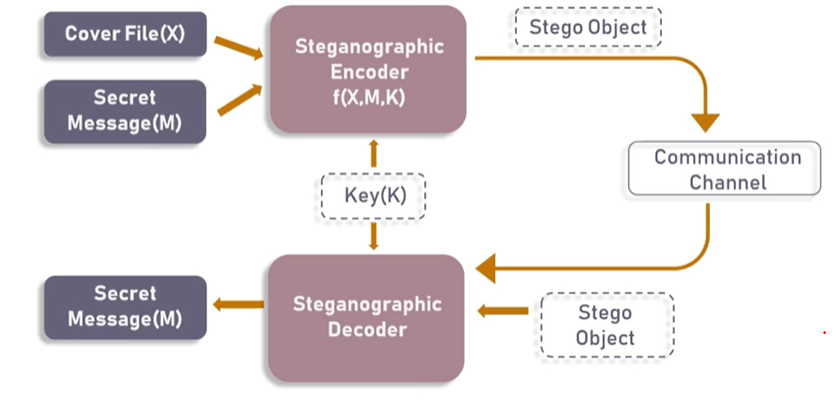
**Steganography**

Steganography is the art and science of embedding secret messages in a cover message in such a way that no one, apart from the sender and intended recipient, suspects the existence of the message

Steganography is the practice of hiding a secret message inside of (or even on top of) something that is not secret. That something can be just about anything you want. These days, many examples of steganography involve embedding a secret piece of text inside of a picture. Or hiding a secret message or script inside of a Word or Excel document.

The purpose of steganography is to conceal and deceive. It is a form of covert communication and can involve the use of any medium to hide messages. It’s not a form of cryptography, because it doesn’t involve scrambling data or using a key. Instead, it is a form of data hiding and can be executed in clever ways. Where cryptography is a science that largely enables privacy, steganography is a practice that enables secrecy – and deceit.

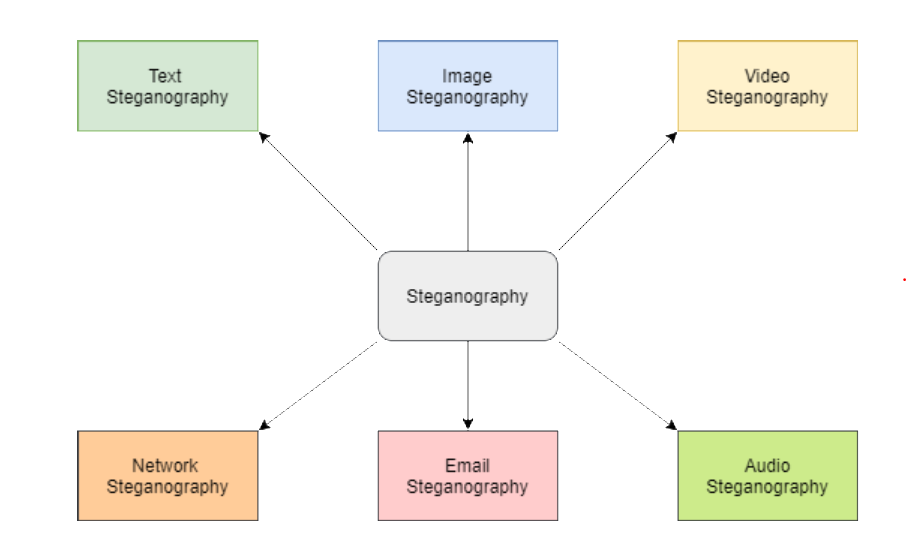
**Basic Steganographic Model**



As seen in the above image, both the original image file(X) and secret message (M) that needs to be hidden are fed into a steganographic encoder as input. Steganographic Encoder function, f(X,M,K) embeds the secret message into a cover image file by using techniques like least significant bit encoding. The resulting stego image looks very similar to your cover image file, with no visible changes. This completes encoding. To retrieve the secret message, stego object is fed into Steganographic Decoder.

**Types of Steganography**

Steganography works have been carried out on different transmission media like images, video, text, or audio.



**Image Steganography**

* Due to the large amount of redundancy created in the manner in which digital images are represented, images are the most appropriate carrier type for steganography. Steganography on images is also the most popular form of steganography, since images occur frequently on websites, as e-mail attachments, etc. There is thus minimum cause for suspicion when a digital image is used.
* The image Steganography is used to hide a secret message inside an image. The most widely used technique to hide secret bit inside the LSB of the cover image. Because this method uses bits of each pixel in the image, it is necessary to use a lossless compression format, otherwise the hidden information will get lost in the transformations of a lossy compression algorithm.
* When using a 24 bit color image, a bit of each of the red, green and blue color components can be used, so a total of 3 bits can be used for each pixel, in this way we can use more secret bit to hide data in it.

*#import all the required libraries*

**import** **cv2**

**import** **numpy** **as** **np**

**import** **types**

**from** **google.colab.patches** **import** cv2\_imshow *#Google colab crashes if you try to display*

*#image using cv2.imshow() thus use this import*

**if** type(message) == str:

**return** ''.join([ format(ord(i), "08b") **for** i **in** message ])

**elif** type(message) == bytes **or** type(message) == np.ndarray:

**return** [ format(i, "08b") **for** i **in** message ]

**elif** type(message) == int **or** type(message) == np.uint8:

**return** format(message, "08b")

**else**:

**raise** **TypeError**("Input type not supported")

*# Function to hide the secret message into the image*

**def** hideData(image, secret\_message):

*# calculate the maximum bytes to encode*

n\_bytes = image.shape[0] \* image.shape[1] \* 3 // 8

print("Maximum bytes to encode:", n\_bytes)

*#Check if the number of bytes to encode is less than the maximum bytes in the image*

**if** len(secret\_message) > n\_bytes:

**raise** **ValueError**("Error encountered insufficient bytes, need bigger image or less data !!")

secret\_message += "#####" *# you can use any string as the delimeter*

data\_index = 0

*# convert input data to binary format using messageToBinary() fucntion*

binary\_secret\_msg = messageToBinary(secret\_message)

data\_len = len(binary\_secret\_msg) *#Find the length of data that needs to be hidden*

**for** values **in** image:

**for** pixel **in** values:

*# convert RGB values to binary format*

r, g, b = messageToBinary(pixel)

*# modify the least significant bit only if there is still data to store*

**if** data\_index < data\_len:

*# hide the data into least significant bit of red pixel*

pixel[0] = int(r[:-1] + binary\_secret\_msg[data\_index], 2)

data\_index += 1

**if** data\_index < data\_len:

*# hide the data into least significant bit of green pixel*

pixel[1] = int(g[:-1] + binary\_secret\_msg[data\_index], 2)

data\_index += 1

**if** data\_index < data\_len:

*# hide the data into least significant bit of blue pixel*

pixel[2] = int(b[:-1] + binary\_secret\_msg[data\_index], 2)

data\_index += 1

*# if data is encoded, just break out of the loop*

**if** data\_index >= data\_len:

**break**

**return** image

**def** showData(image):

binary\_data = ""

**for** values **in** image:

**for** pixel **in** values:

r, g, b = messageToBinary(pixel) *#convert the red,green and blue values into binary format*

binary\_data += r[-1] *#extracting data from the least significant bit of red pixel*

binary\_data += g[-1] *#extracting data from the least significant bit of red pixel*

binary\_data += b[-1] *#extracting data from the least significant bit of red pixel*

*# split by 8-bits*

all\_bytes = [ binary\_data[i: i+8] **for** i **in** range(0, len(binary\_data), 8) ]

*# convert from bits to characters*

decoded\_data = ""

**for** byte **in** all\_bytes:

decoded\_data += chr(int(byte, 2))

**if** decoded\_data[-5:] == "#####": *#check if we have reached the delimeter which is "#####"*

**break**

*#print(decoded\_data)*

**return** decoded\_data[:-5] *#remove the delimeter to show the original hidden message*

*# Encode data into image*

**def** encode\_text():

image\_name = input("Enter image name(with extension): ")

image = cv2.imread(image\_name) *# Read the input image using OpenCV-Python.*

*#It is a library of Python bindings designed to solve computer vision problems.*

*#details of the image*

print("The shape of the image is: ",image.shape) *#check the shape of image to calculate the number of bytes in it*

print("The original image is as shown below: ")

resized\_image = cv2.resize(image, (500, 500)) *#resize the image as per your requirement*

cv2\_imshow(resized\_image) *#display the image*

data = input("Enter data to be encoded : ")

**if** (len(data) == 0):

**raise** **ValueError**('Data is empty')

filename = input("Enter the name of new encoded image(with extension): ")

encoded\_image = hideData(image, data) *# call the hideData function to hide the secret message into the selected image*

cv2.imwrite(filename, encoded\_image)

*# Decode the data in the image*

**def** decode\_text():

*# read the image that contains the hidden image*

image\_name = input("Enter the name of the steganographed image that you want to decode (with extension) :")

image = cv2.imread(image\_name) *#read the image using cv2.imread()*

print("The Steganographed image is as shown below: ")

resized\_image = cv2.resize(image, (500, 500)) *#resize the original image as per your requirement*

cv2\_imshow(resized\_image) *#display the Steganographed image*

text = showData(image)

**return** text

*# Image Steganography*

**def** Steganography():

a = input("Image Steganography **\n** 1. Encode the data **\n** 2. Decode the data **\n** Your input is: ")

userinput = int(a)

**if** (userinput == 1):

print("**\n**Encoding....")

encode\_text()

**elif** (userinput == 2):

print("**\n**Decoding....")

print("Decoded message is " + decode\_text())

**else**:

**raise** **Exception**("Enter correct input")

Steganography() *#encode image*