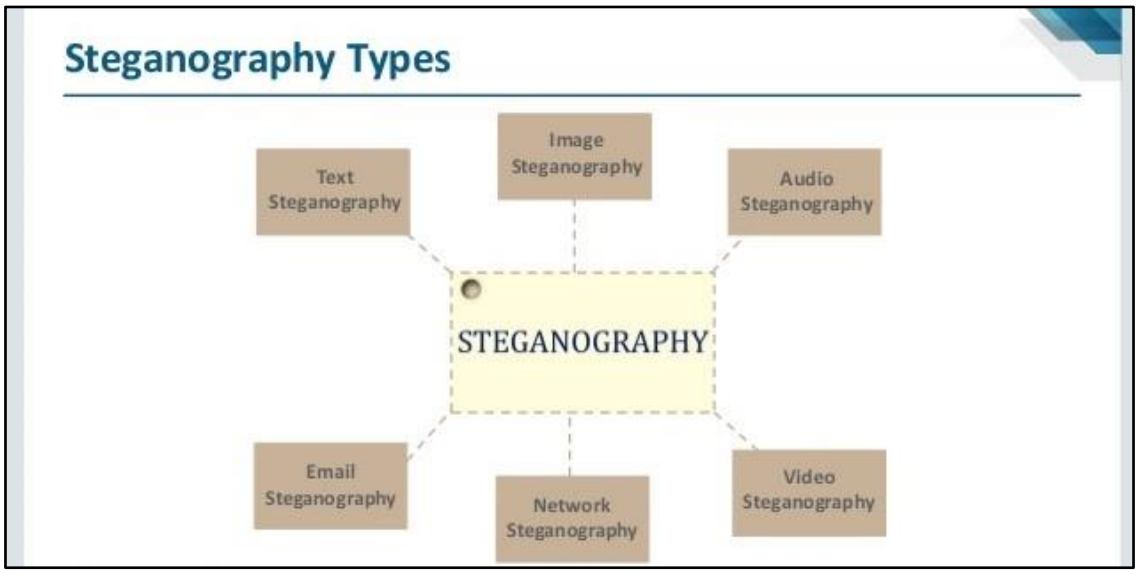
# MULTI-MESSAGE STEGANOGRAPHY TOOL

## Introduction to Steganography

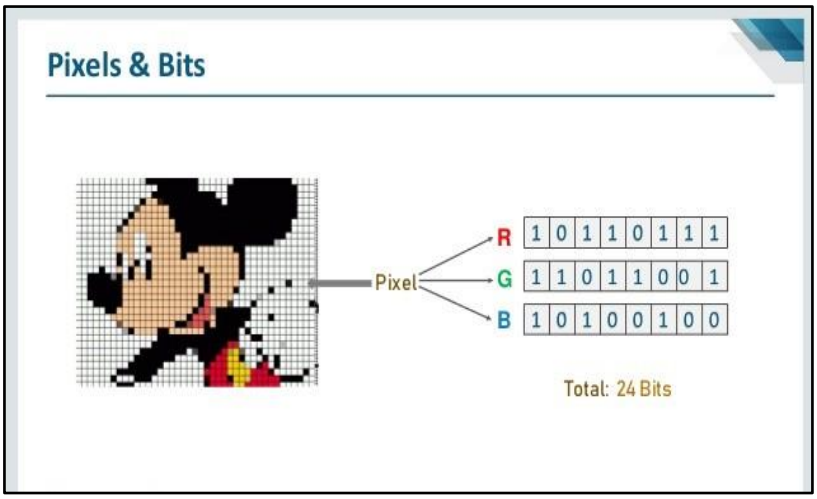
- "Steganography" comes from the Greek word, it means covered or secret writing.
- We will pronounce as STEGAN-O-GRAPHY, where STEGAN means "covered" and GRAPHY means "writing".
- Steganography is the technique of hiding private or sensitive information within something that appears to be a usual image.
- When the user wants to send the secret information the text will be encoded and sent to the receiver, the receiver decodes it to obtain the secret information.




## Image Steganography

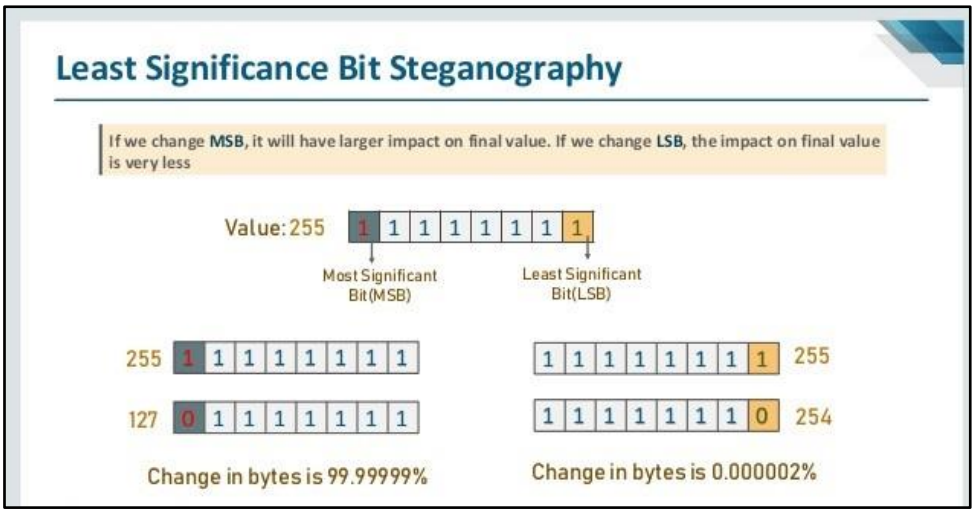
- Image Steganography is the art of embedding secret messages in image file in such a way that no one other than sender and recipient, suspects the existence of the message.
- Therefore, there will be low risk that a third person can access the secret text or information.

## INTRODUCTION TO PIXELS AND BITS




- A pixel is the smallest building of an image and the colors in any pixel are (assume RGB) a function of the combination of proportions of red, green, and blue.
- These pixels are so small that we are unable to see them. Each pixel represents an individual colour. Pixels are blend to form new colours .Thousands or even more pixels groups to form an image.

## Methodology of Solving




- LSB is the most popular Steganographic technique. If we change MSB, there will be large impact on final image.
- Whereas if we change LSB, there will be lower impact on final image.
- Hence, we used LSB technique to obtain better results in image steganography.

**My Solution for encoding multiple messages in an image:**

After the user inputs no. of messages, he wants to encode and the image in which he wants to encode the data, and the messages that needs to be encoded. Internally, all these messages will get stored in a list. From this list we construct a string with '@%' as first two characters of the string, then we add message by message with '#$' as separator between messages. Then we consider this string as the single message that needs to be encoded and we follow LSB technique to encode the string.

**Addition of '@%' and '#$' to the string helps us to decode messages.**

During decoding, we again follow LSB technique to decode the string. Now we check whether our decoded string starts with '@%' or not. if it doesn't start with '@%' then it means there is no hidden message in the image. Otherwise, message is hidden in the image. Now we split the decoded string with '#$' as separator, then we get our list of hidden messages.

Instead of '@%' and '#$' we can use any special characters. But those should be the characters that we generally do not see in any text.

# MILE STONES

## ACHIEVED:

1) Creating Graphical User Interface using TKINTER.
2) Using PIL (Python Imaging Library) and OS modules to work with images.
3) Iterating over pixels of an image to read and modify them.
4) Implementing least significant bit (LSB) steganographic technique for encoding and decoding of data.
5) Providing multi message encryption and decryption.

# HOW TO RUN

## 1) Inputs Required

- **During Encoding**: No. of messages to be encoded, Image to be encoded and messages are to be given by user.
- **During Decoding**: Image to be decoded is to be given by user
- Before running the code, **PIL Library** need to be installed. This can be done by pressing

**pip install pillow** in command prompt / terminal. **(pillow is replacement of PIL)**

- Type of images supported are only PNG images.

**2) Files and their locations**

- **background.png** file should be in the same directory as python file. Because, this image is used as the background in the GUI.

**3) Settings Required**

- Ensure pillow and python are installed.

## Reasons for using only PNG images

1) PNG uses lossless compression, which means it retains all original data when saving. This is crucial for steganography because it allows hiding and retrieving data without loss or alteration.
2) Other formats like JPEG use lossy compression, which discards some data to reduce file size, potentially corrupting hidden messages.
3) PNG allows more space for hidden data without noticeable changes to the image.

## Requirement of Libraries:

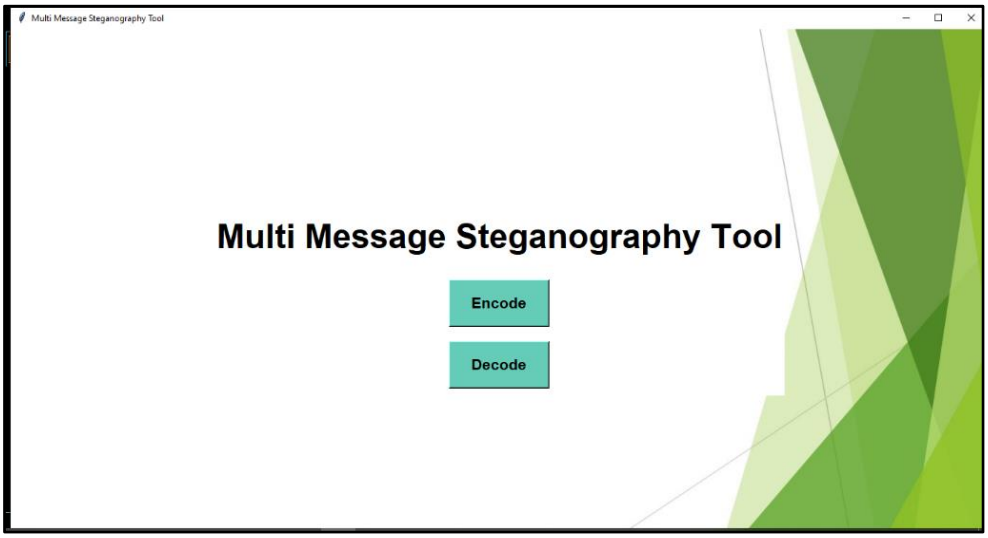
- **Python Imaging Library(PIL)** - Image and ImageTk are imported from PIL. **Image** class is imported to manipulate and process images whereas **ImageTk** class is imported to display PIL images in TKINTER GUI.
- **TKINTER**- filedialog and messagebox modules are imported from the tkinter library. **filedialog** is imported to open or save the files and **messagebox** is imported to display information such as errors etc.
- **OS**-This module is imported to handle file and path operations, primarily for saving encoded image with a new file name.

# TABLE OF FILES USED

| FILE NAME | USE |
|---|---|
| program.py | Python file which is used to run the GUI. |
| background.png | Used as background image in my GUI. |
| PNG format Sample images | To perform encoding and decoding of text. |

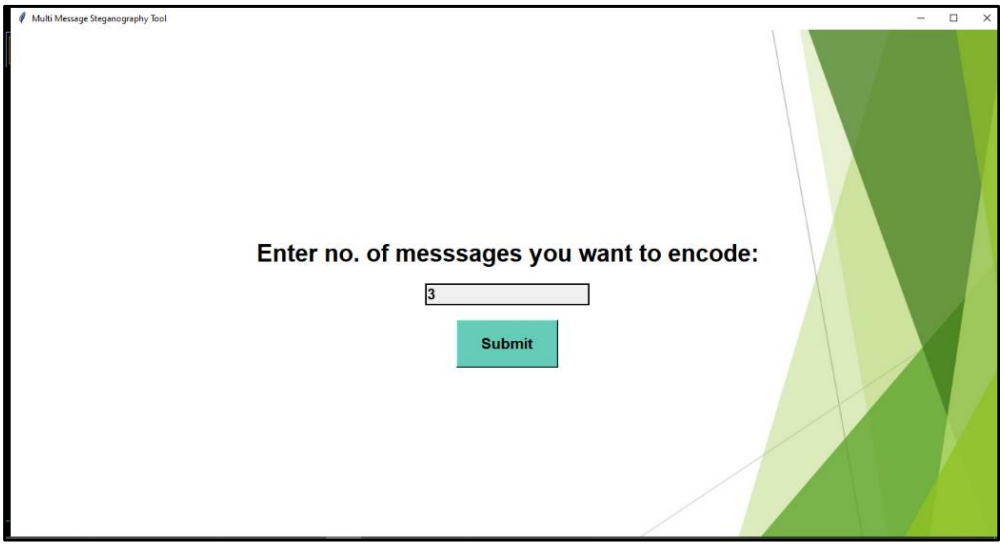
## SCREENSHOTS DESCRIBING ENCODING AND DECODING PROCESS

**1) Main Interface**

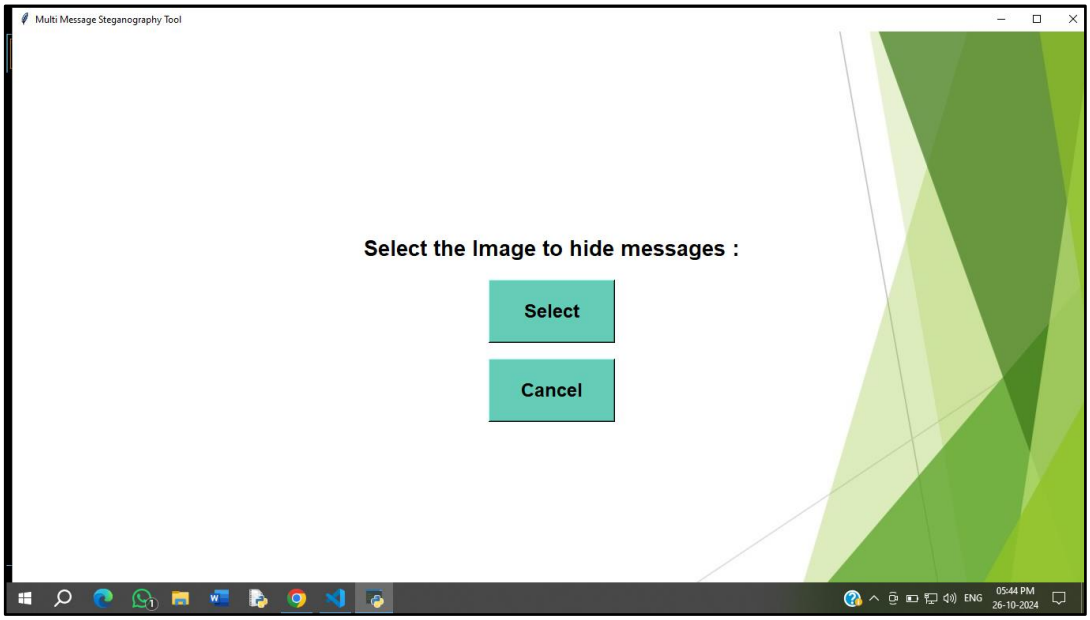



## STEPS FOR ENCODING MESSAGES INTO AN IMAGE

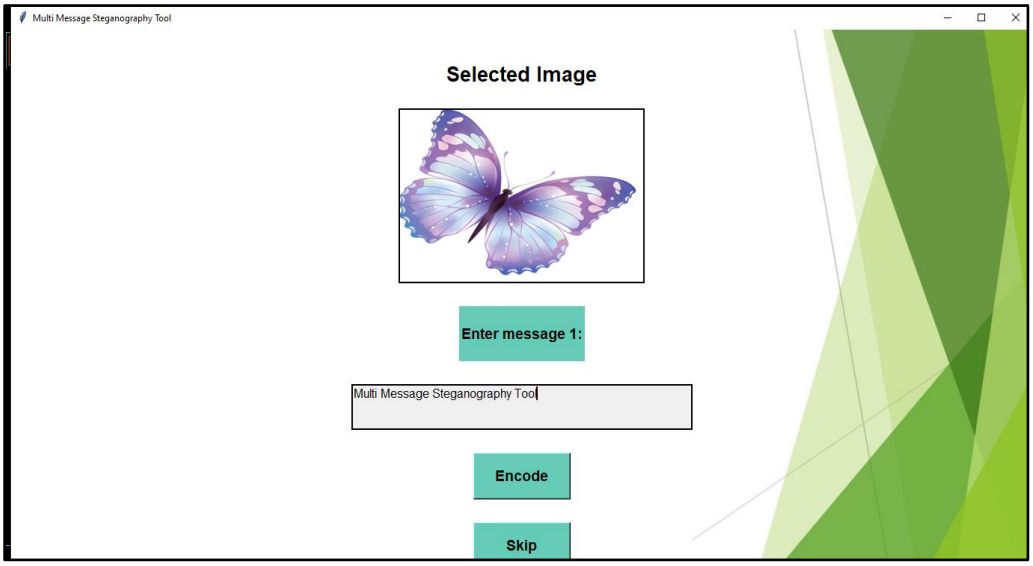
**2) Inputing no. of messages**



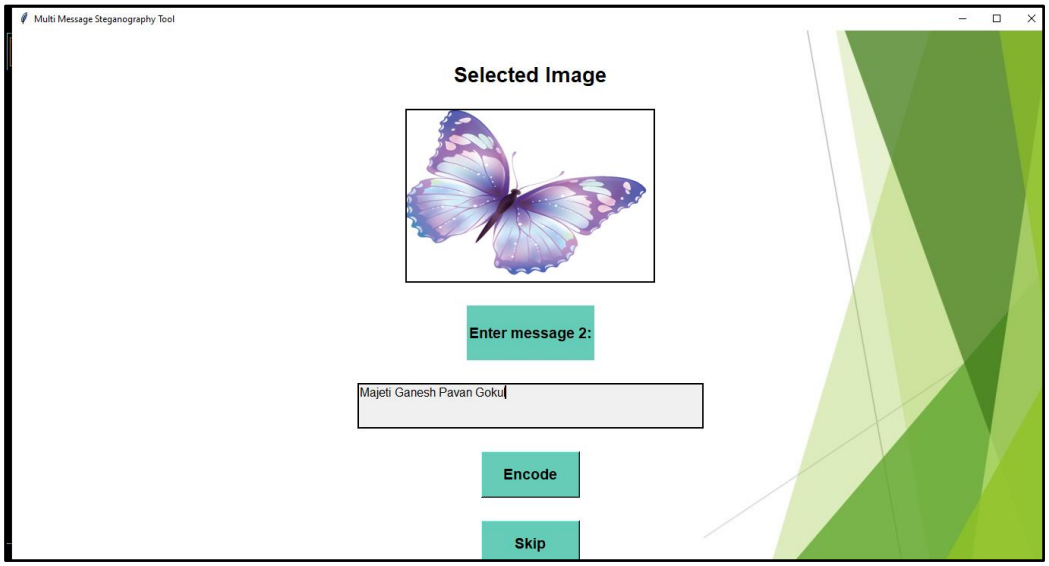
## 3) Selecting the image to encode



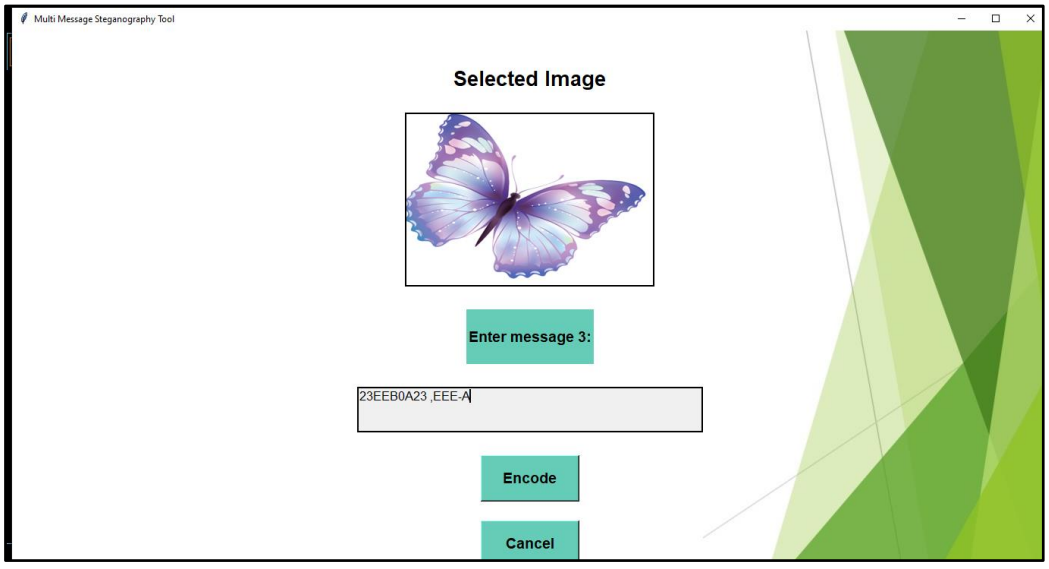

## 4) Inputing Message 1



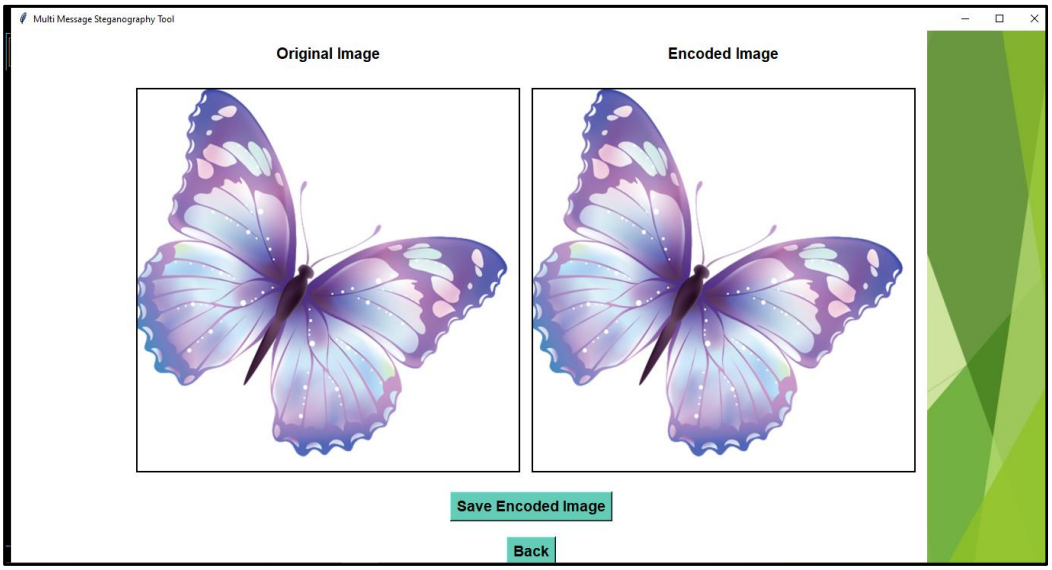
## 5) Inputing Message 2




## 6) Inputing Message 3

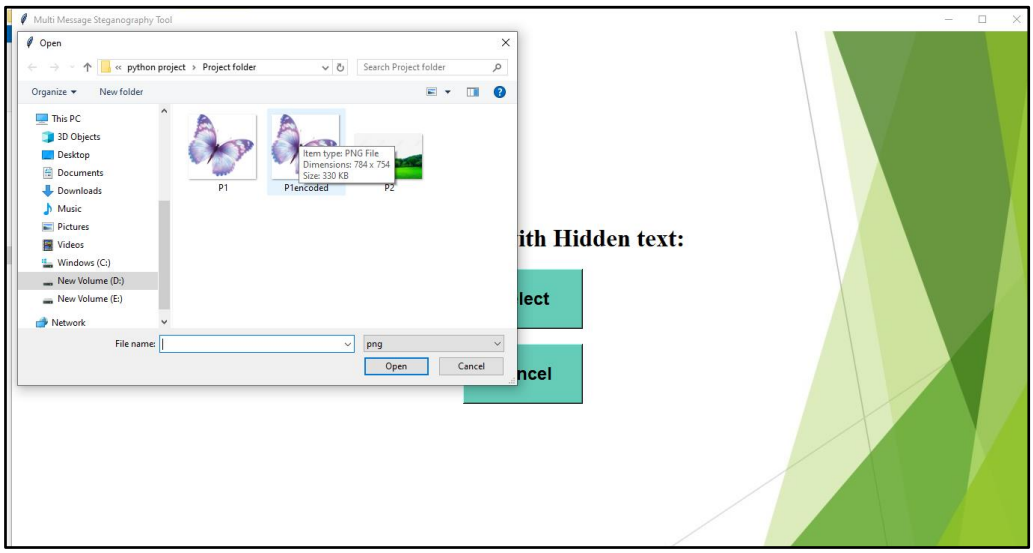


### 7) Saving the encoded image

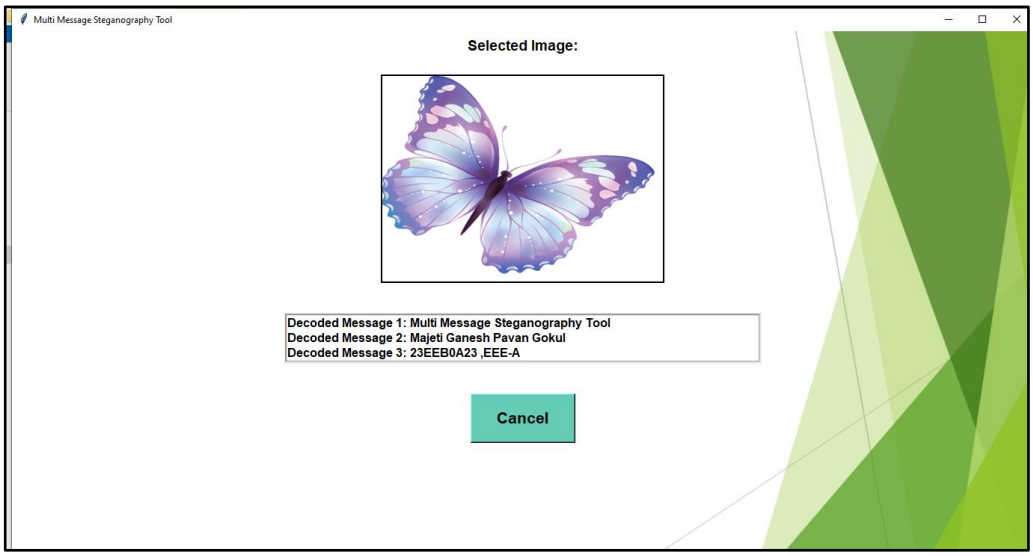



## STEPS TO DECODE MESSAGES FROM AN ENCODED IMAGE

### 8) Selecting the Image to be decoded



**9) Decoded information**




**Name: Majeti Ganesh Pavan Gokul**

**RollNo: 23EEB0A23**

**Section: EEE-A**