A Project Report on

**Steganography**

***21AIE431***

***Applied Cryptography***

***Under the guidance of:***

*Dr Sunil,*

*Department of CEN,*

*Amrita School of Computing.*

***4th June, 2023***

**Submitted By:**

Velamuru Sai Yashwitha Reddy CB.EN.U4AIE21173

***Group - 8***

***Batch - B***

Sadanala Aswitha CB.EN.U4AIE21157

Subikksha CB.EN.U4AIE21167

Pulagala Sai Abhiram CB.EN.U4AIE21148



**AMRITA VISHWA VIDYAPEETHAM**

**ASC, ETTIMADAI, COIMBATORE**

**ACKNOWLEDGEMENT**

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without mentioning about the people whose constant guidance and encouragement made it possible. We take pleasure in presenting before you, our project, which is result of studied blend of both research and knowledge.

We express our earnest gratitude to our internal guide, Dr Sunil, Department of CEN, our project guide, for his constant support, encouragement and guidance. We are grateful for his cooperation and his valuable suggestions.

Finally, we would also like to thank our group mates for the constant encouragement and the team work which was put throughout the project. We could learn about the various concepts in the project with the each other’s help and support.

**DECLARATION**

We the undersigned solemnly declare that the project report is based on our own work carried out during the course of our study under the supervision of Dr Sunil, Department of CEN. We assert the statements made and conclusions drawn are an outcome of our research work. We further certify that

1. The work contained in the report is original and has been done by us under the general supervision of our supervisor.
2. We have followed the guidelines provided by the university in writing the report.

III. Whenever we have used materials (data, theoretical analysis, and text) from other sources, we have given due credit to them in the text of the report and giving their details in the references.

Velamuru Sai Yashwitha Reddy Sadanala Aswitha

Subikksha Pulagala Sai Abhiram

**ABSTRACT**

In this project we hide the message with in the image and a text file. In this project, the sender selects a cover file (image, text) with secret text and hide it into the cover file by using different efficient algorithm and generate a stego file of same format as our cover file (image, text). Then the stego file is sent to the destination with the help of private or public communication networks. On the other side i.e. receiver, the receiver downloads the stego file and by using the appropriate decoding algorithm retrieves the secret text that is hidden in the stego file.

For text steganography specifically, we employ a technique involving Zero-Width Characters (ZWCs). These ZWCs are unique Unicode characters with no visible representation in digital text.

Further, this project delves into image steganography, focusing on the implementation of the Modified Least Significant Bit (LSB) algorithm, a popular and efficient method for embedding data within images

**Text Steganography**

ENCODE:-

ZWCs- In Unicode, there are specific zero-width characters (ZWC) that are used to control special entities such as Zero Width Non Joiner (e.g., ZWNJ separates two letters in special languages) and POP directional, which have no written symbol or width in digital text . In social media, if it utilizes the Unicode standard in order to process digital texts in different languages, then the ZWCs show invisible written symbols; otherwise, they might generate unconventional symbols . We used four ZWCs for hiding the Secret message through the cover text.

The embedding algorithm contains following stages:-

Secret Message (SM): This can be secret or confidential information.

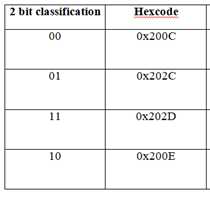
Cover Text (CT): This is an innocent text that can be any type of meaningful text.

For every character of the secret message :-

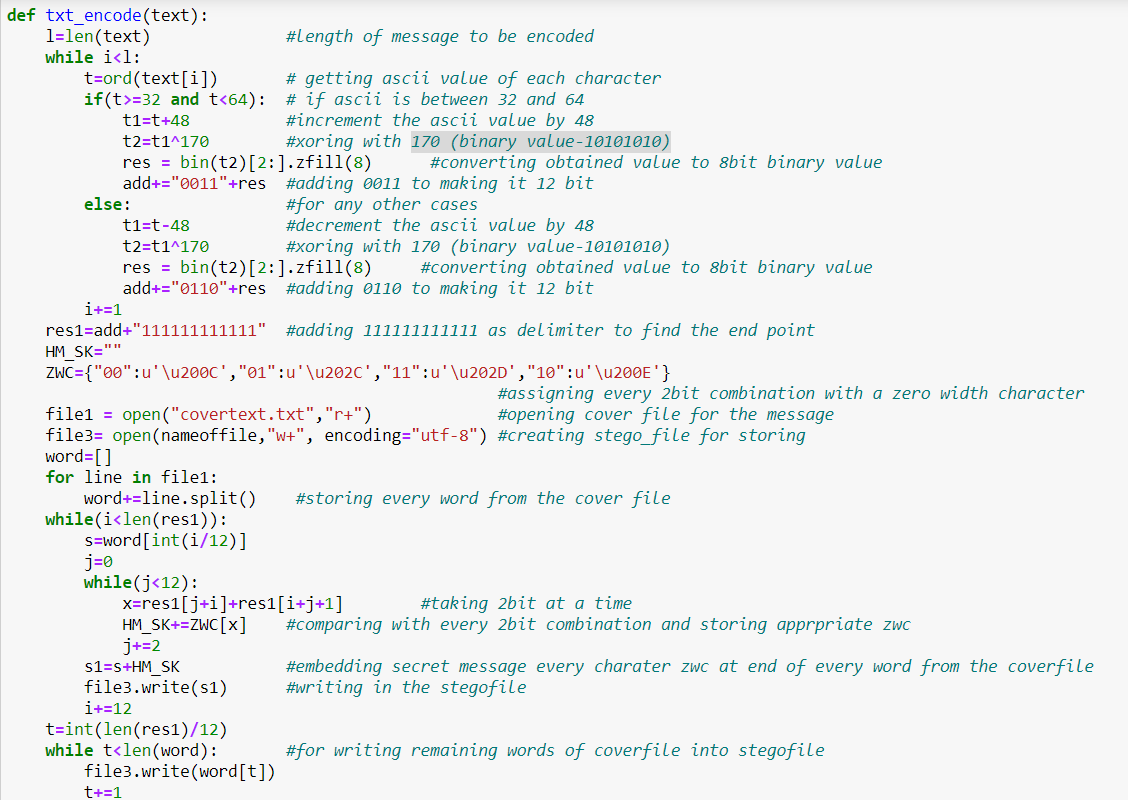
* We get its ascii value and it is incremented or decremented based on if ascii value between 32 and 64 , it is incremented by 48(ascii value for 0) else it is decremented by 48
* Then xor the the obtained value with 170(binary equivalent-10101010)
* Convert the obtained number from first two step to its binary equivalent then add "0011" if it earlier belonged to ascii value between 32 and 64 else add "0110" making it 12 bit for each character.

With the final binary equivalent we also 111111111111 as delimiter to find the end of message

Now from 12 bit representing each character every 2 bit is replaced with equivalent ZWCs according to the table. Each character is hidden after a word in the cover text.



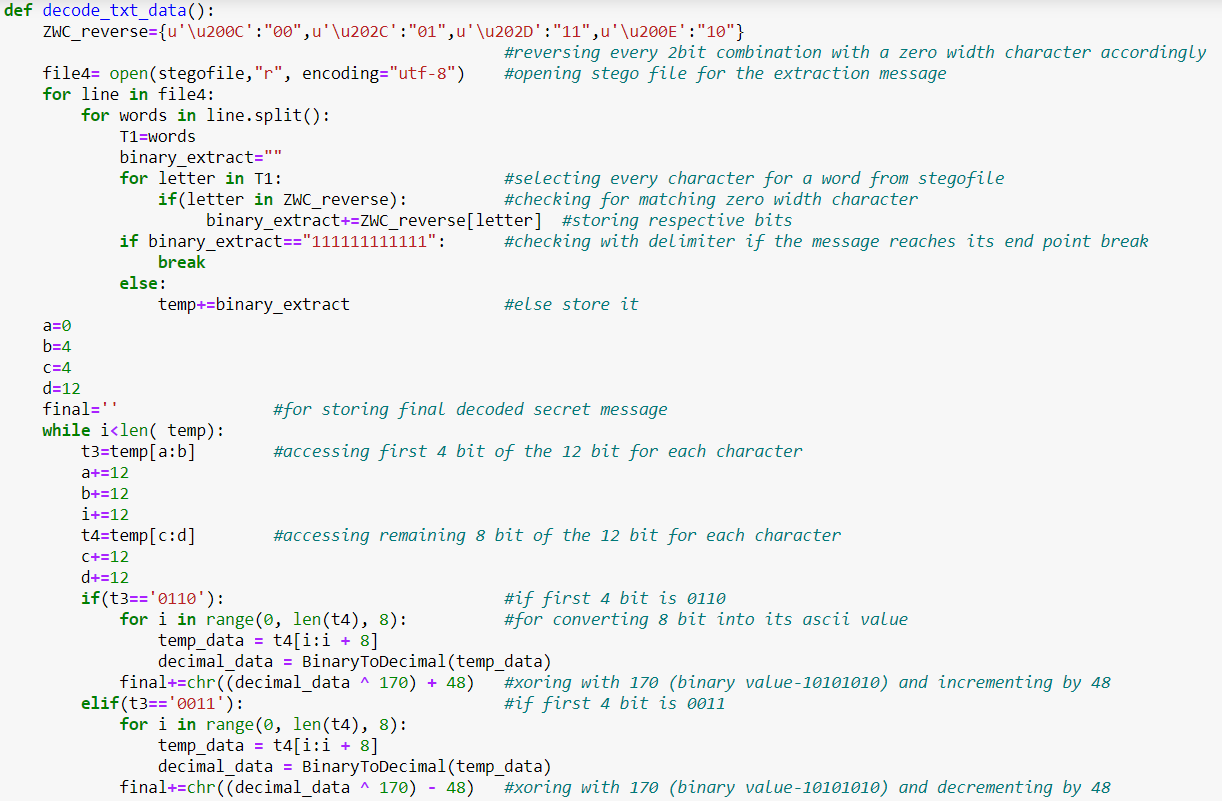
Zero Width Character Table



Text Steganography Encoding Algorithm

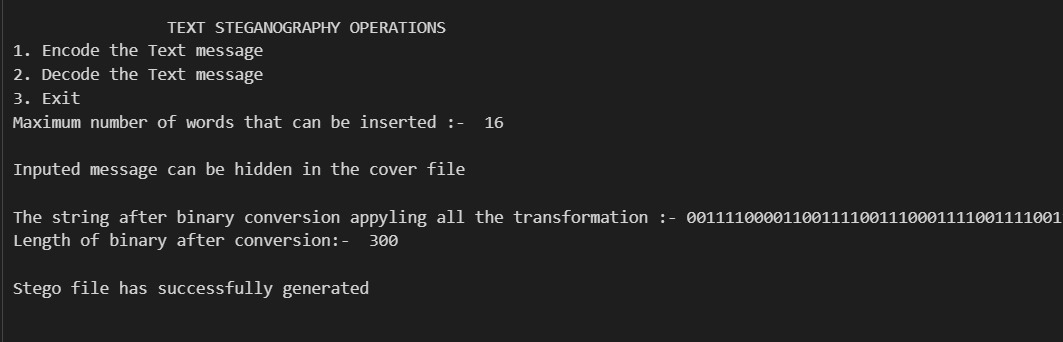
DECODE:-

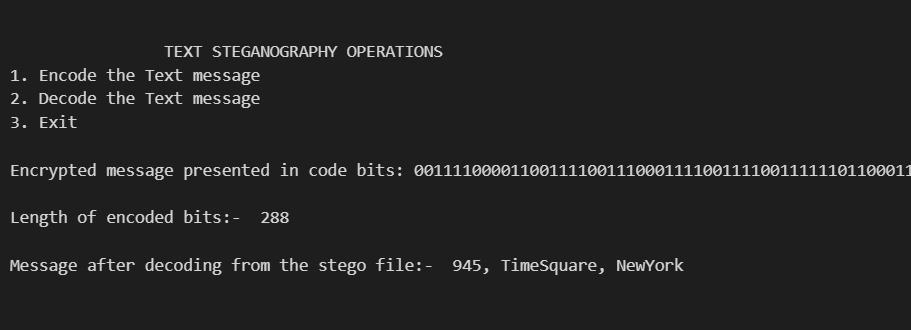
After receiving a stegofile , the extraction algorithm discovers the contractual 2-bit of each ZWCs , every 12 bit from end of the word in the stego file and then the binary equivalent is completely extracted and the delimiter discussed above helps us in getting to the end point. Now we divide the 12 bit into two parts first 4 bit and another 8bit on which we do the xor operation with 170(binary value 10101010). Now according to the first 4bit if its "0110" we increment it by 48 else we decrement by 48. At last we convert the ascii value into its equivalent character to get the final hidden message from the stego file.

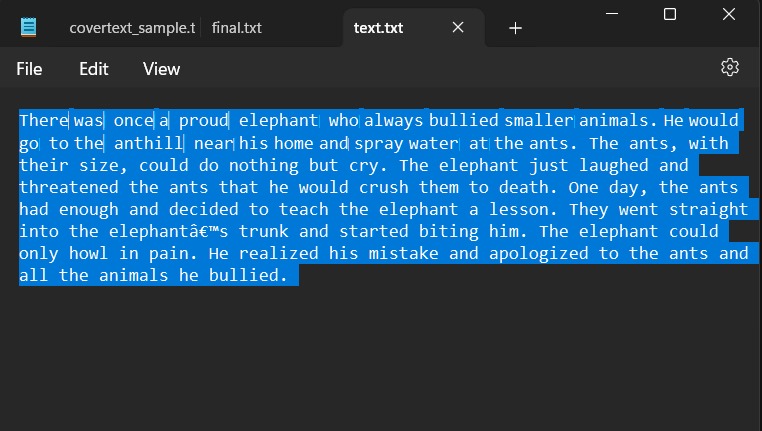


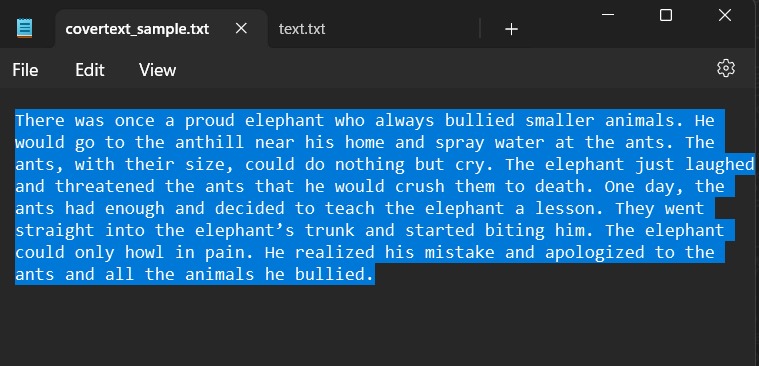
Text Steganography Decoding Algorithm

Output:





Original and Steg text



**Image Steganography**

ENCODE:-

Using **Modified LSB Algorithm** where we overwrite the LSB bit of actual image with the bit of text

message character. At the end of the text message we **push a delimiter to the message string** as a checkpoint useful in the decoding function. We encode data in order of Red, then Green and then Blue pixel for the entire message.

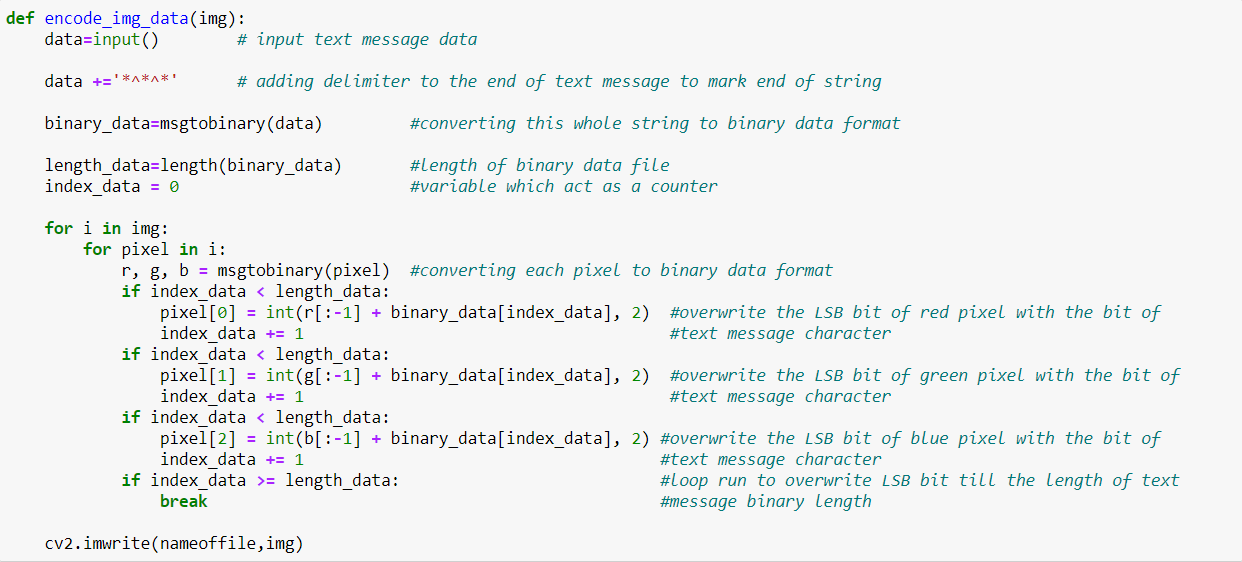
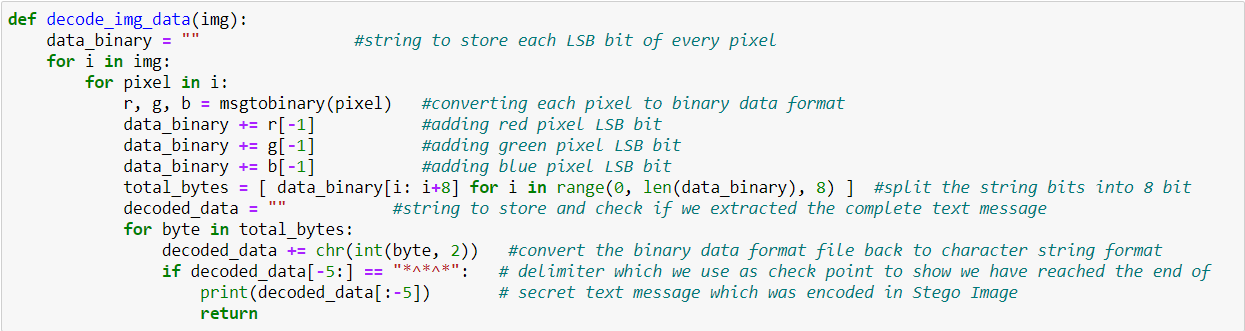


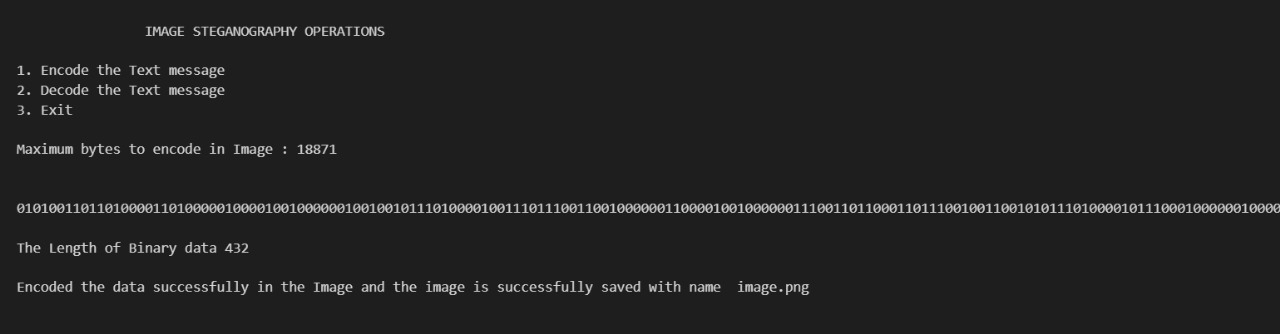
Image Steganography Encoding Algorithm

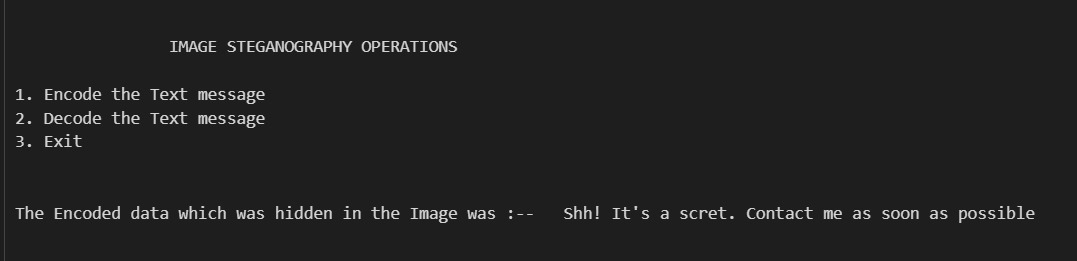
DECODE:-

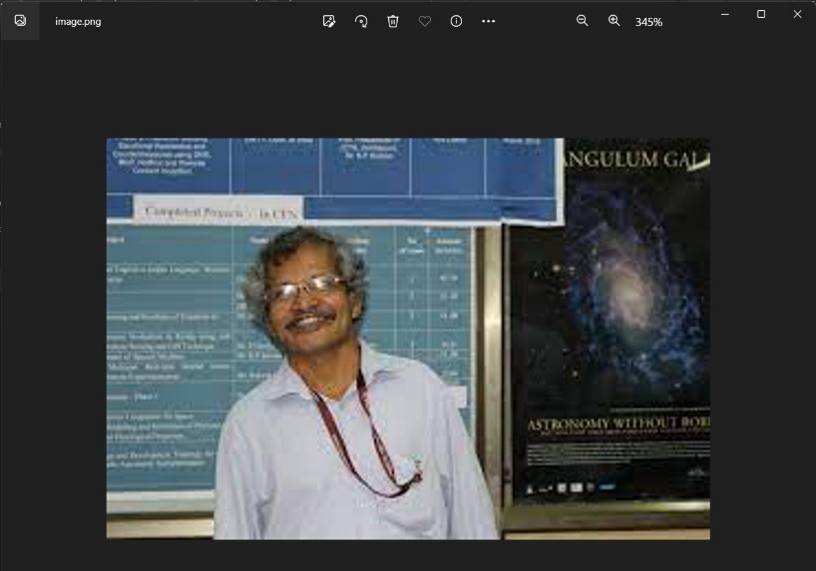
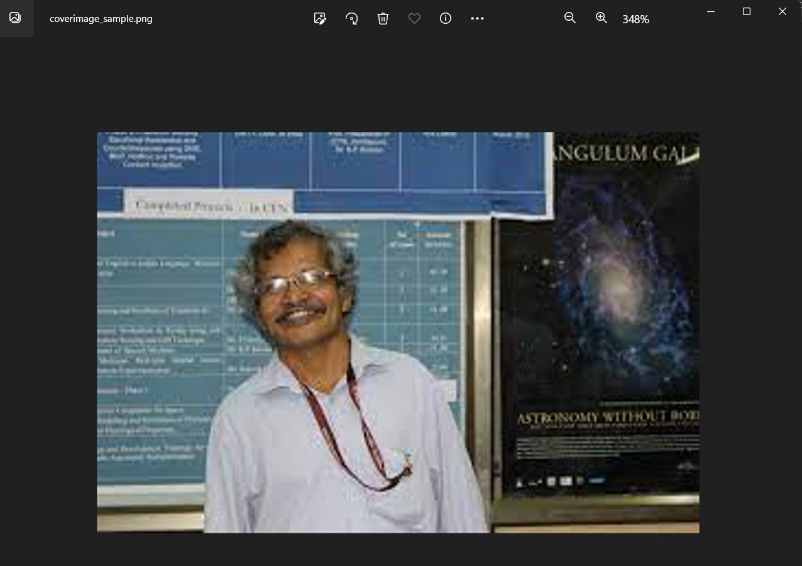
In the decode part, we take **all the LSB bits of each pixel** until we get a checkpoint/delimiter and then **we split them by 8 bits** and convert them to characters data type and **print the string (i.e., the secret text message) without delimiter.**

 Image Steganography Decoding Algorithm

Output:





The cover image and steg image: