

A MINI PROJECT REPORT

on

STEGANOGRAPHY TECHNIQUES USING PYTHON

Submitted to Jawaharlal Nehru Technological University for the partial

Fulfillment of the Requirement for the Award of the Degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

P. SAI SRINIVAS - 19Q91A05L7

V. MRINALINI - 19Q91A05N3

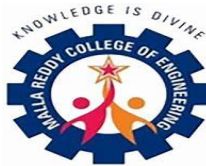
K. NAGA SATHYA VATHI - 19Q91A05K2

K. SHIVA KUMAR - 19Q91A05K0

Under the guidance of

Dr. G. RADHA DEVI

ASSISTANT PROFESSOR



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MALLA REDDY COLLEGE OF ENGINEERING

(Approved by AICTE-Permanently Affiliated to JNTU-Hyderabad)

Accredited by NBA & NAAC, Recognized section 2(f) & 12(B) of UGC New
Delhi ISO 9001:2015 certified Institution

Maisammaguda, Dhulapally (Post via Kompally), Secunderabad- 500100

2022 – 2023

MALLA REDDY COLLEGE OF ENGINEERING

(Approved by AICTE-Permanently Affiliated to JNTU-Hyderabad)

Accredited by NBA & NAAC, Recognized section 2(f) & 12(B) of UGC New Delhi

ISO 9001:2015 Certified Institution

Maisammaguda, Dhulapally (Post via Kompally), Secunderabad- 500100

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

This is to certify that the Mini Project report on “**STEGANOGRAPHY TECHNIQUES USING PYTHON**” is successfully done by the following students of the Department of Computer Science & Engineering of our college in partial fulfillment of the requirement for the award of a B.Tech degree in the year 2022-2023. The results embodied in this report have not been submitted to any other University for the award of any diploma or degree.

P. SAI SRINIVAS: **19Q91A05L7**

V. MRINALINI: **19Q91A05N3**

K. NAGA SATHYAVATHI: **19Q91A05K2**

K. SHIVA KUMAR: **19Q91A05K0**

INTERNAL GUIDE

Dr. G. Radha Devi
Asst. Professor

H O D

Dr. G. Radha Devi
Asst. Professor

PRINCIPAL

Dr. M. Sreedhar Reddy
Professor

Submitted for the viva voice examination held on: _____

Internal Examiner

External Examiner

DECLARATION

We, P. Sai Srinivas, V. Mrinalini, K. Naga Sathya Vathi, and K. Shiva Kumar with Regd. no. 19Q91A05L7, 19Q91A05N3, 19Q91A05K2, and 19Q91A05K0 are hereby declaring that the mini project report entitled “**Steganography Techniques using Python**” done by us under the guidance of **Dr. G. Radha Devi** Assistant Professor, Department of CSE is submitted in the partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING**.

The Results embedded in this project report have not been submitted to any other University or institute for the award of any degree or diploma.

Signature of the Candidate

P. Sai Srinivas **19Q91A05L7**

V. Mrinalini **19Q91A05N3**

K. Naga Sathya Vathi **19Q91A05K2**

K. Shiva Kumar **19Q91A05K0**

DATE:

PLACE: Maisammaguda

ACKNOWLEDGEMENT

First and foremost, we would like to express our immense gratitude towards our institution Malla Reddy College of Engineering, which helped us to attain profound technical skills in the field of Computer Science & Engineering, there by fulfilling our most cherished goal.

We are pleased to thank **Sri Ch. Malla Reddy**, our Founder, Chairman **MRGI**, **Sri Ch. Mahender Reddy**, Secretary, **MRGI** for providing this opportunity and support throughout the course.

It gives us immense pleasure to acknowledge the perennial inspiration of **Dr. M. Sreedhar Reddy** our beloved principal for his kind co-operation and encouragement in bringing out this task.

We would like to thank **Dr. T. V. Reddy** our vice principal, **Dr. G. Radha Devi** HOD, CSE Department for their inspiration adroit guidance and constructive criticism for successful completion of our degree.

We would like to thank **Dr. G. Radha Devi** Assistant Professor our internal guide, for her valuable suggestions and guidance during the exhibition and completion of this project.

Finally, we avail this opportunity to express our deep gratitude to all staff who have contribute their valuable assistance and support making our project success.

P. Sai Srinivas **19Q91A05L7**

V. Mrinalini **19Q91A05N3**

K. Naga Sathya Vathi **19Q91A05K2**

K. Shiva Kumar **19Q91A05K0**

ABSTRACT

This report describes the project development of STEGANOGRAPHY TECHNIQUES USING PYTHON that was developed to encrypt and decrypt information into various types of cover files. Steganography is a technique to hide secret or sensitive information behind a non-secret, ordinary file or message. The aim of the project is to develop a steganography tool that hides information or message behind different cover files such as text, image, audio, video files. Prior to these advanced cryptography methods, steganography was well known for secure communication. There are numerous strategies to conceal data into the cover file; this task utilizes various algorithms. This strategy implants the cycle of the message into the least significant bit of the cover file. LSB hash strategy might change the properties of the original file, yet there is no visual change in the file. This report additionally presents a strategy to Decrypt the Stego file and extract the restricted information from the stego file.

Keywords: Steganography, LSB- Least Significant Bit, JPEG, BMP, DCT, Encryption, Decryption

TABLE OF CONTENTS

CERTIFICATE	II
DECLARATION	III
ACKNOWLEDGEMENT	IV
ABSTRACT	V
TABLE OF CONTENTS	VI
LIST OF FIGURES	VIII
LIST OF SCREEN SHOTS	IX
LIST OF ABBREVIATIONS	X
CHAPTER 1: INTRODUCTION	
1.1 Introduction	1
1.2 Steganography Techniques	2
CHAPTER 2: LITERATURE SURVEY	
2.1 Literature survey	5
CHAPTER 3: SYSTEM ANALYSIS	
3.1 Existing system	6
3.2 Drawbacks	6
3.3 Proposed system	6
3.4 Advantages	7
3.5 System Requirements	8
CHAPTER 4: SYSTEM DESIGN	
4.1 System architecture	9
4.2 UML Diagrams	12
CHAPTER 5: SYSTEM IMPLEMENTATION	
5.1 Python (Programming language)	14
5.2 Open CV	20
5.3 Numpy	25
5.4 Source code	28

CHAPTER 6: RESULTS

6.1 Screenshots	53
-----------------	----

CHAPTER 7: CONCLUSION

7.1 Conclusion	57
----------------	----

CHAPTER 8: FUTURE ENHANCEMENTS

8.1 Future Enhancements	58
-------------------------	----

REFERENCES	59
-------------------	-----------

LIST OF FIGURES

Figure No	Name of the Figure	Page No.
1.	Explaining how steganography works	1
2.	Types of Steganography	2
3.	Working of Image Steganography using LSB substitution Method	7
4.	Flowchart showing the working of Text Steganography	9
5.	Flowchart showing the working of Image Steganography	9
6.	Flowchart showing the working of Audio Steganography	10
7.	Flowchart showing the working of Video Steganography	11
8.	Flowchart showing the working of Network Steganography	11
9.	Image showing the Use Case Diagram	12
10.	Image showing the Class Diagram	12
11.	Image showing the Sequence Diagram	13
12.	Image showing the pixels in an image	21
13.	Image showing how colors are created With RGB	22
14.	Image showing the Nd-array	26

LIST OF SCREEN SHOTS

Figure No	Name of Screenshot	Page No.
1.	Screenshot of python path in command Prompt Interactive mode	16
2.	Screenshot of python console in cmd	16
3.	Screenshot of python program in cmd	17
4.	Screenshot of python program written in Notepad	17
5.	Screenshot of file saving with .py extension	18
6.	Screenshot of executing python program By Interactive mode	18
7.	Screenshot of Python IDLE	19
8.	Screenshot Image Steganography Encoding	53
9.	Screenshot Image Steganography Decoding	53
10.	Screenshot Text Steganography Encoding	54
11.	Screenshot Text Steganography Decoding	54
12.	Screenshot Audio Steganography Encoding	55
13.	Screenshot Audio Steganography Decoding	55
14.	Screenshot Video Steganography Encoding	56
15.	Screenshot Video Steganography Decoding	56

LIST OF ABBREVIATIONS

S. No	Short Form	Full Form
1.	JPEG	Joint Photographic Experts Group
2.	LSB	Least Significant Bit
3.	TCP	Transmission Control Protocol
4.	UDP	User Datagram Protocol
5.	ICMP	Internet Control Message Protocol
6.	IP	Internet Protocol
7.	RGB	Red, Green, Blue
8.	CPU	Central Processing Unit
9.	DCT	Discrete Cosine Transform
10.	BMP	Bitmap
11.	GIF	Graphics Interchange Format
12.	MSB	Most Significant Bit

1.1 INTRODUCTION

Steganography is the art and science of hiding secret data within an ordinary, non-secret, file or message in order to avoid detection; the secret data is only decrypted at its destination. The word steganography is derived from the Greek words steganos which means something that is hidden and the Greek root graph which means to write. When we put these words together we get somewhat close to “secret writing”, or “hidden writing”. The historical backdrop of steganography traces all the way back to 1500 when Johannes Trithemius utilized the term in his book Steganographia. In modern digital steganography, data is first encrypted or obfuscated in some other way and then inserted, using a special algorithm, into data that is part of a particular file format such as a JPEG image, audio or video file. The secret message can be embedded into ordinary data files in many different ways. One technique is to hide data in bits that represent the same colour pixels repeated in a row in an image file. By applying the encrypted data to this redundant data in some inconspicuous way, the result will be an image file that appears identical to the original image but that has "noise" patterns of regular, unencrypted data.

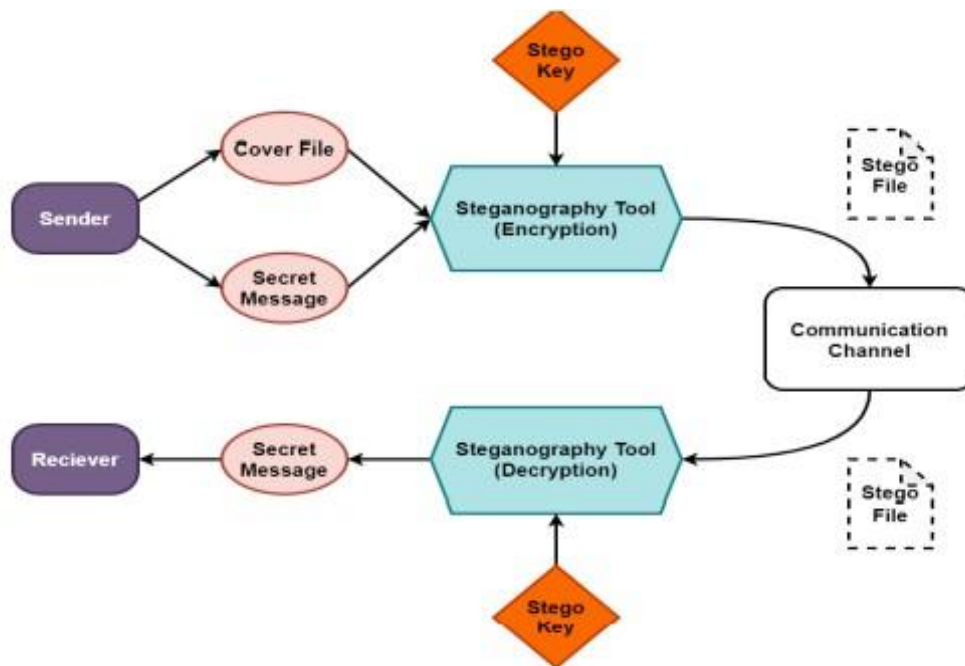


Figure 1.1.1: Explaining how steganography works

1.2 STEGANOGRAPHY TECHNIQUES

Depending on the nature of the cover object (actual object in which secret data is embedded), steganography can be divided into five types:

- i. Text Steganography
- ii. Image Steganography
- iii. Audio Steganography
- iv. Video Steganography
- v. Network Steganography

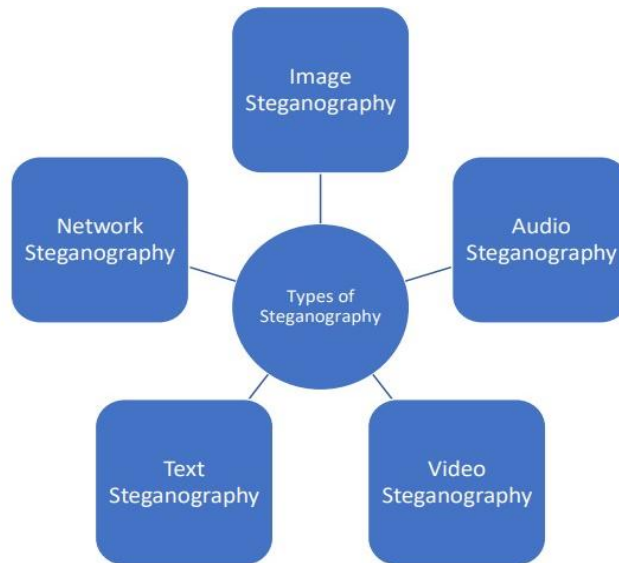


Figure 1.2.1: Types of Steganography

1.2.1 TEXT STEGANOGRAPHY

Text Steganography is hiding information inside the text files. It involves things like changing the format of existing text, changing words within a text, generating random character sequences or using context-free grammars to generate readable texts. A few ways this is done are:

- a. Hiding information in plain text by adding white space and tabs to the end of the lines of documents
- b. Using a widely available cover source like a book or newspaper and using a code comprising of a combination of numbers, letters, or line number. This way the information inside the cover source will not reveal the hidden message and the only way to decode is to gain the key.
- c. The use of background color and font is another widely used technique for steganography. It is widely used in Microsoft Word documents.

1.2.2 IMAGE STEGANOGRAPHY

Hiding the data by taking the cover object as the image is known as image steganography. In digital steganography, images are widely used cover source because there are a huge number of bits present in the digital representation of an image. A few methods used are:

- a. Least significant bit insertion.
- b. Masking and filtering.
- c. Redundant Pattern Encoding.
- d. Encrypt and Scatter.
- e. Algorithms and transformations.
- f. Least significant bit insertion.

1.2.3 AUDIO STEGANOGRAPHY

In audio steganography, the secret message is embedded into an audio signal which alters the binary sequence of the corresponding audio file. Hiding secret messages in digital sound is a much more difficult process when compared to others, such as Image Steganography. A few methods used are:

- a. LSB coding.
- b. Parity coding.
- c. Phase coding.
- d. Spread spectrum.
- e. Echo hiding

1.2.4 VIDEO STEGANOGRAPHY

In Video Steganography you can hide kind of data into digital video format. The advantage of this type is a large amount of data can be hidden inside and the fact that it is a moving stream of images and sounds. You can think of this as the combination of Image Steganography and Audio Steganography. A few methods used are:

- a. Least Significant Bit Insertion.
- b. Real-time Video Steganography

1.2.5 NETWORK STEGANOGRAPHY

It is the strategy of inserting data inside network control protocols utilized in information transmission such TCP, UDP, and ICMP and so forth. You can utilize steganography in some covert channels that you can find in the OSI model. For Example, you can conceal data in the header of a TCP/IP packet in certain fields that are either optional.

2.1 LITERATURE SURVEY

This project addresses the security problem of transmitting the data over internet network, the main idea which comes to our mind, when we start asking that how can we send a message secretly to the destination? The science of steganography answers this question. Using steganography, information can be hidden in carriers such as images, audio files, text files, videos and data transmissions. In this document, I proposed some methods and algorithms of a complete steganography system to hide a digital text of a secret message into various file types as carrier.

Our project scope is developed for hiding information in any type of file to ensure the safety of exchange the data between different parties and provide better security during message transmission. The scope of the project is implementation of steganography tools for hiding information includes any type of information file and cover files and the path where the user wants to save image and extruded file. We will use LSB technique; the proposed approach is to use the suitable algorithm for embedding the data in a cover file; we will show a brief of this algorithm that we used to hiding data.

3.1 EXISTING SYSTEM

In computing, the most significant bit (msb, also called the high-order bit) is the bit position in a binary number having the greatest value. The msb is sometimes referred to as the left-most bit due to the convention in positional notation of writing more significant digits further to the left. The MSB can also correspond to the sign of a signed binary number in one or two's complement notation. "1" meaning negative and "0" meaning positive. It is common to assign each bit a position number, ranging from zero to N-1, where N is the number of bits in the binary representation used. Normally, this is simply the exponent for the corresponding bit weight in base-2 (such as in $2^{31..20}$). Although a few CPU manufacturers assign bit numbers the opposite way (which is not the same as different endianness), the msb unambiguously remain the most significant bit. This may be one of the reasons why the term msb is often used instead of a bit number, although the primary reason is probably that different number representations use different numbers of bits. By extension, the most significant bits (plural) are the bits closest to, and including, the MSB.

3.2 DRAWBACKS

- There was a significant change in the Stego image using this technique

3.3 PROPOSED SYSTEM

This project uses the LSB substitution method. LSB stands for **Least Significant Bit**. The idea behind LSB embedding is that if we change the last bit value of a pixel, there won't be much visible change in the color. For example, 0 is black. Changing the value to 1 won't make much of a difference since it is still black, just a lighter shade.

LSB embedding is the most common technique to embed message bits DCT coefficients. This method has also been used in the spatial domain where the least significant bit value of a pixel is changed to insert a zero or a one. A simple example would be to associate an even coefficient with a zero bit and an odd one with a one-bit value. In order to embed a message bit in a pixel or a DCT coefficient, the sender increases or decreases the value of the coefficient/pixel to embed a zero or a one. The receiver then extracts the hidden message bits by reading the coefficients in the same sequence and decoding them in accordance with the encoding technique performed on it.

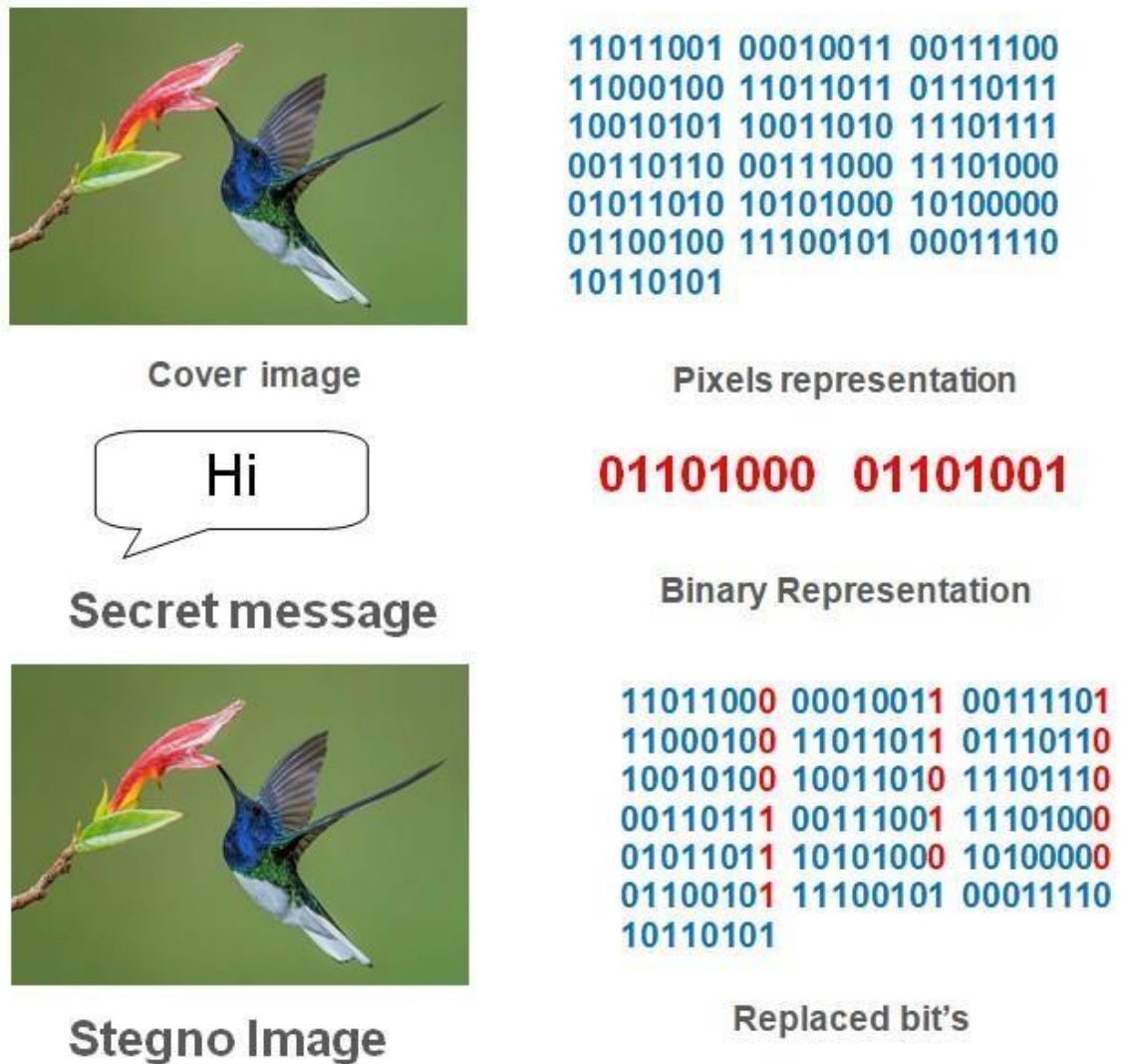


Figure 3.1.1: Working of Image Steganography using LSB Substitution Method

3.4 ADVANTAGES

- Good embedding capacity.
- Change in stego image is usually visually undetectable to the human eye.
- Greater capacity for the spatial domain embedding with almost one bit per pixel for each color component.

3.5 SYSTEM REQUIREMENTS

Processor	x86 64-bit CPU (Intel / AMD Architecture)
Operating System	Microsoft Windows 7 to 10 Mac OS X 10.11 Or higher, 64-bit Linux: RHEL 6/7, 64-bit
Memory	4GB RAM
Screen Resolution	Monitor with screen resolution minimum 1024 x 768
Hard disk Space	Minimum 5GB free disk space

4.1 SYSTEM ARCHITECTURE

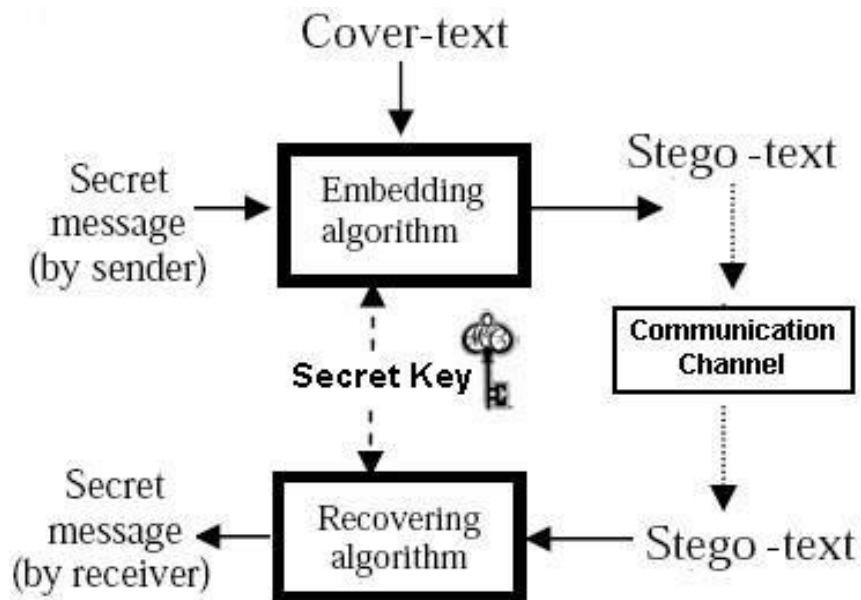


Figure 4.1.1: Flowchart showing the working of Text Steganography

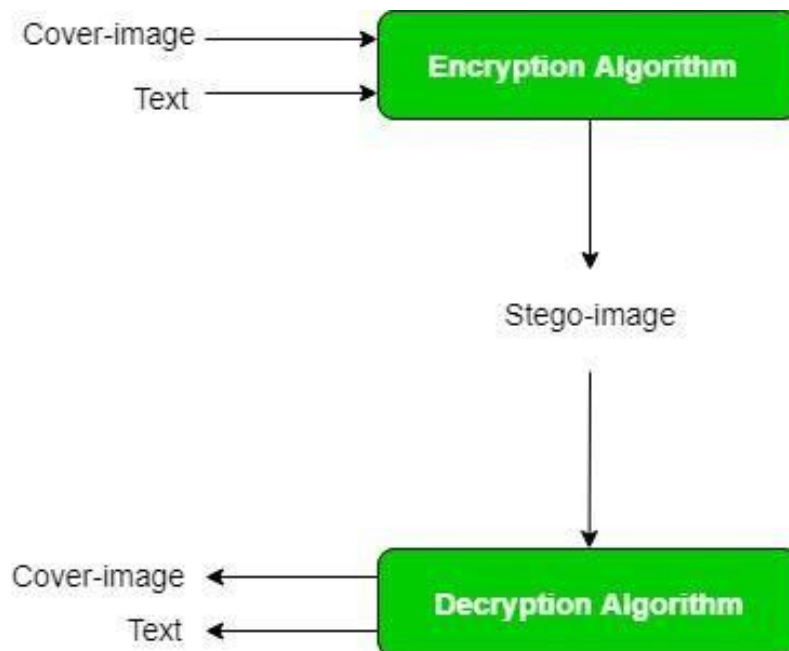


Figure 4.1.2: Flowchart showing the working of Image Steganography

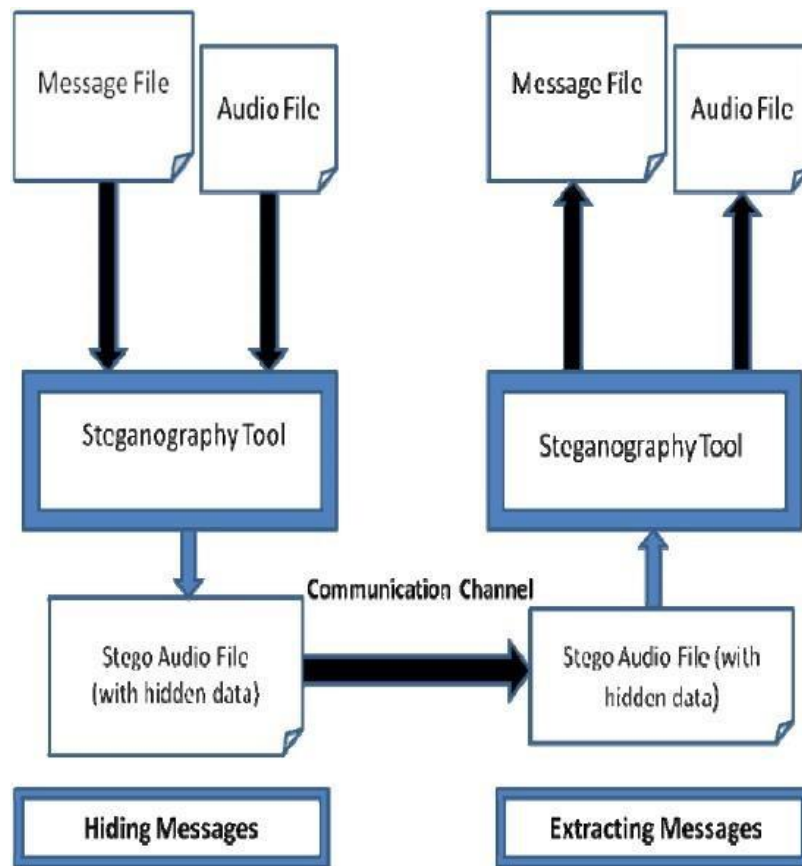


Figure 4.1.3: Flowchart showing the working of Audio Steganography

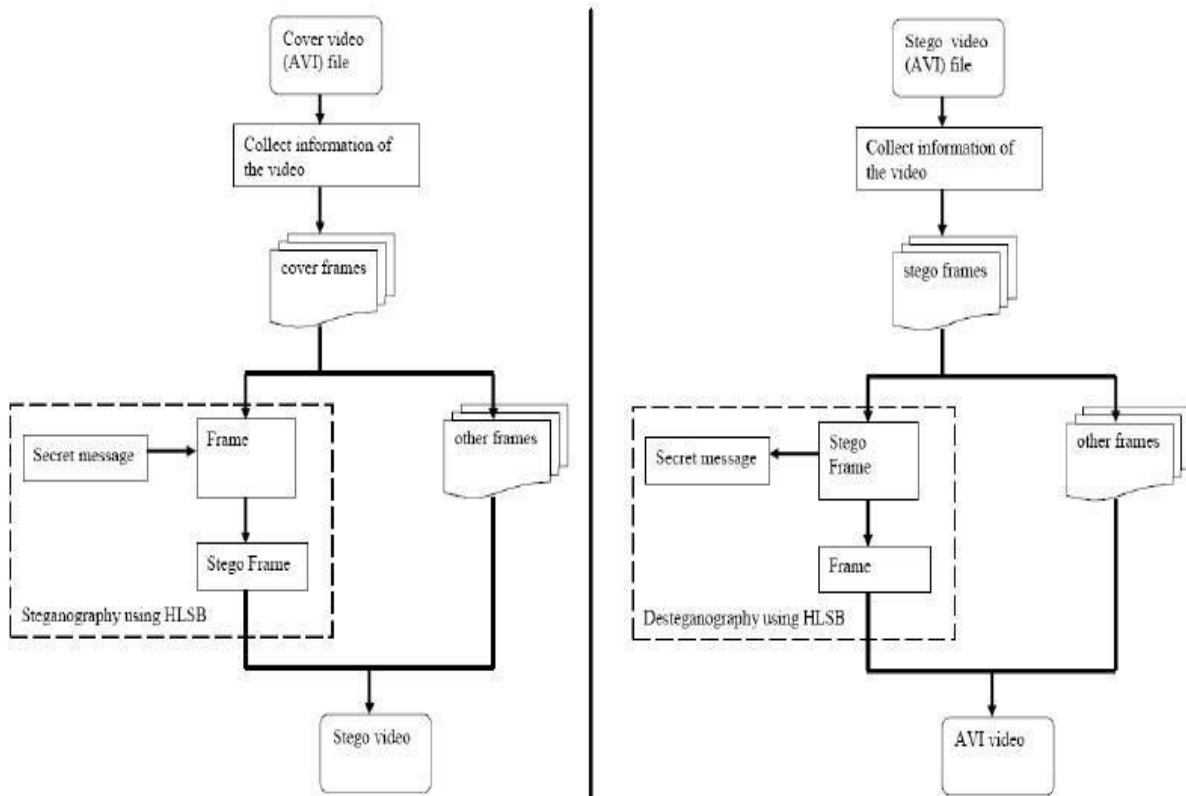


Figure 4.1.4: Flowchart showing the working of Video Steganography

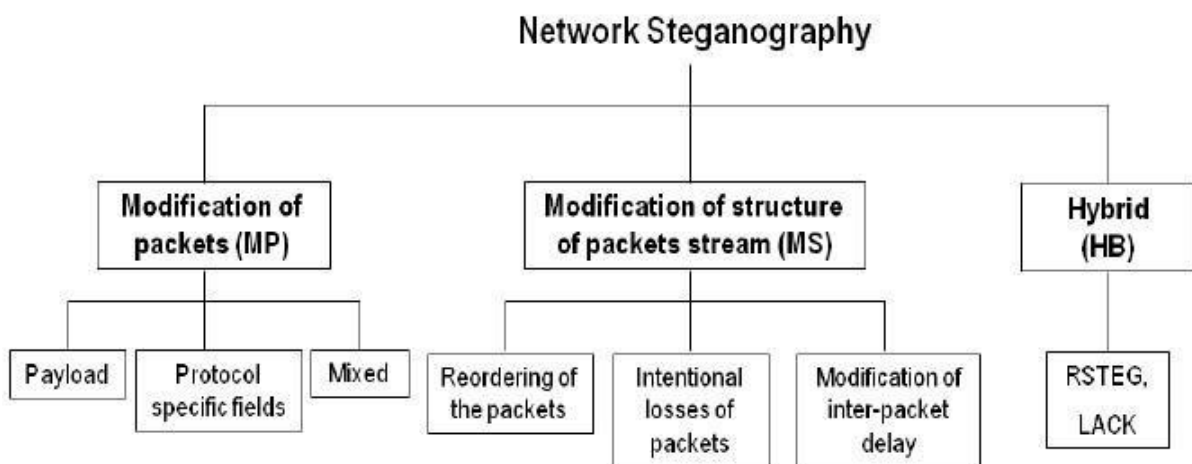


Figure 4.1.5: Flowchart showing the working of Network Steganography

4.2 UML DIAGRAMS

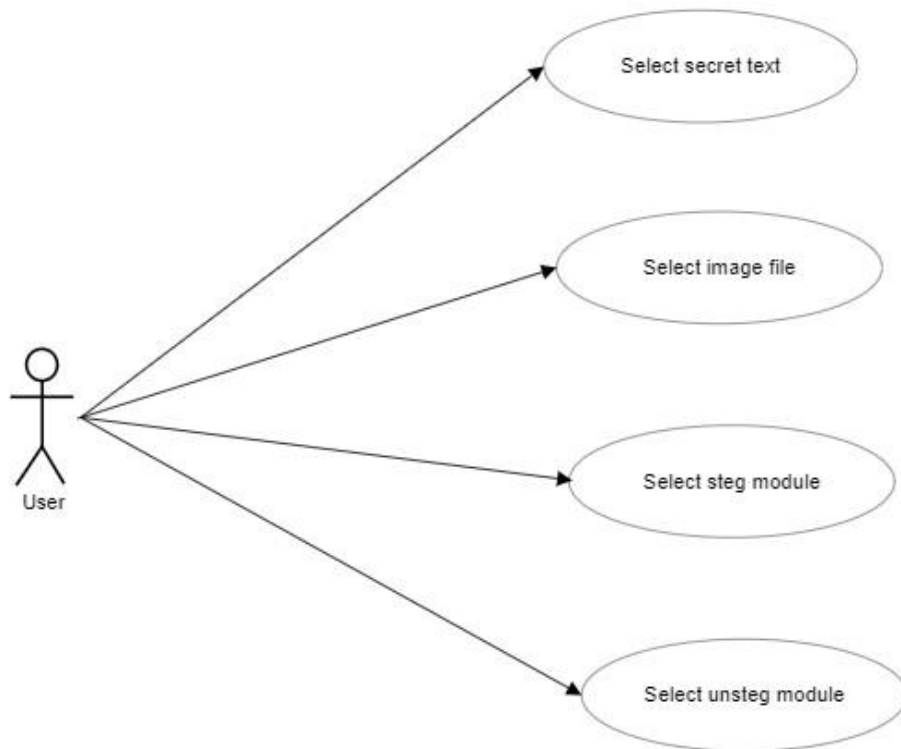


Figure 4.2.1: Use Case diagram

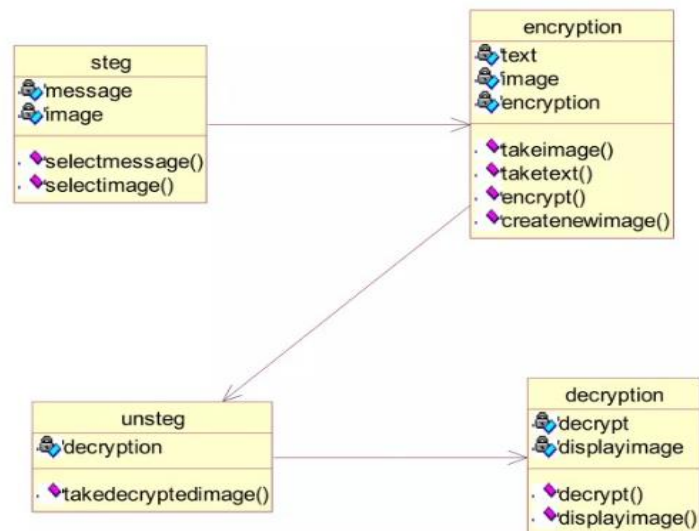


Figure 4.2.2: Class diagram

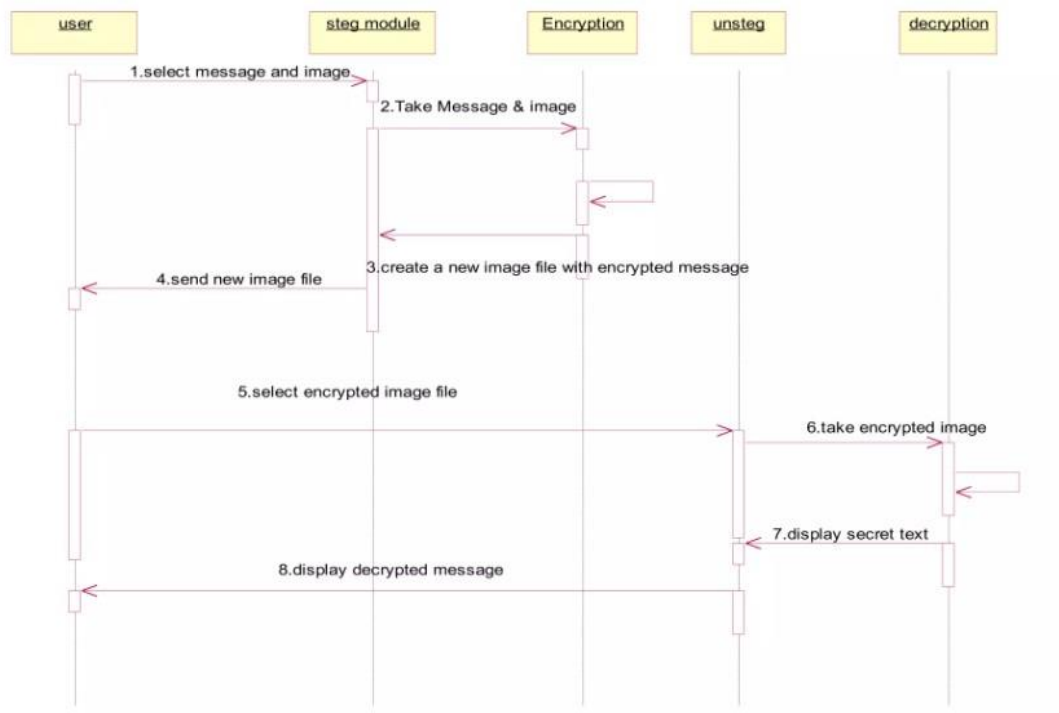


Figure 4.2.3: Sequence diagram

5.1 PYTHON

Python Is an Object-Oriented, High Level Language, Interpreted, Dynamic And Multipurpose Programming Language.

Python Is Easy To Learn Yet Powerful And Versatile Scripting Language Which Makes It Attractive For Application Development.

Python's Syntax And Dynamic Typing With Its Interpreted Nature, Make It An Ideal Language For Scripting And Rapid Application Development In Many Areas.

Python Supports Multiple Programming Pattern, Including Object Oriented Programming, Imperative And Functional Programming Or Procedural Styles.

Python Is Not Intended To Work On Special Area Such As Web Programming. That Is Why It Is Known As Multipurpose Because It Can Be Used With Web, Enterprise, 3d Cad Etc.

We Don't Need To Use Data Types To Declare Variable Because It Is Dynamically Typed So We Can Write A=10 To Declare An Integer Value In A Variable.

Python Makes The Development And Debugging Fast Because There Is No Compilation Step Included In Python Development And Edit-Test-Debug Cycle Is Very Fast.

Python Features

- 1) Easy to use:
- 2) Expressive Language
- 3) Interpreted Language
- 4) Cross-Platform Language
- 5) Free And Open Source
- 6) Object-Oriented Language
- 7) Extensible
- 8) Large Standard Library
- 9) GUI Programming
- 10) Integrated

Python History

- Python Laid its foundation in the late 1980s.
- The implementation of Python was started in December 1989 by Guido Van Rossum at Cwi, Netherland.
- Python is influenced by programming languages like:
Abc Language, Modula-3

Python Applications

Python as a whole can be used in any sphere of development.

Let us see what are the major regions where python proves to be Handy.

- 1) Console Based Application
- 2) Audio or Video based applications
- 3) 3d Card Applications
- 4) Web Applications
- 5) Enterprise Applications
- 6) Applications for Images

Python Example

Python Code Is Simple And Easy To Run. Here Is A Simple Python Code That Will Print "Welcome To Python".

A Simple Python Example Is Given Below.

1. >>> A="Welcome To Python"
2. >>> Print A
3. Welcome To Python
4. >>>

Explanation

- Here We Are Using Idle To Write The Python Code. Detail Explanation To Run Code Is Given In Execute Python Section.
- A Variable Is Defined Named "A" Which Holds "Welcome To Python".
- "Print" Statement Is Used To Print The Content. Therefore "Print A" Statement Will Print The Content Of The Variable. Therefore, The Output "Welcome To Python" Is Produced.

How to Execute Python

There are different ways to execute in Python:

1) Interactive Mode

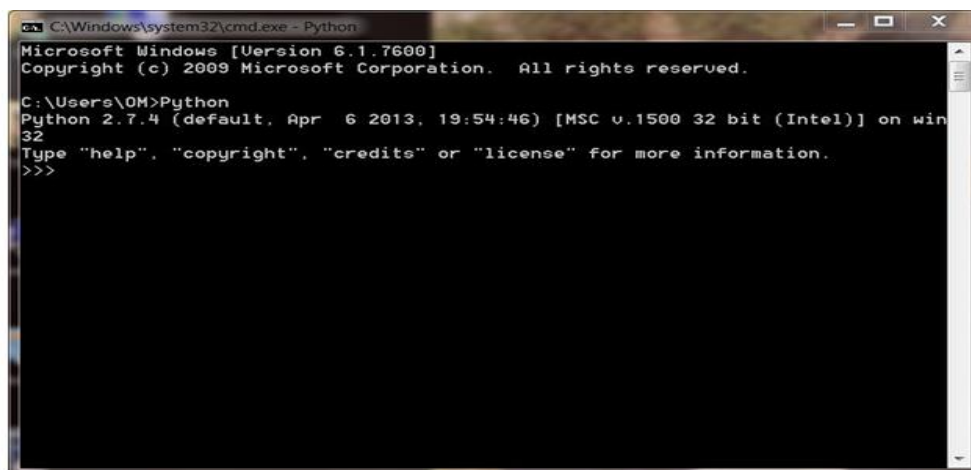
You Can Enter Python In The Command Prompt And Start Working With Python.



javatpoint.cpm

Figure 5.1.1: Command prompt

Press Enter Key And The Command Prompt Will Appear Like:



javatpoint.com

Figure 5.1.2: Python path in command prompt

Now You Can Execute Your Python Commands.

Eg

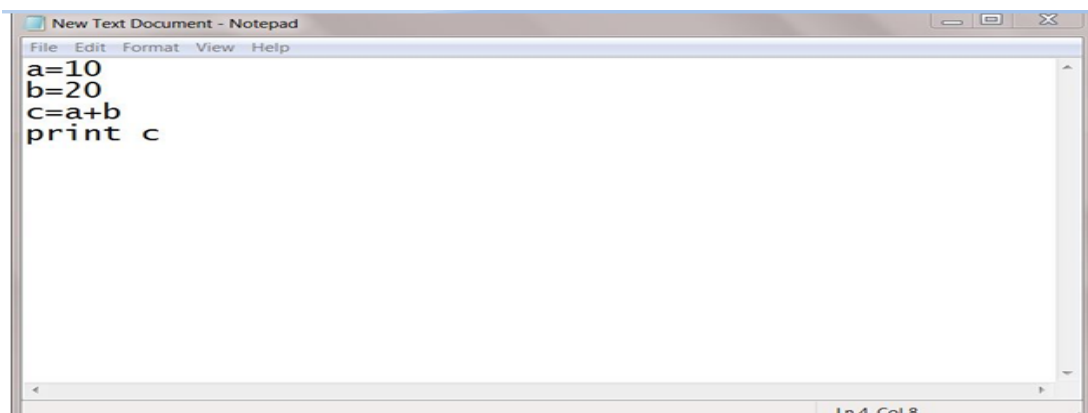


A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe - python". The window shows the output of running the Python interpreter. The text displayed is: "Microsoft Windows [Version 6.1.7600] Copyright (c) 2009 Microsoft Corporation. All rights reserved. C:\Users\OM>python Python 2.7.4 (default, Apr 6 2013, 19:54:46) [MSC v.1500 32 bit (Intel)] on win32 Type "help", "copyright", "credits" or "license" for more information. >>> a=10 >>> b=20 >>> c=a+b >>> print c 30 >>>". Below the screenshot, the text "iavatpoint.com" is visible.

Figure 5.1.3: Python program example

2) Script Mode

Using Script Mode , You Can Write Your Python Code In A Separate File Using Any Editor Of Your Operating System.



A screenshot of a Notepad window titled "New Text Document - Notepad". The window shows a Python script with the following code: "a=10", "b=20", "c=a+b", and "print c". The status bar at the bottom right indicates "Ln 4, Col 8". Below the screenshot, the text "iavatpoint.com" is visible.

Figure 5.1.4: Python program example in notepad

Save It By .Py Extension.

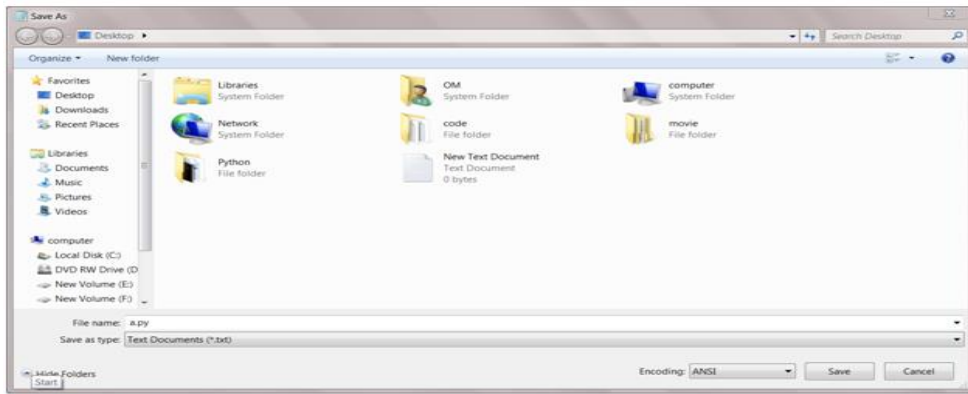


Figure 5.1.5: Python program saving

Now Open Command Prompt And Execute It By :

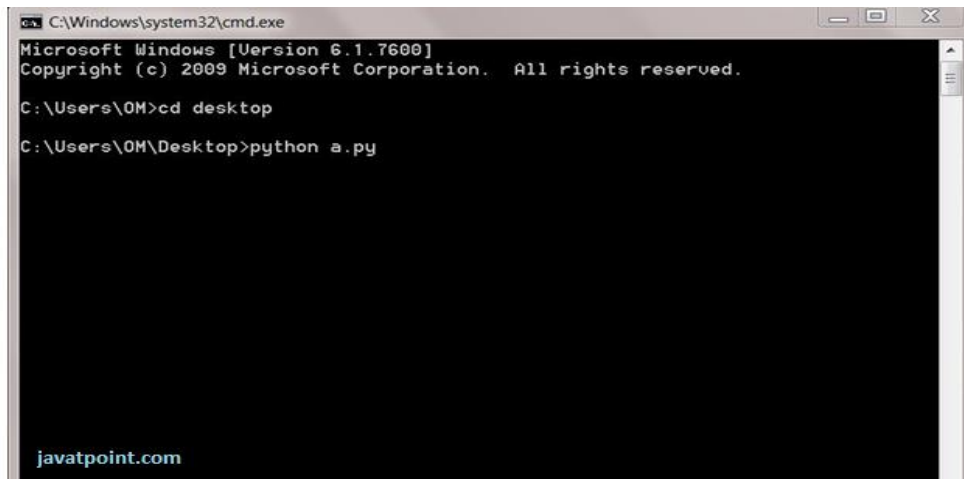


Figure 5.1.6: Python program example execution

3) Using IDE

You Can Execute Your Python Code Using A Graphical User Interface (Gui).

All You Need To Do Is:

Click On Start Button -> All Programs -> Python -> Idle(Python GUI)

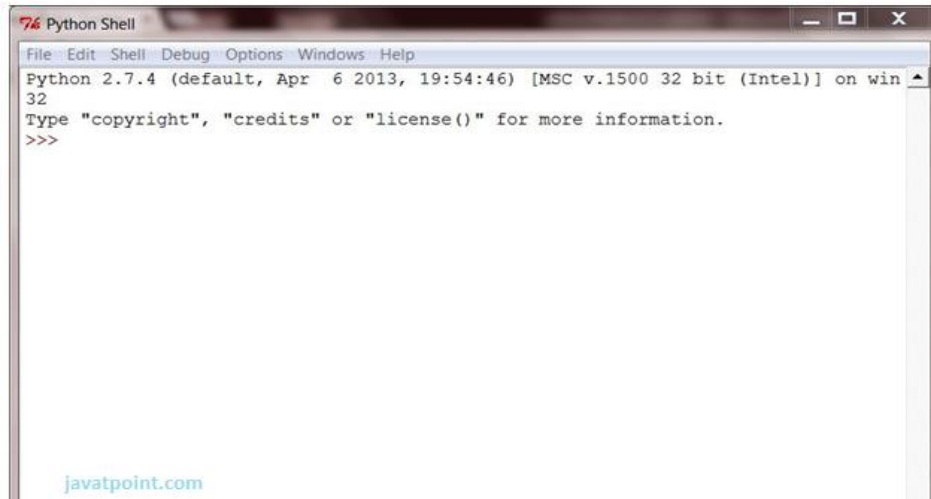


Figure 5.1.7: Python program example using IDLE

You Can Use Both Interactive As Well As Script Mode In Ide.

5.2 OPEN CV

Introduction To Computer Vision

Using Software To Parse The World's Visual Content Is As Big Of A Revolution In Computing As Mobile Was 10 Years Ago, And Will Provide A Major Edge For Developers And Businesses To Build Amazing Products.

Computer Vision Is The Process Of Using Machines To Understand And Analyze Imagery (Both Photos And Videos). While These Types Of Algorithms Have Been Around In Various Forms Since The 1960's, Recent Advances In Machine Learning, As Well As Leaps Forward In Data Storage, Computing Capabilities, And Cheap High-Quality Input Devices, Have Driven Major Improvements In How Well Our Software Can Explore This Kind Of Content.

What Is Computer Vision?

Computer Vision Is The Broad Parent Name For Any Computations Involving Visual Content – That Means Images, Videos, Icons, And Anything Else With Pixels Involved. But Within This Parent Idea, There Are A Few Specific Tasks That Are Core Building Blocks:

- In **Object Classification**, You Train A Model On A Dataset Of Specific Objects, And The Model Classifies New Objects As Belonging To One Or More Of Your Training Categories.
- For **Object Identification**, Your Model Will Recognize A Specific Instance Of An Object – For Example, Parsing Two Faces In An Image And Tagging One As Tom Cruise And One As Katie Holmes.

A Classical Application Of Computer Vision Is Handwriting Recognition For Digitizing Handwritten Content (We'll Explore More Use Cases Below). Outside Of Just Recognition, Other Methods Of Analysis Include:

- Video **Motion Analysis** Uses Computer Vision To Estimate The Velocity Of Objects In A Video, Or The Camera Itself.
- In **Image Segmentation**, Algorithms Partition Images Into Multiple Sets Of Views.
- **Scene Reconstruction** Creates A 3d Model Of A Scene Inputted Through Images Or Video (Check Out Selva).

- In **Image Restoration**, Noise Such As Blurring Is Removed From Photos Using Machine Learning Based Filters.

Any Other Application That Involves Understanding Pixels Through Software Can Safely Be Labeled As Computer Vision.

How Computer Vision Works

One Of The Major Open Questions In Both Neuroscience And Machine Learning Is: How Exactly Do Our Brains Work, And How Can We Approximate That With Our Own Algorithms? The Reality Is That There Are Very Few Working And Comprehensive Theories Of Brain Computation; So Despite The Fact That Neural Nets Are Supposed To “Mimic The Way The Brain Works,” Nobody Is Quite Sure If That’s Actually True. Jeff Hawkins Has An Entire Book On This Topic Called On Intelligence.

The Same Paradox Holds True For Computer Vision – Since We’re Not Decided On How The Brain And Eyes Process Images, It’s Difficult To Say How Well The Algorithms Used In Production Approximate Our Own Internal Mental Processes. For Example, Studies Have Shown That Some Functions That We Thought Happen In The Brain Of Frogs Actually Take Place In The Eyes. We’re A Far Cry From Amphibians, But Similar Uncertainty Exists In Human Cognition.

Machines Interpret Images Very Simply: As A Series Of Pixels, Each With Their Own Set Of Color Values. Consider The Simplified Image Below, And How Grayscale Values Are Converted Into A Simple Array Of Numbers:

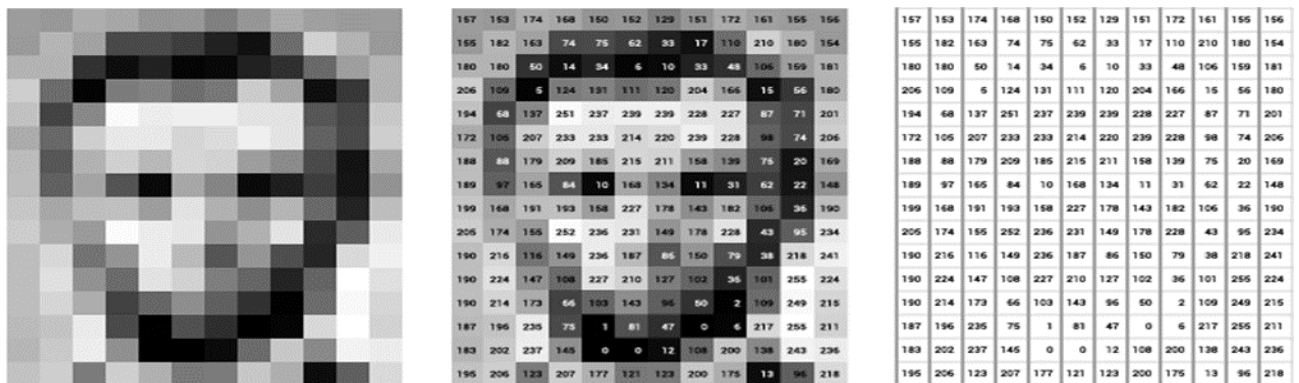


Figure 5.1.8: Pixels in an Image

Think Of An Image As A Giant Grid Of Different Squares, Or Pixels (This Image Is A Very Simplified Version Of What Looks Like Either Abraham Lincoln Or A Dementor). Each Pixel In An Image Can Be Represented By A Number, Usually From 0 – 255. The Series Of Numbers

On The Right Is What Software Sees When You Input An Image. For Our Image, There Are 12 Columns And 16 Rows, Which Means There Are 192 Input Values For This Image.

When We Start To Add In Color, Things Get More Complicated. Computers Usually Read Color As A Series Of 3 Values – Red, Green, And Blue (Rgb) – On That Same 0 – 255 Scale. Now, Each Pixel Actually Has 3 Values For The Computer To Store In Addition To Its Position. If We Were To Colorize President Lincoln (Or Harry Potter’s Worst Fear), That Would Lead To 12 X 16 X 3 Values, Or 576 Numbers.

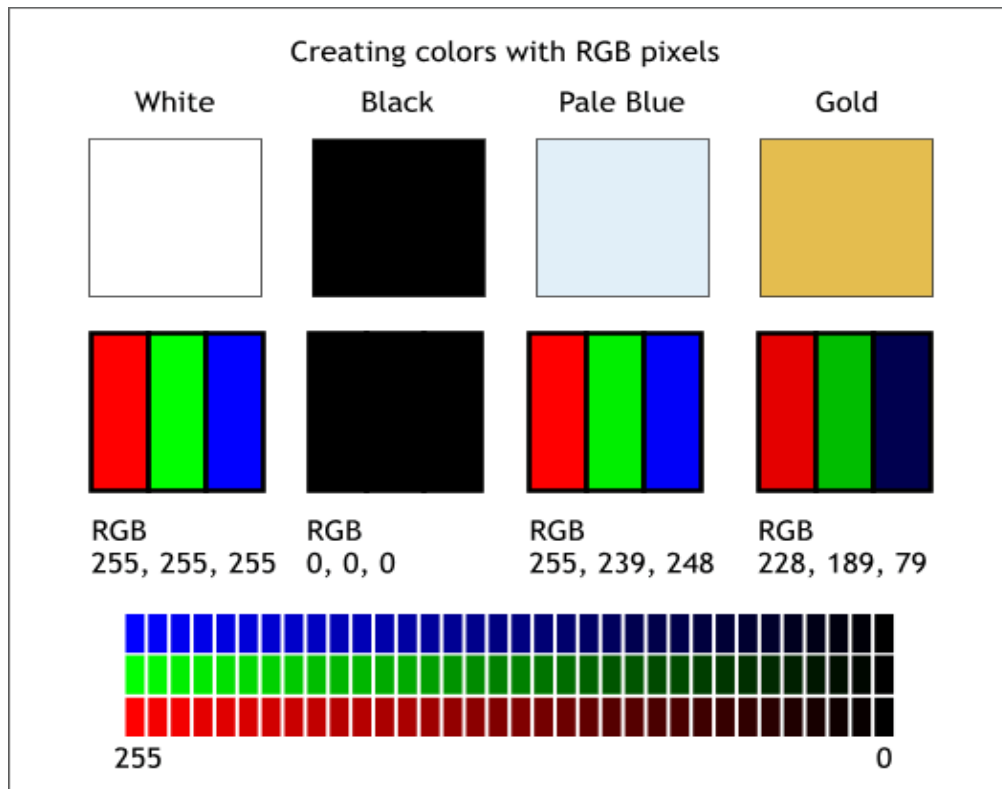


Figure 5.1.9: Creating colors with RGB pixels

For Some Perspective On How Computationally Expensive This Is, Consider This Tree:

- Each Color Value Is Stored In 8 Bits.
- 8 Bits X 3 Colors Per Pixel = 24 Bits Per Pixel.
- A Normal Sized 1024 X 768 Image X 24 Bits Per Pixel = Almost 19m Bits, Or About 2.36 Megabytes.

That’s A Lot Of Memory To Require For One Image, And A Lot Of Pixels For An Algorithm To Iterate Over. But To Train A Model With Meaningful Accuracy – Especially When You’re Talking About Deep Learning – You’d Usually Need Tens Of Thousands Of Images, And The

More The Merrier. Even If You Were To Use Transfer Learning To Use The Insights Of An Already Trained Model, You'd Still Need A Few Thousand Images To Train Yours On.

With The Sheer Amount Of Computing Power And Storage Required Just To Train Deep Learning Models For Computer Vision, It's Not Hard To Understand Why Advances In Those Two Fields Have Driven Machine Learning Forward To Such A Degree.

Business Use Cases For Computer Vision

Computer Vision Is One Of The Areas In Machine Learning Where Core Concepts Are Already Being Integrated Into Major Products That We Use Every Day. Google Is Using Maps To Leverage Their Image Data And Identify Street Names, Businesses, And Office Buildings. Facebook Is Using Computer Vision To Identify People In Photos, And Do A Number Of Things With That Information.

But It's Not Just Tech Companies That Are Leverage Machine Learning For Image Applications. Ford, The American Car Manufacturer That Has Been Around Literally Since The Early 1900's, Is Investing Heavily In Autonomous Vehicles (AVs). Much Of The Underlying Technology In AVs Relies On Analyzing The Multiple Video Feeds Coming Into The Car And Using Computer Vision To Analyze And Pick A Path Of Action.

Another Major Area Where Computer Vision Can Help Is In The Medical Field. Much Of Diagnosis Is Image Processing, Like Reading X-Rays, MRI Scans, And Other Types Of Diagnostics. Google Has Been Working With Medical Research Teams To Explore How Deep Learning Can Help Medical Workflows, And Have Made Significant Progress In Terms Of Accuracy. To Paraphrase From Their Research Page:

But Aside From The Groundbreaking Stuff, It's Getting Much Easier To Integrate Computer Vision Into Your Own Applications. A Number Of High-Quality Third Party Providers Like Clarifai Offer A Simple API For Tagging And Understanding Images, While Kairos Provides Functionality Around Facial Recognition. We'll Dive Into The Open-Source Packages Available For Use Below.

Computer Vision On Algorithmia

Algorithmia Makes It Easy To Deploy Computer Vision Applications As Scalable Microservices. Our Marketplace Has A Few Algorithms To Help Get The Job Done:

- Salnet Automatically Identifies The Most Important Parts Of An Image
- Nudity Detection Detects Nudity In Pictures
- Emotion Recognition Parses Emotions Exhibited In Images

- Deepstyle Transfers Next-Level Filters Onto Your Image
- Face Recognition...Recognizes Faces.
- Image Memorability Judges How Memorable An Image Is.

A Typical Workflow For Your Product Might Involve Passing Images From A Security Camera Into Emotion Recognition And Raising A Flag If Any Aggressive Emotions Are Exhibited, Or Using Nudity Detection To Block Inappropriate Profile Pictures On Your Web Application. For A More Detailed Exploration Of How You Can Use The Algorithmia Platform To Implement Complex And Useful Computer Vision Tasks.

Computer Vision Resources

Packages And Frameworks

Opencv – “Opencv Was Designed For Computational Efficiency And With A Strong Focus On Real-Time Applications. Adopted All Around The World, Opencv Has More Than 47 Thousand People Of User Community And Estimated Number Of Downloads Exceeding 14 Million. Usage Ranges From Interactive Art, To Mines Inspection, Stitching Maps On The Web Or Through Advanced Robotics.”

Simplecv – “Simplecv Is An Open Source Framework For Building Computer Vision Applications. With It, You Get Access To Several High-Powered Computer Vision Libraries Such As Opencv – Without Having To First Learn About Bit Depths, File Formats, Color Spaces, Buffer Management, Eigenvalues, Or Matrix Versus Bitmap Storage.”

Mahotas – “Mahotas Is A Computer Vision And Image Processing Library For Python. It Includes Many Algorithms Implemented In C++ For Speed While Operating In Numpy Arrays And With A Very Clean Python Interface. Mahotas Currently Has Over 100 Functions For Image Processing And Computer Vision And It Keeps Growing.

5.3 NUMPY

- Numpy, Which Stands For Numerical Python, Is A Library Consisting Of Multidimensional Array Objects And A Collection Of Routines For Processing Those Arrays. Using Numpy, Mathematical And Logical Operations On Arrays Can Be Performed. This Tutorial Explains The Basics Of Numpy Such As Its Architecture And Environment. It Also Discusses The Various Array Functions, Types Of Indexing, Etc. An Introduction To Matplotlib Is Also Provided. All This Is Explained With The Help Of Examples For Better Understanding.
- Audience
- This Tutorial Has Been Prepared For Those Who Want To Learn About The Basics And Various Functions Of Numpy. It Is Specifically Useful For Algorithm Developers. After Completing This Tutorial, You Will Find Yourself At A Moderate Level Of Expertise From Where You Can Take Yourself To Higher Levels Of Expertise.
- Prerequisites
- You Should Have A Basic Understanding Of Computer Programming Terminologies. A Basic Understanding Of Python And Any Of The Programming Languages Is A Plus.
- Numpy Is A Python Package. It Stands For 'Numerical Python'. It Is A Library Consisting Of Multidimensional Array Objects And A Collection Of Routines For Processing Of Array.

Numeric, The Ancestor Of Numpy, Was Developed By Jim Hugunin. Another Package Numarray Was Also Developed, Having Some Additional Functionalities. In 2005, Travis Oliphant Created Numpy Package By Incorporating The Features Of Numarray Into Numeric Package. There Are Many Contributors To This Open Source Project.

Operations Using Numpy:

Using Numpy, A Developer Can Perform The Following Operations –

- Mathematical And Logical Operations On Arrays.
- Fourier Transforms And Routines For Shape Manipulation.
- Operations Related To Linear Algebra. Numpy Has In-Built Functions For Linear Algebra And Random Number Generation.

Numpy – A Replacement For Matlab

Numpy Is Often Used Along With Packages Like **Scipy** (Scientific Python) And **Mat-Plotlib** (Plotting Library). This Combination Is Widely Used As A Replacement

For Matlab, A Popular Platform For Technical Computing. However, Python Alternative To Matlab Is Now Seen As A More Modern And Complete Programming Language.

The Most Important Object Defined In Numpy Is An N-Dimensional Array Type Called **Ndarray**. It Describes The Collection Of Items Of The Same Type. Items In The Collection Can Be Accessed Using A Zero-Based Index.

Every Item In An Ndarray Takes The Same Size Of Block In The Memory. Each Element In Ndarray Is An Object Of Data-Type Object (Called **Dtype**).

Any Item Extracted From Ndarray Object (By Slicing) Is Represented By A Python Object Of One Of Array Scalar Types. The Following Diagram Shows A Relationship Between Ndarray, Data Type Object (Dtype) And Array Scalar Type –

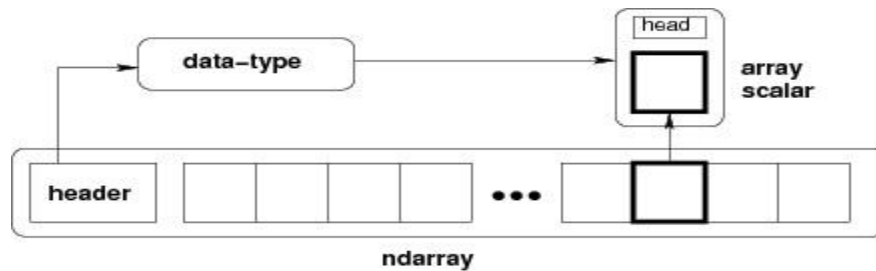


Figure 5.2.1: Nd Array

An Instance Of Ndarray Class Can Be Constructed By Different Array Creation Routines Described Later In The Tutorial. The Basic Ndarray Is Created Using An Array Function In Numpy As Follows –

Numpy.Array

It Creates An Ndarray From Any Object Exposing Array Interface, Or From Any Method That Returns An Array.

Imutils

A Series Of Convenience Functions To Make Basic Image Processing Operations Such As Translation, Rotation, Resizing, Skeletonization, And Displaying Matplotlib Images Easier With Opencv And Python.

Translation Is The Shifting Of An Image In Either The X Or Y Direction. To Translate An Image In Opencv You Need To Supply The (X, Y) -Shift, Denoted As (T_x, T_y) To Construct The Translation Matrix M :

$$M = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$$

And From There, You Would Need To Apply The `Cv2.WarpAffine` Function. Instead Of Manually Constructing The Translation Matrix M And Calling `Cv2.WarpAffine`, You Can Simply Make A Call To The `Translate` Function Of `Imutils`.

Rotation

Rotating An Image In `Opencv` Is Accomplished By Making A Call To `Cv2.GetRotationMatrix2d` And `Cv2.WarpAffine`. Further Care Has To Be Taken To Supply The (X, Y)-Coordinate Of The Point The Image Is To Be Rotated About. These Calculation Calls Can Quickly Add Up And Make Your Code Bulky And Less Readable. The `Rotate` Function In `imutils` Helps Resolve This Problem.

Resizing

Resizing An Image In `Opencv` Is Accomplished By Calling The `Cv2.Resize` Function. However, Special Care Needs To Be Taken To Ensure That The Aspect Ratio Is Maintained. This `resize` Function Of `Imutils` Maintains The Aspect Ratio And Provides The Keyword Arguments `width` And `height` So The Image Can Be Resized To The Intended Width/Height While (1) Maintaining Aspect Ratio And (2) Ensuring The Dimensions Of The Image Do Not Have To Be Explicitly Computed By The Developer. Another Optional Keyword Argument, `inter`, Can Be Used To Specify Interpolation Method As Well.

Skeletonization Is The Process Of Constructing The “Topological Skeleton” Of An Object In An Image, Where The Object Is Presumed To Be White On A Black Background. `Opencv` Does Not Provide A Function To Explicitly Construct The Skeleton, But Does Provide The Morphological And Binary Functions To Do So.

For Convenience, The `skeletonize` Function Of `Imutils` Can Be Used To Construct The Topological Skeleton Of The Image.

The First Argument, `size` Is The Size Of The Structuring Element Kernel. An Optional Argument, `structuring`, Can Be Used To Control The Structuring Element — It Defaults To `Cv2.Morph_Rect`, But Can Be Any Valid Structuring Element.

Displaying With `Matplotlib`

In The Python Bindings Of `Opencv`, Images Are Represented As `Numpy` Arrays In `Bgr` Order. This Works Fine When Using The `Cv2.Imshow` Function. However, If You Intend On Using `Matplotlib`, The `plt.imshow` Function Assumes The Image Is In `Rgb` Order. A Simple Call To `Cv2.cvtColor` Will Resolve This Problem, Or You Can Use The `Opencv2matplotlib` Convenience Function.

5.4 SOURCE CODE

```
import numpy as np

import pandas as pand

import os

import cv2

from matplotlib import pyplot as plt

def txt_encode(text):

    l=len(text)

    i=0

    add=""

    while i<l:

        t=ord(text[i])

        if(t>=32 and t<=64):

            t1=t+48

            t2=t1^170 #170: 10101010

            res = bin(t2)[2:].zfill(8)

            add+="0011"+res

        else:

            t1=t-48
```

```
t2=t1^170

res = bin(t2)[2:].zfill(8)

add+="0110"+res

i+=1

res1=add+"111111111111"

print("The string after binary conversion applying all the transformation :- " + (res1))

length = len(res1)

print("Length of binary after conversion:- ",length)

HM_SK=""

ZWC={"00":u'\u200C',"01":u'\u202C',"11":u'\u202D',"10":u'\u200E'}

file1 = open("coverttext.txt","r+")

nameoffile = input("\nEnter the name of the Stego file after Encoding(with extension):- ")

file3= open(nameoffile,"w+", encoding="utf-8")

word=[]

for line in file1:

    word+=line.split()

i=0

while(i<len(res1)):

    s=word[int(i/12)]
```

```
j=0

x=""

HM_SK=""

while(j<12):

    x=res1[j+i]+res1[i+j+1]

    HM_SK+=ZWC[x]

    j+=2

s1=s+HM_SK

file3.write(s1)

file3.write(" ")

i+=12

t=int(len(res1)/12)

while t<len(word):

    file3.write(word[t])

    file3.write(" ")

    t+=1

file3.close()

file1.close()

print("\nStego file has successfully generated")
```



```
def encode_txt_data():

    count2=0

    file1 = open("coverttext.txt","r")

    for line in file1:

        for word in line.split():

            count2=count2+1

    file1.close()

    bt=int(count2)

    print("Maximum number of words that can be inserted :- ",int(bt/6))

    text1=input("\nEnter data to be encoded:- ")

    l=len(text1)

    if(l<=bt):

        print("\nInputed message can be hidden in the cover file\n")

        txt_encode(text1)

    else:

        print("\nString is too big please reduce string size")

        encode_txt_data()

def BinaryToDecimal(binary):

    string = int(binary, 2)
```

```
    return string

def decode_txt_data():

    ZWC_reverse={u"\u200C':"00",u"\u202C':"01",u"\u202D':"11",u"\u200E':"10"}

    stego=input("\nPlease enter the stego file name(with extension) to decode the message:- ")

    file4= open(stego,"r", encoding="utf-8")

    temp=""

    for line in file4:

        for words in line.split():

            T1=words

            binary_extract=""

            for letter in T1:

                if(letter in ZWC_reverse):

                    binary_extract+=ZWC_reverse[letter]

            if binary_extract=="111111111111":

                break

            else:

                temp+=binary_extract

    print("\nEncrypted message presented in code bits:",temp)
```

```
lengthd = len(temp)

print("\nLength of encoded bits:- ",lengthd)

i=0

a=0

b=4

c=4

d=12

final=""

while i<len(temp):

    t3=temp[a:b]

    a+=12

    b+=12

    i+=12

    t4=temp[c:d]

    c+=12

    d+=12

    if(t3=='0110'):

        decimal_data = BinaryToDecimal(t4)

        final+=chr((decimal_data ^ 170) + 48)
```

```
elif(t3=='0011'):

    decimal_data = BinaryToDecimal(t4)

    final+=chr((decimal_data ^ 170) - 48)

print("\nMessage after decoding from the stego file:- ",final)

def txt_steg():

    while True:

        print("\n\t\tTEXT STEGANOGRAPHY OPERATIONS")

        print("1. Encode the Text message")

        print("2. Decode the Text message")

        print("3. Exit")

        choice1 = int(input("Enter the Choice:"))

        if choice1 == 1:

            encode_txt_data()

        elif choice1 == 2:

            decrypted=decode_txt_data()

        elif choice1 == 3:

            break

        else:

            print("Incorrect Choice")
```

```

print("\n")

def msgtobinary(msg):

    if type(msg) == str:

        result= ".join([ format(ord(i), "08b") for i in msg ])

    elif type(msg) == bytes or type(msg) == np.ndarray:

        result= [ format(i, "08b") for i in msg ]

    elif type(msg) == int or type(msg) == np.uint8:

        result=format(msg, "08b")

    else:

        raise TypeError("Input type is not supported in this function")

    return result

def encode_img_data(img):

    data=input("\nEnter the data to be Encoded in Image :")

    if (len(data) == 0):

        raise ValueError('Data entered to be encoded is empty')

    nameoffile = input("\nEnter the name of the New Image (Stego Image) after
Encoding(with extension):")

    no_of_bytes=(img.shape[0] * img.shape[1] * 3) // 8

    print("\t\nMaximum bytes to encode in Image :", no_of_bytes)

```

```
if(len(data)>no_of_bytes):

    raise ValueError("Insufficient bytes Error, Need Bigger Image or give Less Data !!")

data +='*^*^*'

binary_data=msgtobinary(data)

print("\n")

print(binary_data)

length_data=len(binary_data)

print("\nThe Length of Binary data",length_data)

index_data = 0

for i in img:

    for pixel in i:

        r, g, b = msgtobinary(pixel)

        if index_data < length_data:

            pixel[0] = int(r[:-1] + binary_data[index_data], 2)

            index_data += 1

        if index_data < length_data:

            pixel[1] = int(g[:-1] + binary_data[index_data], 2)

            index_data += 1

        if index_data < length_data:
```

```
    pixel[2] = int(b[:-1] + binary_data[index_data], 2)

    index_data += 1

if index_data >= length_data:

    break

cv2.imwrite(nameoffile,img)

print("\nEncoded the data successfully in the Image and the image is successfully saved
with name ",nameoffile)

def decode_img_data(img):

    data_binary = ""

    for i in img:

        for pixel in i:

            r, g, b = msgtobinary(pixel)

            data_binary += r[-1]

            data_binary += g[-1]

            data_binary += b[-1]

    total_bytes = [ data_binary[i: i+8] for i in range(0,len(data_binary), 8) ]

    decoded_data = ""

    for byte in total_bytes:

        decoded_data += chr(int(byte, 2))
```

```
if decoded_data[-5:] == "*^*^*":

    print("\n\nThe Encoded data which was hidden in the Image was :--",decoded_data[:-5])

    return

def img_steg():

    while True:

        print("\n\t\tIMAGE STEGANOGRAPHY OPERATIONS\n")

        print("1. Encode the Text message")

        print("2. Decode the Text message")

        print("3. Exit")

        choice1 = int(input("Enter the Choice: "))

        if choice1 == 1:

            image=cv2.imread("stego.png")

            encode_img_data(image)

        elif choice1 == 2:

            image1=cv2.imread(input("Enter the Image you need to Decode to get the Secret message : "))

            decode_img_data(image1)

        elif choice1 == 3:
```



```
        break

    else:

        print("Incorrect Choice")

        print("\n")

def encode_aud_data():

    import wave

    nameoffile=input("Enter name of the file (with extension) :- ")

    song = wave.open(nameoffile, mode='rb')

    nframes=song.getnframes()

    frames=song.readframes(nframes)

    frame_list=list(frames)

    frame_bytes=bytearray(frame_list)

    data = input("\nEnter the secret message :- ")

    res = ".join(format(i, '08b') for i in bytearray(data, encoding='utf-8'))

    print("\nThe string after binary conversion :- " + (res))

    length = len(res)

    print("\nLength of binary after conversion :- ",length)

    data = data + '*^*^*'

    result = []
```

```

for c in data:

    bits = bin(ord(c))[2:].zfill(8)

    result.extend([int(b) for b in bits])

j = 0

for i in range(0,len(result),1):

    res = bin(frame_bytes[j])[2:].zfill(8)

    if res[len(res)-4]== result[i]:

        frame_bytes[j] = (frame_bytes[j] & 253) #253: 111111013

    else:

        frame_bytes[j] = (frame_bytes[j] & 253) | 2

        frame_bytes[j] = (frame_bytes[j] & 254) | result[i]

    j = j + 1

frame_modified = bytes(frame_bytes)

stegofile=input("\nEnter name of the stego file (with extension) :- ")

with wave.open(stegofile, 'wb') as fd:

    fd.setparams(song.getparams())

    fd.writeframes(frame_modified)

print("\nEncoded the data successfully in the audio file.")

song.close()

```

```
def decode_aud_data():

    import wave

    nameoffile=input("Enter name of the file to be decoded :- ")

    song = wave.open(nameoffile, mode='rb')

    nframes=song.getnframes()

    frames=song.readframes(nframes)

    frame_list=list(frames)

    frame_bytes=bytearray(frame_list)

    extracted = ""

    p=0

    for i in range(len(frame_bytes)):

        if(p==1):

            break

        res = bin(frame_bytes[i])[2:].zfill(8)

        if res[len(res)-2]==0:

            extracted+=res[len(res)-4]

        else:

            extracted+=res[len(res)-1]

    all_bytes = [ extracted[i: i+8] for i in range(0, len(extracted),8) ]
```

```
decoded_data = ""

for byte in all_bytes:

    decoded_data += chr(int(byte, 2))

    if decoded_data[-5:] == "*^*^*":

        print("The Encoded data was :--",decoded_data[:-5])

        p=1

        break

def aud_steg():

    while True:

        print("\n\t\tAUDIO STEGANOGRAPHY OPERATIONS")

        print("1. Encode the Text message")

        print("2. Decode the Text message")

        print("3. Exit")

        choice1 = int(input("Enter the Choice:"))

        if choice1 == 1:

            encode_aud_data()

        elif choice1 == 2:

            decode_aud_data()

        elif choice1 == 3:
```

```
        break

    else:

        print("Incorrect Choice")

        print("\n")

def KSA(key):

    key_length = len(key)

    S=list(range(256))

    j=0

    for i in range(256):

        j=(j+S[i]+key[i % key_length]) % 256

        S[i],S[j]=S[j],S[i]

    return S

def PRGA(S,n):

    i=0

    j=0

    key=[]

    while n>0:

        n=n-1

        i=(i+1)%256
```

```

j=(j+S[i])%256

S[i],S[j]=S[j],S[i]

K=S[(S[i]+S[j])%256]

key.append(K)

return key

def preparing_key_array(s):

    return [ord(c) for c in s]

def encryption(plaintext):

    key=input("Enter the key : ")

    key=preparing_key_array(key)

    S=KSA(key)

    keystream=np.array(PRG(S,len(plaintext)))

    plaintext=np.array([ord(i) for i in plaintext])

    cipher=keystream^plaintext

    ctext=""

    for c in cipher:

        ctext=ctext+chr(c)

    return ctext

def decryption(ciphertext):

```

```

key=input("Enter the key : ")

key=preparing_key_array(key)

S=KSA(key)

keystream=np.array(PRGA(S,len(ciphertext)))

ciphertext=np.array([ord(i) for i in ciphertext])

decoded=keystream^ciphertext

dtext=""

for c in decoded:

    dtext=dtext+chr(c)

return dtext

def embed(frame):

    data=input("\nEnter the data to be Encoded in Video :")

    data=encryption(data)

    print("The encrypted data is : ",data)

    if (len(data) == 0):

        raise ValueError('Data entered to be encoded is empty')

    data +='*^*^*'

    binary_data=msgtobinary(data)

    length_data = len(binary_data)

```

```
index_data = 0

for i in frame:

    for pixel in i:

        r, g, b = msgtobinary(pixel)

        if index_data < length_data:

            pixel[0] = int(r[:-1] + binary_data[index_data], 2)

            index_data += 1

        if index_data < length_data:

            pixel[1] = int(g[:-1] + binary_data[index_data], 2)

            index_data += 1

        if index_data < length_data:

            pixel[2] = int(b[:-1] + binary_data[index_data], 2)

            index_data += 1

        if index_data >= length_data:

            break

    return frame

def extract(frame):

    data_binary = ""

    final_decoded_msg = ""
```



```

for i in frame:

    for pixel in i:

        r, g, b = msgtobinary(pixel)

        data_binary += r[-1]

        data_binary += g[-1]

        data_binary += b[-1]

    total_bytes = [ data_binary[i: i+8] for i in range(0,len(data_binary), 8) ]

    decoded_data = ""

    for byte in total_bytes:

        decoded_data += chr(int(byte, 2))

        if decoded_data[-5:] == "*^*^*^*":

            for i in range(0,len(decoded_data)-5):

                final_decoded_msg += decoded_data[i]

            final_decoded_msg = decryption(final_decoded_msg)

            print("\n\nThe Encoded data which was hidden in the Video was :--
\n",final_decoded_msg)

            return

def encode_vid_data():

```

```
cap=cv2.VideoCapture(input("Enter the name of the Video file in which you want to
Encode data :"))

fourcc = cv2.VideoWriter_fourcc(*'XVID')

frame_width = int(cap.get(3))

frame_height = int(cap.get(4))

size = (frame_width, frame_height)

max_frame=cap.get(cv2.CAP_PROP_FRAME_COUNT)

print("Total number of Frame in selected Video :",max_frame)

n=int(input("Enter the frame number where you want to embed data :"))

out = cv2.VideoWriter(input("\nEnter the name of the Stego video after Encoding(with
extension):"),fourcc, 25.0, size)

frame_number = 0

while(cap.isOpened()):

    frame_number += 1

    ret, frame = cap.read()

    if ret == False:

        break

    if frame_number == n:

        change_frame_with = embed(frame)
```

```
    frame_ = change_frame_with

    frame = change_frame_with

    out.write(frame)

    print("\nEncoded the data successfully in the video file.")

    return frame_

def decode_vid_data(frame_):

    cap = cv2.VideoCapture(input("\nEnter the name of the Stego video to be Decoded(with extension):"))

    max_frame=cap.get(cv2.CAP_PROP_FRAME_COUNT)

    print("Total number of Frame in selected Video :",max_frame)

    n=int(input("Enter the secret frame number from where you want to extract data:"))

    frame_number = 0

    while(cap.isOpened()):

        frame_number += 1

        ret, frame = cap.read()

        if ret == False:

            break

        if frame_number == n:

            extract(frame_)
```

```
    return

def vid_steg():

    while True:

        print("\n\t\tVIDEO STEGANOGRAPHY OPERATIONS")

        print("1. Encode the Text message")

        print("2. Decode the Text message")

        print("3. Exit")

        choice1 = int(input("Enter the Choice:"))

        if choice1 == 1:

            a=encode_vid_data()

        elif choice1 == 2:

            decode_vid_data(a)

        elif choice1 == 3:

            break

        else:

            print("Incorrect Choice")

            print("\n")

def main():

    print("\t\tSTEGANOGRAPHY")
```



```
print("Incorrect Choice")
```

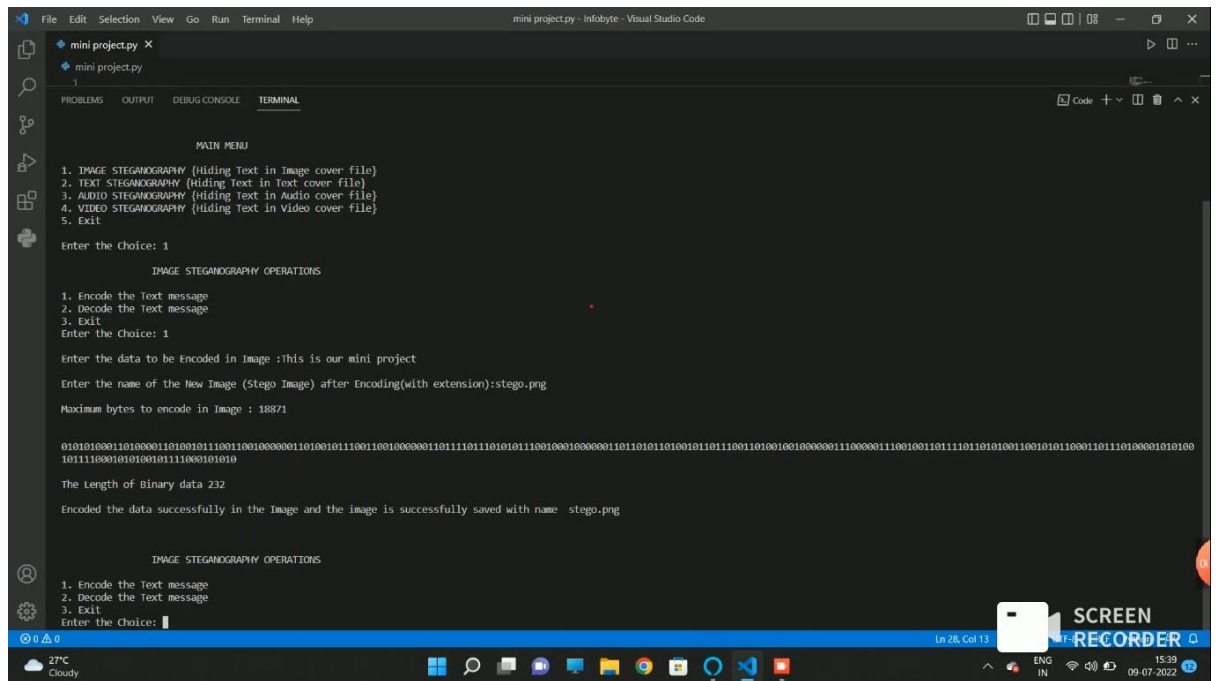
```
print("\n\n")
```

```
if __name__ == "__main__":
```

```
    main()
```

6.1 SCREENSHOTS

6.1.1 IMAGE STEGANOGRAPHY

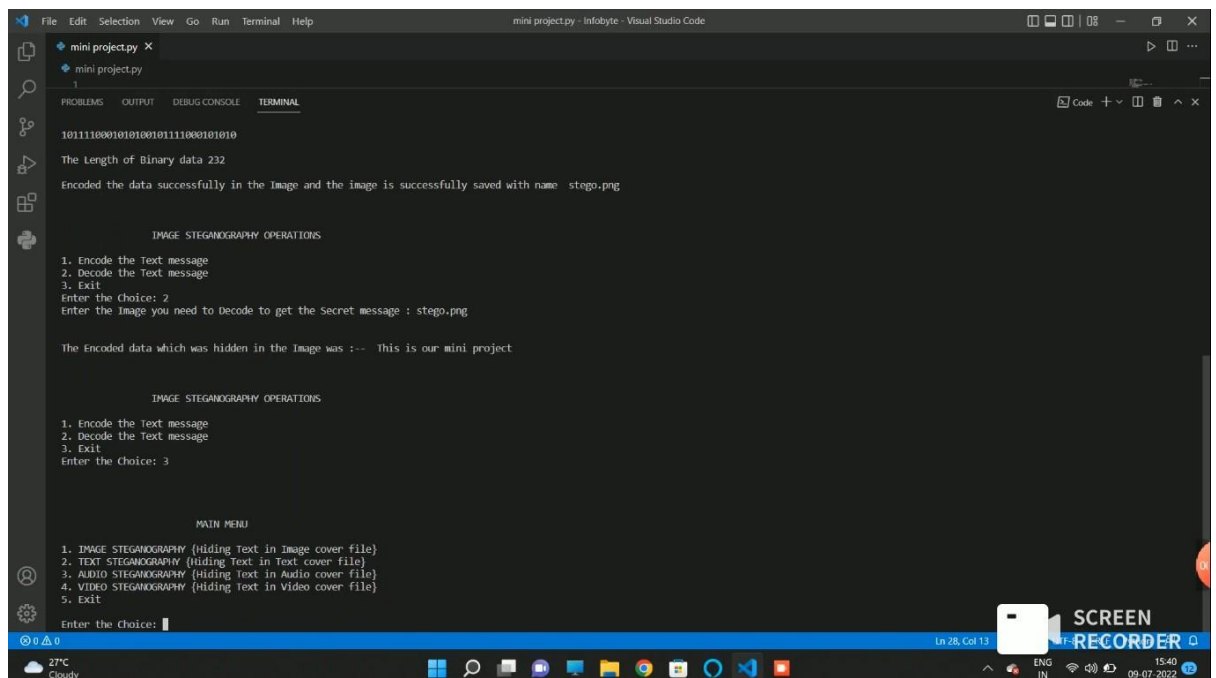


```

File Edit Selection View Go Run Terminal Help
mini project.py - Infobyte - Visual Studio Code
mini project.py X
mini project.py
1
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
MAIN MENU
1. IMAGE STEGANOGRAPHY (Hiding Text in Image cover file)
2. TEXT STEGANOGRAPHY (Hiding Text in Text cover file)
3. AUDIO STEGANOGRAPHY (Hiding Text in Audio cover file)
4. VIDEO STEGANOGRAPHY (Hiding Text in Video cover file)
5. Exit
Enter the choice: 1
IMAGE STEGANOGRAPHY OPERATIONS
1. Encode the Text message
2. Decode the Text message
3. Exit
Enter the Choice: 1
Enter the data to be encoded in Image :This is our mini project
Enter the name of the New Image (Stego Image) after Encoding(with extension):stego.png
Maximum bytes to encode in Image : 18871
0101010001101000110100101100110010000001101001011001100100000011010101101001011011001101001001000000110000011001001101101010010010110000101010001010100
101111000101010010111000101010
The Length of Binary data 232
Encoded the data successfully in the Image and the image is successfully saved with name stego.png
IMAGE STEGANOGRAPHY OPERATIONS
1. Encode the Text message
2. Decode the Text message
3. Exit
Enter the Choice:

```

Figure 6.1.1.1: Image Steganography Encoding



```

File Edit Selection View Go Run Terminal Help
mini project.py - Infobyte - Visual Studio Code
mini project.py X
mini project.py
1
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
101111000101010010111000101010
The Length of Binary data 232
Encoded the data successfully in the Image and the image is successfully saved with name stego.png
IMAGE STEGANOGRAPHY OPERATIONS
1. Encode the Text message
2. Decode the Text message
3. Exit
Enter the Choice: 2
Enter the Image you need to Decode to get the Secret message : stego.png
The Encoded data which was hidden in the Image was :-- This is our mini project
IMAGE STEGANOGRAPHY OPERATIONS
1. Encode the Text message
2. Decode the Text message
3. Exit
Enter the Choice: 3
MAIN MENU
1. IMAGE STEGANOGRAPHY (Hiding Text in Image cover file)
2. TEXT STEGANOGRAPHY (Hiding Text in Text cover file)
3. AUDIO STEGANOGRAPHY (Hiding Text in Audio cover file)
4. VIDEO STEGANOGRAPHY (Hiding Text in Video cover file)
5. Exit
Enter the Choice:

```

Figure 6.1.1.2: Image Steganography Decoding



6.1.3 AUDIO STEGANOGRAPHY

```

File Edit Selection View Go Run Terminal Help
mini project.py - Infobyte - Visual Studio Code
mini project.py X
mini project.py
1
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Enter the Choice:3

MAIN MENU
1. IMAGE STEGANOGRAPHY (Hiding Text in Image cover file)
2. TEXT STEGANOGRAPHY (Hiding Text in Text cover file)
3. AUDIO STEGANOGRAPHY (Hiding Text in Audio cover file)
4. VIDEO STEGANOGRAPHY (Hiding Text in Video cover file)
5. Exit
Enter the Choice: 3

AUDIO STEGANOGRAPHY OPERATIONS
1. Encode the Text message
2. Decode the Text message
3. Exit
Enter the Choice:1
Enter name of the file (with extension) :- sample.wav
Enter the secret message :- This is our mini project

The string after binary conversion :- 010101000110100001101001011100110010000001101001110011001000000110111011010111001000100000011011010110100101100110010010000001110000011100100110111011
01010011001010110001101110100

Length of binary after conversion :- 192
Enter name of the stego file (with extension) :- stego.wav
Encoded the data successfully in the audio file.

AUDIO STEGANOGRAPHY OPERATIONS
1. Encode the Text message
2. Decode the Text message
3. Exit
Enter the Choice:

```

Figure 6.1.3.1: Audio Steganography Encoding

```

File Edit Selection View Go Run Terminal Help
mini project.py - Infobyte - Visual Studio Code
mini project.py X
mini project.py
1
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
The string after binary conversion :- 01010100011010000110100101110011001000000110100111001100100000011011010110100100100000011011010110100101100110010010000001110000011100100110111011
01010011001010110001101110100

Length of binary after conversion :- 192
Enter name of the stego file (with extension) :- stego.wav
Encoded the data successfully in the audio file.

AUDIO STEGANOGRAPHY OPERATIONS
1. Encode the Text message
2. Decode the Text message
3. Exit
Enter the Choice:2
Enter name of the file to be decoded :- stego.wav
The Encoded data was :- This is our mini project

AUDIO STEGANOGRAPHY OPERATIONS
1. Encode the Text message
2. Decode the Text message
3. Exit
Enter the Choice:3

MAIN MENU
1. IMAGE STEGANOGRAPHY (Hiding Text in Image cover file)
2. TEXT STEGANOGRAPHY (Hiding Text in Text cover file)
3. AUDIO STEGANOGRAPHY (Hiding Text in Audio cover file)
4. VIDEO STEGANOGRAPHY (Hiding Text in Video cover file)
5. Exit
Enter the Choice:

```

Figure 6.1.3.2: Audio Steganography Decoding

6.1.4 VIDEO STEGANOGRAPHY

```

mini project.py X
mini project.py
1
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
MAIN MENU
1. IMAGE STEGANOGRAPHY (Hiding Text in Image cover file)
2. TEXT STEGANOGRAPHY (Hiding Text in Text cover file)
3. AUDIO STEGANOGRAPHY (Hiding Text in Audio cover file)
4. VIDEO STEGANOGRAPHY (Hiding Text in Video cover file)
5. Exit
Enter the Choice: 4
VIDEO STEGANOGRAPHY OPERATIONS
1. Encode the Text message
2. Decode the Text message
3. Exit
Enter the Choice:1
Enter the name of the Video file in which you want to Encode data :video.mp4
Total number of Frame in selected Video : 1368.0
Enter the frame number where you want to embed data :7
Enter the name of the Stego video after Encoding(with extension):stego.mp4
OpenCV: FFMPEG: tag 0x44A95658/'XVID' is not supported with codec id 12 and format 'mp4 / MP4 (MPEG-4 Part 14)'
OpenCV: FFMPEG: fallback to use tag 0x7634786d/'mp4v'
Enter the data to be Encoded in Video :This is our mini project
Enter the key : key
j'>z[{4]The encrypted data is :  .]
Encoded the data successfully in the video file.
VIDEO STEGANOGRAPHY OPERATIONS
1. Encode the Text message
2. Decode the Text message
3. Exit
Enter the Choice:

```

Figure 6.1.4.1: Video Steganography Encoding

```

mini project.py X
mini project.py
1
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Encoded the data successfully in the video file.
VIDEO STEGANOGRAPHY OPERATIONS
1. Encode the Text message
2. Decode the Text message
3. Exit
Enter the Choice:2
Enter the name of the Stego video to be Decoded(with extension):stego.mp4
Total number of Frame in selected Video : 1368.0
Enter the secret frame number from where you want to extract data:7
Enter the key : Key
The Encoded data which was hidden in the Video was :--
This is our mini project
VIDEO STEGANOGRAPHY OPERATIONS
1. Encode the Text message
2. Decode the Text message
3. Exit
Enter the Choice:3
MAIN MENU
1. IMAGE STEGANOGRAPHY (Hiding Text in Image cover file)
2. TEXT STEGANOGRAPHY (Hiding Text in Text cover file)
3. AUDIO STEGANOGRAPHY (Hiding Text in Audio cover file)
4. VIDEO STEGANOGRAPHY (Hiding Text in Video cover file)
5. Exit
Enter the Choice:

```

Figure 6.1.4.2: Video Steganography Decoding

7.1 CONCLUSION

Although only some of the main image steganographic techniques were discussed in this document, one can see that there exists a large selection of approaches to hiding information in various carrier files. All the major file formats have different methods of hiding messages, with different strong and weak points respectively. Where one technique lacks in payload capacity, the other lacks in robustness. For example, the patchwork approach has a very high level of robustness against most type of attacks, but can hide only a very small amount of information.

Least significant bit (LSB) in both BMP and GIF makes up for this, but both approaches result in suspicious files that increase the probability of detection when in the presence of a warden.

The proposed approach in this project uses a new steganographic approach called image steganography. The application creates a stego file in which the personal data is embedded inside the cover file.

Used the Least Significant Bit algorithm in this project for developing the application which is faster and reliable and compression ratio is moderate compared to other algorithms.

8.1 FUTURE ENHANCEMENTS

The future work on this project is to improve the compression ratio of the carrier file to the text. Also create a proper Desktop Application.

REFERENCES

1. <https://en.wikipedia.org/wiki/Steganography>
2. <https://www.comptia.org/blog/what-is-steganography>
3. <https://www.geeksforgeeks.org/image-steganography-in-cryptography/>
4. <https://www.jigsawacademy.com/blogs/cyber-security/steganography/>
5. <https://searchsecurity.techtarget.com/definition/steganography>
6. <https://towardsdatascience.com/steganography-hiding-an-image-insideanother-77ca66b2acb1>
7. <https://www.edureka.co/blog/steganography-tutorial>
8. <https://www.ukessays.com/essays/computer-science/steganography-usesmethods-tools3250.php>
9. <https://www.thepythoncode.com/article/hide-secret-data-in-images-usingsteganographypython>
10. <https://www.youtube.com/watch?v=xepNoHgNj0w&t=1922s>