

STEGANOGRAPHY

Project submitted in partial fulfillment of the degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by :

M. Pranay Kumar Reddy	(O161064),
M. Sreelatha	(O161315),
B. Jeenath	(O161615),
G. Elumalai	(O162089),
M. Jitendar	(O161050),

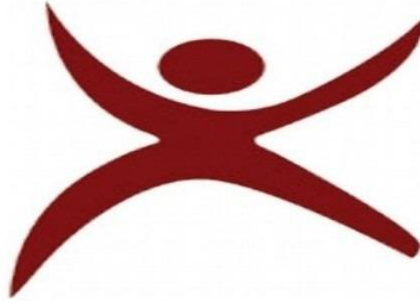
Under the supervision of

Mrs.N MADHAVILATHA M.Tech (Ph.D.)

Assistant Professor



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING,
RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES,
ONGOLE CAMPUS,
APRIL 2022.



BONAFIDE CERTIFICATE

This is to certify that the project report entitled **STEGANOGRAPHY** submitted by **M PRANAY KUMAR REDDY (O161064), M SREE LAHTA (O161315), B JEENATH (O161615), E ELUMALAI (O162089), M JITHENDER (O161050)** in partial fulfilment of the requirement for the award of Bachelor of Technology in Computer Science and Engineering is a record of bonafide project work carried out under my supervision during the academic year 2021-22.

I Am indebted to **Mrs.N MADHAVILATHA**, my project guide for conscientious guidance and encouragement to accomplish this project.

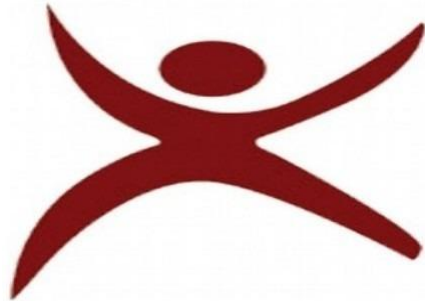
I am extremely thankful and pay my gratitude to **Mrs. K. SANDHYA**, HOD CSE, for her valuable guidance and support on the completion of this project.

The report hasn't been submitted previously in part or in full to this or any other university or institution for the award of any degree.

Mrs.N MadhaviLatha M.Tech (Ph.D.),
Assistant Professor,
Department of CSE,
RGUKT, ONGOLE.

Mrs.K. Sandhya M.Tech (Ph.D.),
Head of the Department,
Department of CSE,
RGUKT, ONGOLE.

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING,
RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES,
ONGOLE CAMPUS, APRIL 2022.**



CERTIFICATE

This is certify that the project report entitled submitted by “**HIDING AND RETRIVING THE DATA THROUGH IMAGES USING PYTHON (STEGANOGRAPHY)**” **M. Pranay Kumar Reddy** (O161064), **Sreelatha** (O161315), **B.Jeenath**(O161615), **G.Elumalai** (O162089), **M.Jitendar** (O161050) and to the Department of Computer Science and Engineering, Rajiv Gandhi University of Knowledge Technologies, Ongole, during the academic year 2021-2022 is a partial fulfillment for the award of Under graduate degree of Bachelor of Technology in Computer Science and Engineering, is a bonafide record carried out by them under my supervision. The project has fulfilled all the requirements as per as regulations of this institute and in my opinion reached the standard for submission.

Mrs.N MadhaviLatha M.Tech (Ph.D.),
Assistant Professor,
Department of CSE,
RGUKT, ONGOLE.

Mrs.K. Sandhya M.Tech (Ph.D.),
Head of the Department,
Department of CSE,
RGUKT, ONGOLE

Date:

Place: RGUKT Ongole

Approval Sheet

This report entitled STEGANOGRAPHY by M Pranay Kumar Reddy(O161064) M Sreelatha (O161315) B Jeenath(O161615) G Elumalai(O162089) M Jitendar(O161050) is N. MADHAVI LATHA approved for the degree of Bachelor of Technology in COMPUTER SCIENCE AND ENGINEERING.

Examiner

Supervisor

Chairman

Date : _____

Place : _____

DECLARATION

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Signature :

Name:

ID:

Name:

ID:

Name

ID:

Name

ID:

Name

ID:

DATE:

ACKNOWLEDGEMENT

I am highly indebted to **N. MADHAVI LATHA** for their guidance and constant supervision as well as for providing necessary information regarding the project and also for their kind co-operation, encouragement and their support in completing the project.

I would like to express my special gratitude and thanks to branch coordinator **K. SANDHYA** mam for giving me such attention and time.

I have taken efforts in this project. However, it could not have been possible without the kind support and help of many individuals and RGUKT. We would like to extend my sincere thanks to all of them.

Our thanks and appreciation also go to my whole team in developing the project and people who have willingly helped me out with their abilities.

With Sincere Regards,
M PRANAY KUMAR REDDY
M SREELATHA
B JEENATH
G ELUMALAI
M JITENDAR.

Date: _____

ABSTRACT

Introduction:

Steganography is the art and science of writing hidden messages in such a way that no one apart from the intended recipient knows of the existence of the message. The word "Steganography" is of Greek origin and means "covered, or hidden writing". An encrypted file may still hide information using Steganography, so even if the encrypted file is deciphered, the hidden message is not seen. Steganography used in electronic communication include steganographic coding inside of a transport layer, such as an image file, or a protocol, such as UDP. The advantage of steganography over cryptography alone is that messages do not attract attention to themselves, to messengers, or to recipients. A steganographic message (the plaintext) is often first encrypted by some traditional means, and then a hiddentext is modified in some way to contain the encrypted message (ciphertext), resulting in stegotext.

Data hiding is a characteristic of object-oriented programming. Because an object can only be associated with data in predefined classes or templates, the object can only "know" about the data it needs to know about. There is no possibility that someone maintaining the code may inadvertently point to or otherwise access the wrong data unintentionally.

Tools Used:

1)Python Compiler : We wrote a code using python by using pillow library. Python 3.0 version had been used in this project for encoding and decoding the data through images.

Keywords: Steganography, Images, python, pillow library, Data, encrypted, Encoding and Decoding

CONTENTS

Contents

BONAFIED CERTIFICATE	I
CERTIFICATE.....	II
APPROVAL SHEET	III
DECLARATION.....	IV
ACKNOWLEDGEMENT.....	V
ABSTRACT.....	VI
CONTENTS.....	VII
LIST OF FIGURES.....	IX

1. INTRODUCTION	1
1.1 Project Defenition.....	3
1.2 Background.....	4
1.3 Motivation.....	4
1.4 Objective.....	4
1.5 Steganography Architecture.....	5
1.6 Oraganization of Dissertation.....	6
2 SYSTEM ANALYSIS	9
2.1 Software Requirement Specification.....	9
2.1.1 Problem Statement	9
2.1.2 Proposed System.....	10
2.1.3 Software Requirements.....	10
2.1.4 Hardware Requirements.....	10
2.1.5 Software tools used –Steganography.....	10
2.1.5.1 Pure Steganography.....	11
2.1.5.2 Public Key Steganography.....	12
2.1.5.3 Public Key Steganography.....	12
2.1.6 Steganography Algorithms.....	13
2.1.6.1 LSB Algorithm.....	13
2.1.7 Python.....	14
2.1.7.1 The very Basics of Python.....	15
2.1.7.2 Basic Properties of Python.....	16
2.1.7.3 Invoking Python.....	16
2.1.7.4 Basic Core Language.....	17
2.1.7.5 Modules.....	17
2.1.7.6 PIL (Python Imaging Library).....	18
2.1.7.7 Image Archives.....	18
2.1.7.8 Image Display.....	18
2.2 Image Processing.....	19

2.2.1	Using the Image Class.....	19
2.2.2	Reading and Writing Images.....	20
3	SYSTEM DESIGN.....	21
3.1	Data Flow Diagram.....	21
3.2	Unified Modified Language.....	22
3.2.1	Use Case Diagram.....	22
3.2.2	Sequence Diagram.....	23
3.2.3	Class Diagram.....	24
3.2.4	State Chart Diagram.....	24
4	SYSTEM IMPLEMENTATION.....	26
4.1	Hiding text inside an Image using Python.....	26
4.2	Encode the Data.....	26
4.3	Decode the Data.....	27
4.4	Code in Python.....	28
4.5	Output.....	34
5	SYSTEM TESTING.....	35
5.1	Unit Testing.....	35
5.2	Performance Testing.....	35
5.3	Compatibility Testing.....	36
6	CONCLUSION AND FUTURE SCOPE.....	37
7	REFERENCES.....	38
8	ANNEXURE.....	39

LIST OF FIGURES

1. Pure Steganography.....	11
2. Secret Steganography...../.....	12
3. Public Key Steganography.....	12
4. Input and output Images.....	14
5. Data Flow Diagram.....	21
6. Use Case Diagram.....	22
7. Sequence Diagram.....	23
8. Class Diagram.....	24
9. State Chart Diagram.....	25
10. Output.....	34

1. INTRODUCTION

Data hiding is a characteristic of object-oriented programming. Because an object can only be associated with data in predefined classes or templates, the object can only "know" about the data it needs to know about. There is no possibility that someone maintaining the code may inadvertently point to or otherwise access the wrong data unintentionally. Thus, all data not required by an object can be said to be "hidden". Data hiding in media, in images, video and audio, is of interest for the protection of copyrighted digital media, and to the government for information systems security and for covert (steganographic) communications. It can also be used in forensic applications for inserting hidden data into audio files for the authentication of spoken words and other sounds, and in the music business for the monitoring of the songs over broadcast radio. The technology enables the insertion and recovery of hidden data in audio files through manipulation of the relative phase of harmonically related components of a host audio signal.

Steganography is the art and science of writing hidden messages in such a way that no one apart from the intended recipient knows of the existence of the message. The word "Steganography" is of Greek origin and means "covered, or hidden writing". An encrypted file may still hide information using Steganography, so even if the encrypted file is deciphered, the hidden message is not seen. Steganography used in electronic communication include steganographic coding inside of a transport layer, such as an image file, or a protocol, such as UDP. The advantage of steganography over cryptography alone is that messages do not attract attention to themselves, to messengers, or to recipients. A steganographic message (the plaintext) is often first encrypted by some traditional means, and then a hiddentext is modified in some way to contain the encrypted message (ciphertext), resulting in stegotext..

This project report intends to give an overview of image steganography, its uses and techniques. It also attempts to identify the requirements of a good steganography algorithm and briefly reflects on which steganographic techniques are more suitable for which applications. This project report describes the software functional and nonfunctional requirements for release 1.0 of the Steganography Image system.

Steganography is the art of hiding the fact that communication is taking place, by hiding information in other information. Many different carrier file formats can be used, but digital images are the most popular because of their frequency on the internet. One of the reasons that intruders can be successful is the most of the information they acquire from a system is in a form that they can read and comprehend. Intruders may reveal the information to others, modify it to misrepresent an individual or organization, or use it to launch an attack. One solution to this problem is, through the use of steganography. Steganography is a technique of hiding information in digital media. In contrast to cryptography, it is not to keep others from knowing the hidden information but it is to keep others from thinking that the information even exists.

Steganography become more important as more people join the cyberspace revolution.

Steganography is the art of concealing information in ways that prevents the detection of hidden messages. Steganography include an array of secret communication methods that hide the message from being seen or discovered.

Due to advances in ICT, most of information is kept electronically. Consequently, the security of information has become a fundamental issue. Besides cryptography, steganography can be employed to secure information. In cryptography, the message or encrypted message is embedded in a digital host before passing it through the network, thus the existence of the message is unknown. Besides hiding data for confidentiality, this approach of information hiding can be extended to copyright protection for digital media: audio, video and images.

The growing possibilities of modern communications need the special means of security

especially on computer network. The network security is becoming more important as the number of data being exchanged on the internet increases. Therefore, the confidentiality and data integrity are requires to protect against unauthorized access and use. This has resulted in an explosive growth of the field of information hiding.

The growing use of Internet needs to take attention while we send and receive personal information in a secured manner. For this, there are many approaches that can

transfer the data into different forms so that their resultant data can be understood if it can be returned back into its original form. This technique is known as encryption. However, a major disadvantage of this method is that the existence of data is not hidden. If someone gives enough time then the unreadable encrypted data may be converted into its original form.

A solution to this problem has already been achieved by using a “steganography” technique to hide data in a cover media so that other cannot notice it. The characteristics of the cover media depends on the amount of data that can be hidden, the perceptibility of the message and its robustness.

In this document, we propose a new system for hiding data stands on many methods and algorithms for image hiding where I store on data file, called sink file in an image file called as container image. The primary objective is to use steganography techniques so as to provide more security and simultaneously using less storage.

1.1 PROJECT DEFENITION :

In this project, we propose to develop a system to hiding data by using "STEGANOGRAPHY" technique as I used many methods stands on some techniques to have at the back-end a software for hiding data based on hiding algorithms. After studying the data hiding algorithms we found many ways to hiding data by using the multimedia files and the main question for me was "Where hidden data hides?" as we found by our search to know where the data hides it's important to know what is the file type of the data that it shall be hidden and the cover file type so it is possible to alter graphic or sound files slightly without losing their overall viability for the viewer and listener. With audio, you can use bits of file that contain sound not audible to the human ear. With graphic images, you can remove redundant bits of color from the image and still produce a picture that looks intact to human eye and is difficult to discern from its original. It is in those bits that stego hides its data.

By the final of our research we developed a software uses an algorithm, to embed data in an image; The purposed system is called "Steganography", the aim of this project is to encrypt the data; the meaning of encrypt is to hide the data over an image using different steganographic algorithms, in this system LSB is the algorithms that we use to hiding the data.

1.2 BACKGROUND:

Information hiding is an emerging research area, which encompasses applications such as copyright protection for digital media, watermarking, fingerprinting, and steganography. In watermarking applications, the message contains information such as owner identification and a digital time stamp, which usually applied for copyright protection. Fingerprint, the owner of the data set embeds a serial number that uniquely identifies the user of the data set. This adds to copyright information to makes it possible to trace any unauthorized used of the data set back to the user. Steganography hide the secrete message within the host data set and presence imperceptible and is to be reliably communicated to a receiver. The host data set is purposely corrupted, but in a covert way, designed to be invisible to an information analysis.

1.3 MOTIVATION :

Digital representation of media facilitates access and potentially improves the portability, efficiency, and accuracy of the information presented. Undesirable effects of facile data access include an increased opportunity for violation of copyright and tampering with or modification of content. The motivation for this work includes the provision of protection of intellectual property rights, an indication of content manipulation, and a means of annotation. Data hiding represents a class of processes used to embed data, such as copyright information, into various forms of media such as image, audio, or text with a minimum amount of perceivable degradation to the "host" signal; i.e., the embedded data should be invisible and inaudible to a human observer. Note that data hiding, while similar to compression, is distinct from encryption. Its goal is not to restrict or regulate access to the host signal, but rather to ensure that embedded data remain inviolate and recoverable.

1.4 OBJECTIVE :

The main objective of the data hiding process using image steganography is to ensure security and a proper flow between two users. Two important uses of data hiding in digital media are to provide proof of the copyright, and assurance of content integrity. Therefore, the data should stay hidden in a host signal, even if that signal is subjected to manipulation as degrading as filtering, resampling, cropping, or lossy data compression. Other applications of data hiding, such as the inclusion of augmentation data, need not be

invariant to detection or removal, since these data are there for the benefit of both the author and the content consumer. Thus, the techniques used for data hiding vary depending on the quantity of data being hidden and the required invariance of those data to manipulation. Since no one method is capable of achieving all these goals, a class of processes is needed to span the range of possible applications.

1.5 STEGANOGRAPHY ARCHITECTURE :

Steganography is the art of hiding and transmitting data through apparently innocuous carriers in an effort to conceal the existence of the data, the word Steganography literally means covered or hiding writing as derived from Greek. Steganography has its place in security. It is not intended to replace cryptography but supplement it. [1] Hiding a message with Steganography methods reduces the chance of a message being detected. If the message is also encrypted then it provides another layer of protection.

Therefore, some Steganographic methods combine traditional Cryptography with Steganography; the sender encrypts the secret message prior to the overall communication process, as it is more difficult for an attacker to detect embedded cipher text in a cover. It has been used through the ages by ordinary people, spies, rulers, government, and armies. There are many stories about Steganography.

For example, ancient Greece used methods for hiding messages such as hiding In the field of Steganography, some terminology has developed. The adjectives 'cover', 'embedded', and 'stego' were defined at the information hiding workshop held in Cambridge, England. The term "cover" refers to description of the original, innocent message, data, audio, video, and so on. Steganography is not a new science; it dates back to ancient times.

Hidden information in the cover data is known as the "embedded" data and information hiding is a general term encompassing many sub disciplines, is a term around a wide range of problems beyond that of embedding message in content. The term hiding here can refer to either making the information undetectable or keeping the existence of the information secret.

Information hiding is a technique of hiding secret using redundant cover data such as images, audios, movies, documents, etc. This technique has recently become important in a number of application areas. For example, digital video, audio, and images are

increasingly embedded with imperceptible marks, which may contain hidden signatures or watermarks that help to prevent unauthorized copy. It is a performance that inserts secret messages into a cover file, so that the existence of the messages is not apparent.

Research in information hiding has tremendously increased during the past decade with commercial interests driving the field. Although the art of concealment “hidden information” as old as the history, but the emergence of computer and the evolution of sciences and techniques breathe life again in this art with the use of new ideas, techniques, drawing on the computer characteristics in the way representation of the data, well-known computer representation of all data including (Multimedia) is binary these representations are often the digital levels and areas and change values-aware of slight not aware or felt by Means sensual of human such as hearing, sight, the advantage use of these properties to hide data in multimedia by replace the values of these sites to the values of data to be hidden, taking into account the acceptable limits for the changeover, and not excee

1.6 Organization of Dissertation:

Steganography thesis is concentrating in making a better Steganography system with sufficient and effective Steganography techniques. The art of encrypting confidential message by a covered medium as audio, picture, video files or text is known as Steganography. Recent Steganography Thesis is carried by combining of cryptography and Steganography techniques for secure transmission of digital data. Steganography thesis relies on analysing the existing Steganography algorithm and its challenges.

Steganography thesis is based on improving secure transmission of data. The previous work in Steganography thesis provides a basic concept of Steganography to new researchers. Excellent technology has been put in to protect personal information. The powerful Steganography programs for hiding MP4 or quick time multimedia files are truecrypt. For Steganography thesis we have good PhD advisors and research scholars to bring out the best in Steganography thesis. Steganography Thesis editing service is also accompanied in our concern.

The technical challenges of data hiding are formidable. Any "holes" to fill with data in a

host signal, either statistical or perceptual, are likely targets for removal by lossy signal compression. The key to successful data hiding is the finding of holes that are not suitable for exploitation by compression algorithms. A further challenge is to fill these holes with data in a way that remains invariant to a large class of host signal transformations. Data-hiding techniques should be capable of embedding data in a host signal with the following restrictions and features:

1. The host signal should be non objectionally degraded and the embedded data should be minimally perceptible. (The goal is for the data to remain hidden. As any magician will tell you, it is possible for something to be hidden while it remains in plain sight; you merely keep the person from looking at it. We will use the words hidden, inaudible, imperceivable, and invisible to mean that an observer does not notice the presence of the data, even if they are perceptible.)
2. The embedded data should be directly encoded into the media, rather than into a header or wrapper, so that the data remain intact across varying data file formats.
3. The embedded data should be immune to modifications ranging from intentional and intelligent attempts at removal to anticipated manipulations, e.g., channel noise, filtering, resampling, cropping, encoding, lossy compressing, printing and scanning, digital-to-analog (D/A) conversion, and analog-to-digital (A/D) conversion, etc.
4. Asymmetrical coding of the embedded data is desirable, since the purpose of data hiding is to keep the data in the host signal, but not necessarily to make the data difficult to access.
5. Error correction coding should be used to ensure data integrity. It is inevitable that there will be some degradation to the embedded data when the host signal is modified.
6. The embedded data should be self-clocking or arbitrarily re-entrant. This ensures that the embedded data can be recovered when only fragments of the host signal are available, e.g., if a sound bite is extracted from an interview, data embedded in the audio segment can be recovered. This feature also facilitates automatic decoding of the hidden data, since there is no need to refer to the original host signal.

Applications Trade-offs exist between the quantity of embedded data and the degree of immunity to host signal modification. By constraining the degree of host signal degradation, a data-hiding method can operate with either high embedded data rate, or high resistance to modification, but not both. As one increases, the other must decrease. While this can be shown mathematically for some data-hiding systems such as a spread spectrum, it seems to hold true for all data-hiding systems. In any system, you can trade bandwidth for

robustness by exploiting redundancy. The quantity of embedded data and the degree of host signal modification vary from application to application.

Consequently, different techniques are employed for different applications. Several prospective applications of data hiding are discussed in this section. An application that requires a minimal amount of embedded data is the placement of a digital water mark. The embedded data are used to place an indication of ownership in the host signal, serving the same purpose as an author's signature or a company logo. Since the information is of a critical nature and the signal may face intelligent and intentional attempts to destroy or remove it, the coding techniques used must be immune to a wide variety of possible modifications. A second application for data hiding is tamper-proofing. It is used to indicate that the host signal has been modified from its authored state. Modification to the embedded data indicates that the host signal has been changed in some way. A third application, feature location, requires more data to be embedded. In this application, the embedded data are hidden in specific locations within an image. It enables one to identify individual content features, e.g., the name of the person on the left versus the right side of an image. Typically, feature location data are not subject to intentional removal. However, it is expected that the host signal might be subjected to a certain degree of modification, e.g., images are routinely modified by scaling, cropping, and tone-scale enhancement. As a result, feature location data-hiding techniques must be immune to geometrical and nongeometrical modifications of a host signal.

2. SYSTEM ANALYSIS

Steganography is the process of hiding a secret audio/video/text within a larger one in such a way that someone cannot know the presence or contents of the hidden audio/video/text. Steganography is, many times, confused with cryptography as both the techniques are used to secure information. The difference lies in the fact that steganography hides the data so that nothing appears out of ordinary while cryptography encrypts the text, making it difficult for an outsider to infer anything from it even if they do attain the encrypted text. Both of them are combined to increase the security against various malicious attacks. The purpose of Steganography is to maintain secret communication between two parties. Using the LSB technique, which facilitates plain text hiding in an image as well as hiding files in an image. It works with JPEG and PNG formats for the cover image and always creates PNG Stego image due to its lossless compression. Least Significant Bit Embeddings (LSB) are a general steganographic technique that may be employed to embed data into a variety of digital media, the most studied applications are using LSB embedding to hide one image inside another. In this image steganography software, we can hide the data using LSB embed techniques.

2.1 Software Requirements Specification:

A software Requirements Specification (SRS) is a document that describes what the software will do and how it will be expected to perform it. It also describes the functionality of the product needs to fulfil all stakeholders needs. A typical SRS document includes the purpose of the project, overall description, and specific requirements to build the software.

2.1.1 Problem Statement:

How can we send a message secretly to the destination with securely? In this study, we proposed a new framework of an image steganography system to hide a digital text of secret message.

2.1.2 Proposed System:

This proposed system provides the user with two options: Encrypt and Decrypt.

- In encryption, we provide secret information and that secret information will be hidden in image file.
- Decryption is getting the hidden information from the image file.

2.1.3 Software Requirements :

For developing the application the following are the software requirements:

- Operating System: Linux or Windows 7 or above versions.
- User Interface: Command Interface.
- Programming Language: PYTHON3.

2.1.4 Hardware Requirements :

For developing this project we have used the following hardware requirements:

- Processor: Intel i3 or higher versions
- RAM: Minimum 2GB
- Space on Hard Disk: Minimum 5GB.

2.1.5 Software Tools Used:

Steganography Types:

As it is known there is much communication between people and organizations through the use of the phone, the fax, computer communications, radio, and of course all of these communications should be secure. There are basically three Steganography types: -

- Pure Steganography.
- Secret key Steganography.
- Public key Steganography.

2.1.5.1 Pure Steganography

Pure Steganography is a Steganography system that doesn't require prior exchange of some secret information before sending message; therefore, no information is required to

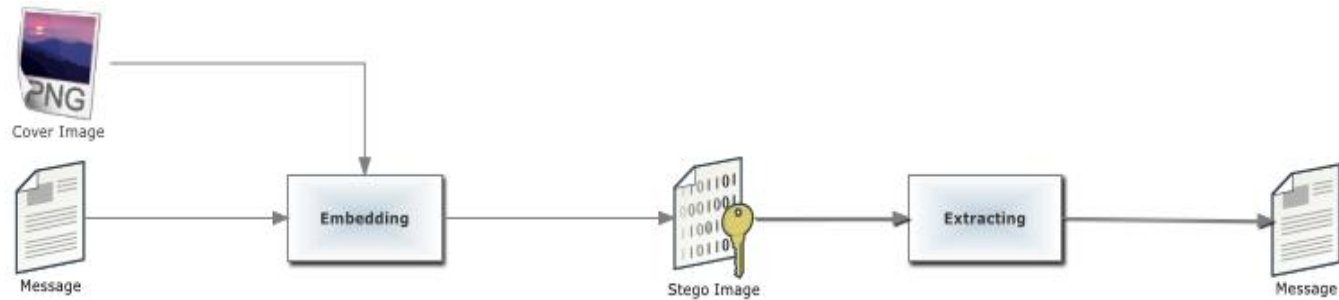


Figure 1-1: Pure Steganography

Figure 1: Pure Steganography

start the

communication process: the security of the system thus depends entirely on its secrecy.

In most applications, pure Steganography is preferred, since no stego-key must be shared between the communication partners, although a pure Steganography protocols don't provide any security if an attacker knows the embedding method.

2.1.5.2 Secret Key Steganography

A secret key Steganography system is similar to a symmetric cipher, where the sender chooses a cover and embeds the secret message into the cover using a secret key. If the secret key used in the embedding process is known to the receiver, he can reverse the process and extract the secret message.

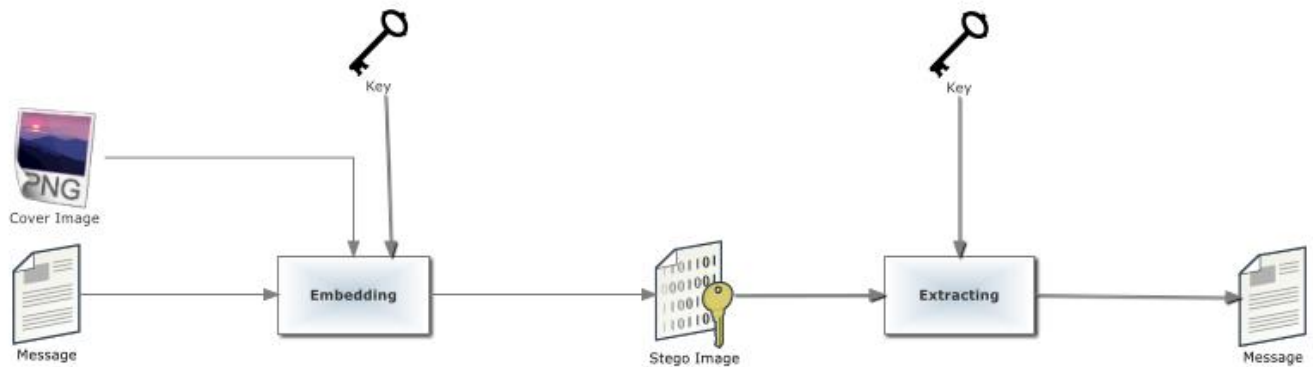


Figure 1-2: Secret Key Steganography

Figure 2: Secret Key Steganography

Anyone who doesn't know the secret key should not be able to obtain evidence of the encoded information.

2.1.5.3 Public Key Steganography

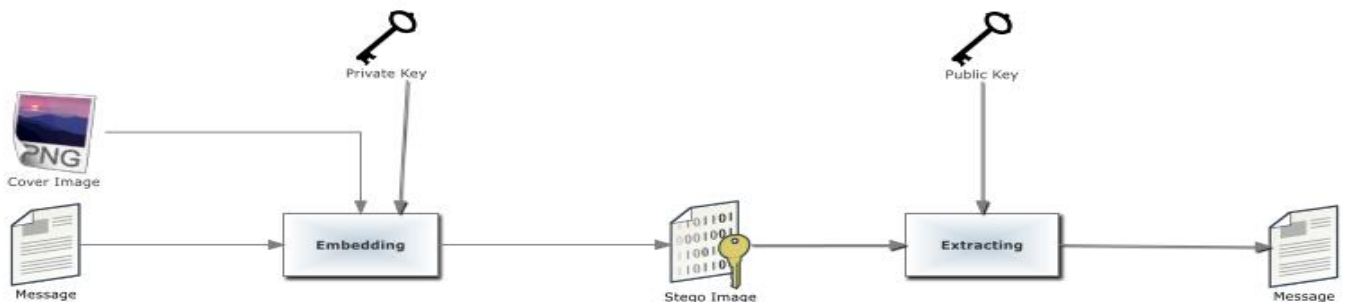


Figure 1-3: Public Key Steganography

Figure 3: Public Key Steganography

Public key Steganography does not depend on the exchange of a secret key. It requires two keys, one of them private (secret) and the other public: the public key is stored in a

public database, whereas the public key is used in the embedding process. The secret key is used to reconstruct the secret message

2.1.6 Steganography Algorithms

For encryption and decryption of text messages using the secret keys steganographic system uses algorithms known as steganographic algorithms. The mostly used algorithms for embedding data into images are:

- LSB (Least Significant Bit) Algorithm (Our domain)
- JSteg Algorithm
- F5 Algorithm

2.1.6.1 LSB Algorithm

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.

The advantage of LSB embedding is that it has good embedding capacity and the change is usually visually undetectable to the human eye. If all the coefficients are used, it can provide a capacity of almost one bit per coefficients using the frequency domain technique. On the other hand, it can provide a greater capacity for the spatial domain embedding with almost 1 bit per pixel for each color component. However, sending a raw image such as a Bitmap (BMP) to the receiver would create suspicion in and of itself, unless the image file is very small. Fridrich et al. proposed a steganalysis method which provides a high detection rate for shorter hidden messages [2]. Westfeld and Pfitzmann proposed another steganalysis algorithm for BMP images where the message length is comparable to the pixel count. Most of the popular formats today are compressed in the frequency domain and therefore it is not a common practice to embed bits directly in the spatial domain. Hence, frequency domain embeddings are the preferred choice for image steganography



Figure 4: Input and Output images

2.1.7 What is Python?

Python is a high-level scripting language which can be used for a wide variety of text processing, system administration and internet-related tasks. Unlike many similar languages, its core language is very small and easy to master, while allowing the addition of modules to perform a virtually limitless variety of tasks. Python is a true object-oriented language, and is available on a wide variety of platforms. There's even a python interpreter written entirely in Java, further enhancing python's position as an excellent solution for internet-based problems. Python was developed in the early 1990's by Guido van Rossum, then at CWI in Amsterdam, and currently at CNRI in Virginia. In some ways, python grew out of a project to design a computer language which would be easy for beginners to learn, yet would be powerful enough for even advanced users. This heritage is reflected in python's small, clean syntax and the thoroughness of the implementation of ideas like object-oriented programming, without eliminating the ability to program in a more traditional style. So python is an excellent choice as a first programming language without sacrificing the power and advanced capabilities that users will eventually need. Although pictures of snakes often appear on python books and websites, the name is derived from Guido van Rossum's favorite TV show, "Monty Python's Flying Circus". For this reason, lots of online and print documentation for the language has a light and humorous touch. Interestingly, many experienced programmers report that python has brought back a lot of

the fun they used to have programming, so van Rossum's inspiration may be well expressed in the language itself.

2.1.7.1 The very Basics of Python

There are a few features of python which are different than other programming languages, and which should be mentioned early on so that subsequent examples don't seem confusing. Further information on all of these features will be provided later, when the topics are covered in depth. Python statements do not need to end with a special character – the python interpreter knows that you are done with an individual statement by the presence of a newline, which will be generated when you press the "Return" key of your keyboard. If a statement spans more than one line, the safest course of action is to use a backslash (\) at the end of the line to let python know that you are going to continue the statement on the next line; you can continue using backslashes on additional continuation lines. (There are situations where the backslashes are not needed which will be discussed later.)

Python provides you with a certain level of freedom when composing a program, but there are some rules which must always be obeyed. One of these rules, which some people find very surprising, is that python uses indentation (that is, the amount of white space before the statement itself) to indicate the presence of loops, instead of using delimiters like curly braces ({}) or keywords (like "begin" and "end") as in many other languages.

The amount of indentation you use is not important, but it must be consistent within a given depth of a loop, and statements which are not indented must begin in the first column. Most python programmers prefer to use an editor like emacs, which automatically provides consistent indentation; you will probably find it easier to maintain your programs if you use consistent indentation in every loop, at all depths, and an intelligent editor is very useful in achieving this.

2.1.7.2 Basic Principles of Python

Python has many features that usually are found only in languages which are much more complex to learn and use. These features were designed into python from its very first beginnings, rather than being accumulated into an end result, as is the case with many other scripting languages. If you're new to programming, even the basic descriptions which follow may seem intimidating. But don't worry – all of these ideas will be made clearer in the chapters which follow. The idea of presenting these concepts now is to make you aware of how python works, and the general philosophy behind python programming. If some of the concepts that are introduced here seem abstract or overly complex, just try to get a general feel for the idea, and the details will be fleshed out later.

2.1.7.3 Invoking Python

There are three ways to invoke python, each with its' own uses. The first way is to type "python" at the shell command prompt. This brings up the python interpreter with a message similar to this one:

```
Python 2.2.1 (#2, Aug 27 2002, 09:01:47)
[GCC 2.95.4 20011002 (Debian prerelease)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
```

The three greater-than signs (>>>) represent python's prompt; you type your commands after the prompt, and hit return for python to execute them. If you've typed an executable statement, python will execute it immediately and display the results of the statement on the screen. For example, if I use python's print statement to print the famous "Hello, world" greeting, I'll immediately see a response:

```
>>> print 'hello,world'
hello,world
```

The print statement automatically adds a newline at the end of the printed string. This is true regardless of how python is invoked. (You can suppress the newline by following the string to be printed with a comma.) When using the python interpreter this way, it executes statements immediately, and, unless the value of an expression is assigned to a variable (See Section 6.1), python will display the value of that expression as soon as it's typed. This makes python a very handy calculator:

```
>>> cost = 27.00
>>> taxrate = .075
>>> cost * taxrate 2.025
>>> 16 + 25 + 92 * 3 317
```

When you use python interactively and wish to use a loop, you must, as always, indent the body of the loop consistently when you type your statements. Python can't execute your statements until the completion of the loop, and as a reminder, it changes its prompt from greater-than signs to periods. Here's a trivial loop that prints each letter of a word on a separate line — notice the change in the prompt, and that python doesn't respond until you enter a completely blank line.

2.1.7.4 Basic Core Language

Python is designed so that there really isn't that much to learn in the basic language. For example, there is only one basic structure for conditional programming (if/else/elif), two looping commands (while and for), and a consistent method of handling errors (try/except) which apply to all python programs. This doesn't mean that the language is not flexible and powerful, however. It simply means that you're not confronted with an overwhelming choice of options at every turn, which can make programming a much simpler task.

2.1.7.5 Modules

Python relies on modules, that is, self-contained programs which define a variety of functions and data types, that you can call in order to do tasks beyond the scope of the basic core language by using the import command. For example, the core distribution of python contains modules for processing files, accessing your computer's operating system and the internet, writing CGI scripts (which handle communicating with pages displayed in web browsers), string handling and many other tasks. Optional modules, available on the Python web site (<http://www.python.org>), can be used to create graphical user interfaces, communicate with data bases, process image files, and so on.

This structure makes it easy to get started with python, learning specific skills only as you need them, as well as making python run more efficiently by not always including every capability in every program.

2.1.7.6 PIL(PYTHON IMAGING LIBRARY)

The Python Imaging Library adds image processing capabilities to your Python interpreter. This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities. The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool. Let's look at a few possible uses of this library.

2.1.7.7 Image Archives

The Python Imaging Library is ideal for image archival and batch processing applications. You can use the library to create thumbnails, convert between file formats, print images, etc. The current version identifies and reads a large number of formats. Write support is intentionally restricted to the most commonly used interchange and presentation formats.

2.1.7.8 Image Display

The current release includes Tk PhotoImage and BitmapImage interfaces, as well as a Windows DIB interface that can be used with PythonWin and other Windows-based toolkits. Many other GUI toolkits come with some kind of PIL support. For debugging, there's also a show() method which saves an image to disk, and calls an external display utility.

2.2 Image Processing

The library contains basic image processing functionality, including point operations, filtering with a set of built-in convolution kernels, and colour space conversions. The library also supports image resizing, rotation and arbitrary affine transforms. There's a histogram method allowing you to pull some statistics out of an image. This can be used for automatic contrast enhancement, and for global statistical analysis.

2.2.1 Using the Image class

The most important class in the Python Imaging Library is the Image class, defined in the module with the same name. You can create instances of this class in several ways; either by loading images from files, processing other images, or creating images from scratch. To load an image from a file, use the `open()` function in the Image module:

```
>>> from PIL import Image
>>> im = Image.open("hopper.ppm")
```

If successful, this function returns an Image object. You can now use instance attributes to examine the file contents.

```
>>> print(im.format, im.size, im.mode)
PPM (512, 512) RGB
```

The `format` attribute identifies the source of an image. If the image was not read from a file, it is set to `None`. The `size` attribute is a 2-tuple containing width and height (in pixels). The `mode` attribute defines the number and names of the bands in the image, and also the pixel type and depth. Common modes are “L” (luminance) for greyscale images, “RGB” for true color images, and “CMYK” for pre-press images. If the file cannot be opened, an `IOError` exception is raised. Once you have an instance of the Image class, you can use the methods defined by this class to process and manipulate the image. For example, let's display the image we just loaded:

```
>>> im.show()
```

Note: The standard version of `show()` is not very efficient, since it saves the image to a temporary file and calls a utility to display the image. If you don't have an appropriate utility installed, it won't even work. When it does work though, it is very handy for

debugging and tests. The following sections provide an overview of the different functions provided in this library.

2.2.2 Reading and Writing Images

The Python Imaging Library supports a wide variety of image file formats. To read files from disk, use the `open()` function in the `Image` module. You don't have to know the file format to open a file. The library automatically determines the format based on the contents of the file. To save a file, use the `save()` method of the `Image` class. When saving files, the name becomes important. Unless you specify the format, the library uses the filename extension to discover which file storage format to use.

The `merge` function and many more takes a mode and a tuple of images, and combines them into a new image. The following sample swaps the three bands of an RGB image:

```
>> Splitting and merging bands
>> Geometrical transforms
>> Simple geometry transforms
>> Transposing an image
>> Color transforms
>> Converting between modes
>> Image enhancement
>> Filters
>> Applying filters
>> Point Operations
>> Applying point transforms
>> Processing individual bands
>> Enhancement
>> Enhancing images
>> Image sequences
>> Reading sequenc
```

3. SYSTEM DESIGN

System design is the process of defining the components, modules, interface and data for a system to satisfy specified requirements. It is the process of creating or altering systems along with the process, practices, models and methodologies used to develop them. System Design is the process of designing architecture of the developing software application. The purpose of System Design is to provide sufficient detailed data and information about the system and its elements to enable the consistent with architectural entities as defined in modules and views of the system architecture.

3.1 Data Flow Diagram:

Data Flow diagrams are used to graphically represent the flow of data in a business information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation. DFD graphically representing the functions, or processes, which capture, manipulate, store, and distribute data between a system and its environment and between components of a system. The visual representation makes it a good communication tool between User and System designer. Structure of DFD allows starting from a broad overview and expand it to a hierarchy of detailed diagrams.

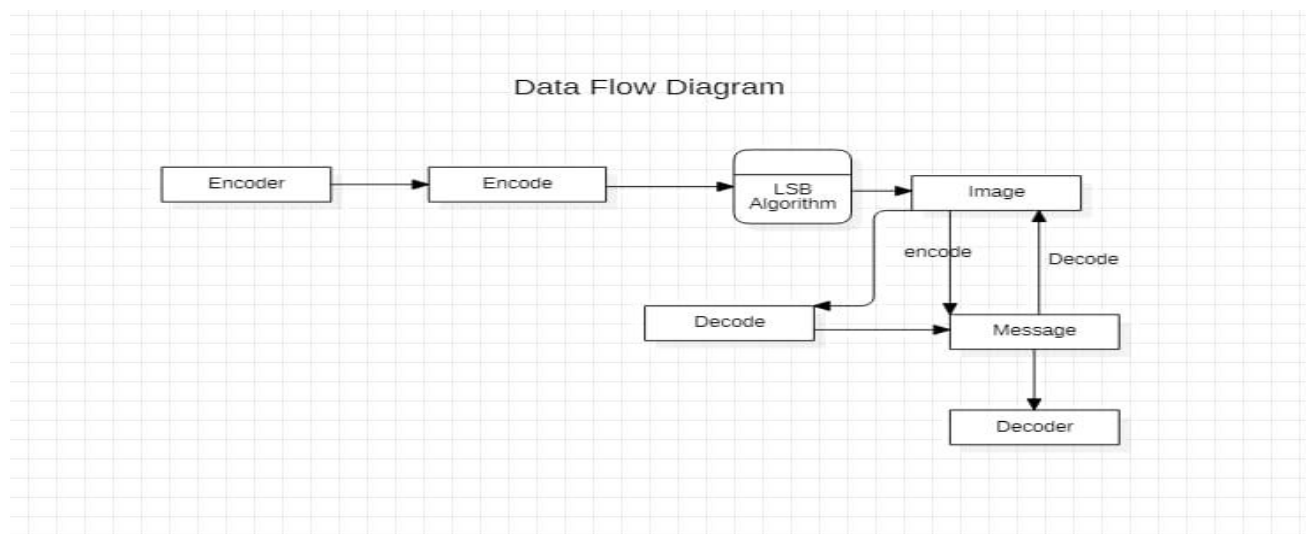


Figure 5: Data Flow Diagram

This figure shows DFD of Steganography. Flow starts with Encoder where it encodes the message using LSB algorithm through images which are hiding the data. Through Encoders and decoders message will be encoded and decoded.

3.2 UML Diagrams:

Unified Modelling Language (UML) is a general purpose modelling language. The main aim of UML is to define a standard way to visualize the way a system has been designed. It is quite similar to blueprints used in other fields of engineering. UML is not a programming language, it is rather a visual language. We use UML diagrams to portray the behaviour and structure of a system.

UML helps software engineers, businessmen and system architects with modelling, design and analysis. UML is linked with object oriented design and analysis. UML makes the use of elements and forms associations between them to form diagrams.

3.2.1 Use Case Diagram:

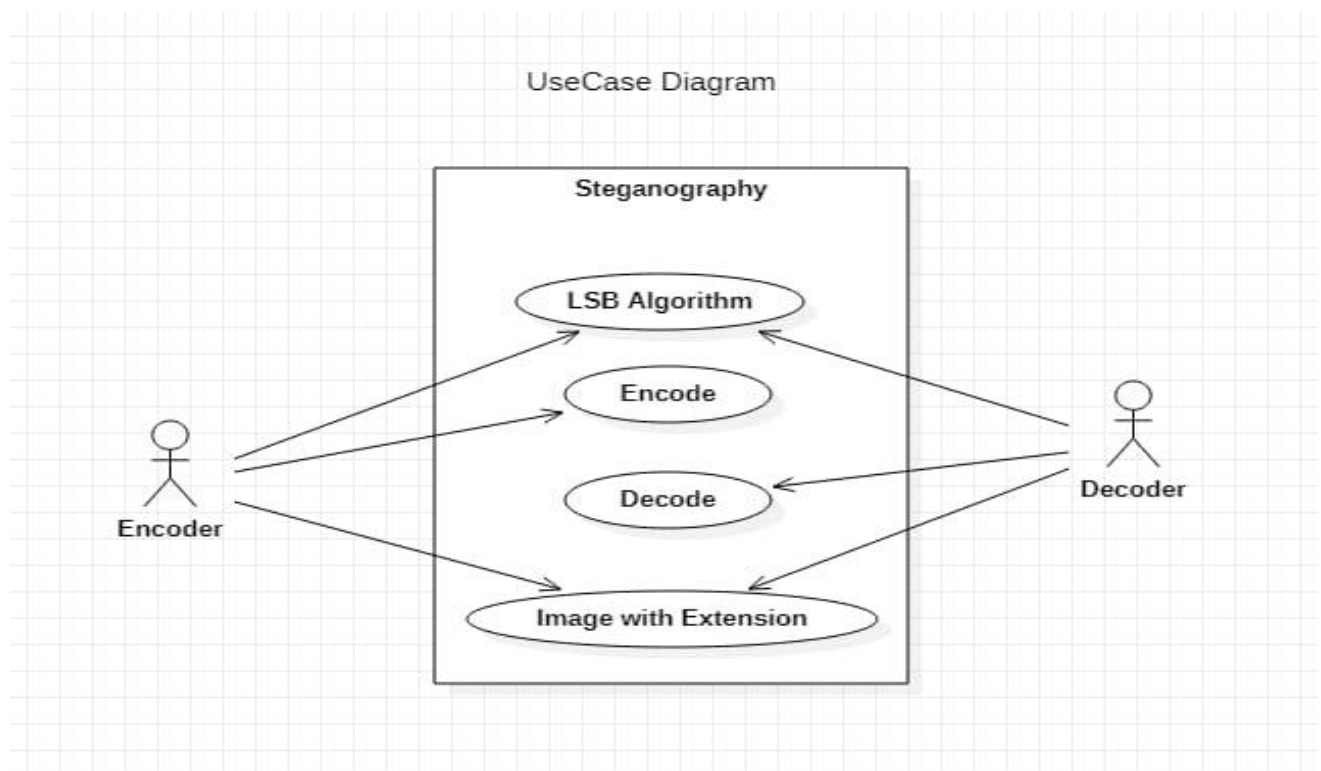


Figure 6: Use Case Diagram

The following use case diagram has two actors Encoder and Decoder who can perform actions such as Encoding and decoding and interacts using LSB algorithm.

3.2.2 Sequence Diagram:

Sequence Diagram are an easy and intuitive way of describing the behaviour of a system by viewing the interaction between the system and the environment. A sequence diagram shows an interaction arranged in a time sequence. A sequence diagram has two dimensions: vertical dimension represents time; the horizontal dimension represents the objects existence during the interaction.

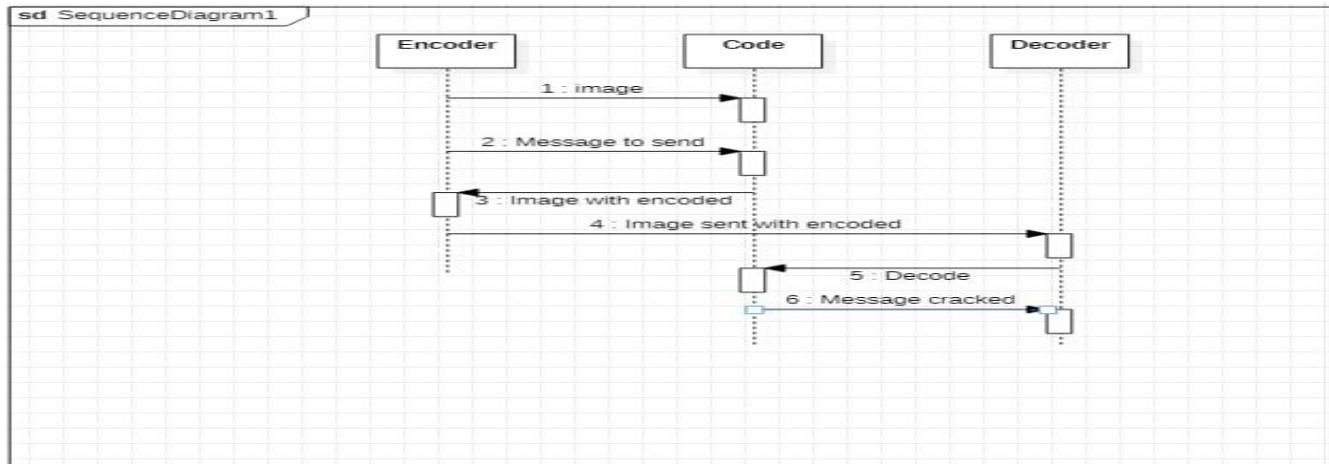


Figure 7: Sequence Diagram

This above figure depicts sequence actions of Steganography where encoder tries to encode the message hidden in images and images gets encoded sent to decoder to get decoded.

3.2.3 Class Diagram:

Class Diagram is a static diagram. It represents the static view of an application. It is used for visualizing, describing, and documenting different aspects of a system. It describes the attributes and operations of a class and also the constraints imposed on the system.

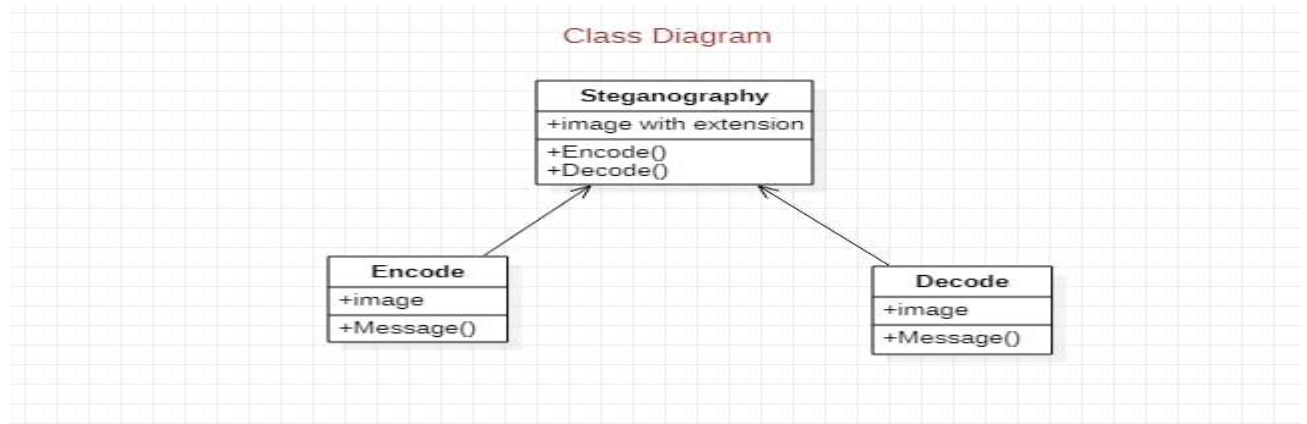


Figure 8: Class Diagram

3.2.4 State Chat Diagram:

State Chat Diagram is one of the five UML diagrams used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. State Chart Diagrams are useful to model the reactive systems. Reactive systems can be defined as a system that responds to external or internals events. State chart Diagrams describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of State chart diagram is to model lifetime of an object from creation to termination. State Chart diagrams are also used for forward and reverse engineering of a system. However, the main purpose is to model the reactive system.

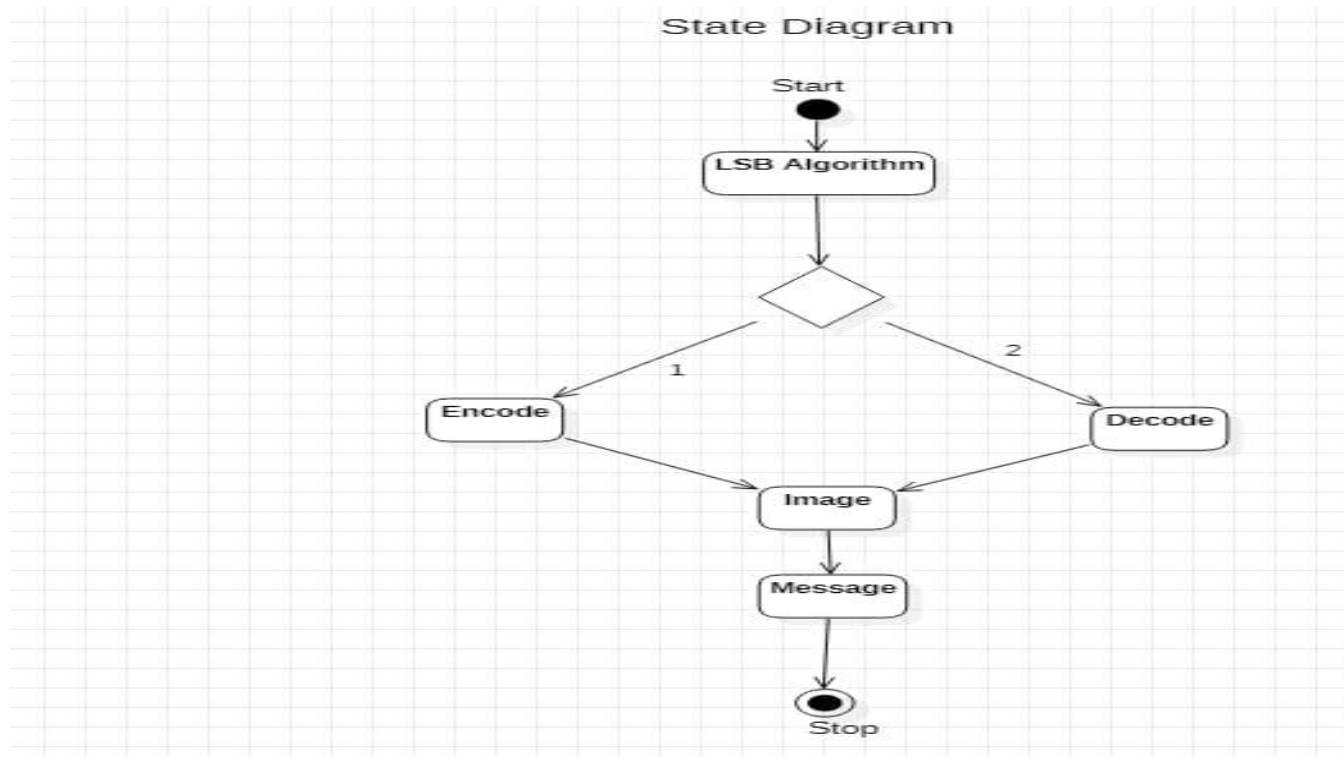


Figure 9: State Chat Diagram

This state diagram depicts and starts with the LSB algorithm where it has two options having Encoding and Decoding methods using an image to encode and decode the messages hidden and encompoused on the images.

4. SYSYTEM IMPLEMENTATION

4.1 Hiding text inside an image using Python:

We will build an example to demonstrate the encryption of data into images. The data is represented using its ASCII value. Each letter is represented using 8bits, or 1 byte. Each pixel in the image has three bits, one each for RGB frames. Each pixel can hold 3 bits. We will encode three pixels together, making a place for 9 bits. Out of the 9 bits, we will store 1 letter in the 8 bits and one (End Of File)EOF bit. If the EOF bit is high, then it indicates that the end of the message has been reached. Otherwise, it indicates the program to read more pixels for decoding.

We have a simple encryption technique included in the algorithm above. Instead of encoding 3 letters together, we can encode any other number of letters. Unless anyone knows this number, decrypting the file is going to be tough.

4.2 Encode the data :

Every byte of data is converted to its 8-bit binary code using ASCII values. Now pixels are read from left to right in a group of 3 containing a total of 9 values. The first 8-values are used to store binary data. The value is made odd if 1 occurs and even if 0 occurs.

For

Example:

:

Suppose the message to be hidden is 'Hii'. Since the message is of 3-bytes, therefore, pixels required to encode the data is $3 \times 3 = 9$. Consider a 4 x 3 image with a total 12-pixels, which are sufficient to encode the given data.

```
[(27, 64, 164), (248, 244, 194), (174, 246, 250), (149, 95, 232)
(188, 156, 169), (71, 167, 127), (132, 173, 97), (113, 69, 206)
(255, 29, 213), (53, 153, 220), (246, 225, 229), (142, 82, 175)]
```

20

ASCII value of 'H' is 72 whose binary equivalent is 01001000.

Taking first 3-pixels (27, 64, 164), (248, 244, 194), (174, 246, 250) to encode. Now change

the pixel to odd for 1 and even for 0. So, the modifies pixels are (26, 63, 164), (248, 243, 194), (174, 246, 250). Since we have to encode more data, therefore, the last value should be even. Similarly, 'i' can be encoded in this image.

The new image will look like :

```
[ (26, 63, 164), (248, 243, 194), (174, 246, 250), (148, 95, 231),  
(188, 155, 168), (70, 167, 126), (132, 173, 97), (112, 69, 206),  
(254, 29, 213), (53, 153, 220), (246, 225, 229), (142, 82, 175) ]
```

4.3 Decode the data :



To decode, three pixels are read at a time, till the last value is odd, which means the message is over. Every 3-pixels contain a binary data, which can be extracted by the same encoding logic. If the value is odd the binary bit is 1 else 0.

4.4 Code in python:

Python program implementing Image Steganography

PIL module is used to extract

pixels of image and modify it

from PIL import Image

Convert encoding data into 8-bit binary

form using ASCII value of characters

def genData(data):

list of binary codes

of given data

newd = []

for i in data:

newd.append(format(ord(i), '08b'))

return newd

Pixels are modified according to the

8-bit binary data and finally returned

def modPix(pix, data):

datalist = genData(data)

lendata = len(datalist)

```

imdata = iter(pix)

for i in range(lendata):

    # Extracting 3 pixels at a time
    pix = [value for value in imdata.__next__():3] +

imdata.__next__():3] +
imdata.__next__():3]]

# Pixel value should be made

# odd for 1 and even for 0

for j in range(0, 8):

    if (datalist[i][j] == '0' and pix[j]% 2 != 0):

        pix[j] -= 1

    elif (datalist[i][j] == '1' and pix[j] % 2 == 0):

        if(pix[j] != 0):

            pix[j] -= 1

        else:

            pix[j] += 1

    #pix[j] -= 1

```

```

# Eighth pixel of every set tells

# whether to stop or read further.

# 0 means keep reading; 1 means the
# message is over.

if (i == lendata - 1):

if (pix[-1] % 2 == 0):

if (pix[-1] != 0):

pix[-1] -= 1

23

else:

pix[-1] += 1

else:

if (pix[-1] % 2 != 0):

pix[-1] -= 1

pix = tuple(pix)

yield pix[0:3]

yield pix[3:6]

yield pix[6:9]

def encode_enc(newimg, data):

w = newimg.size[0]

(x, y) = (0, 0)

```



```

for pixel in modPix(newimg.getdata(), data):

    # Putting modified pixels in the new image

    newimg.putpixel((x, y), pixel)

    if (x == w - 1):

        x = 0

        y += 1

    else:

        x += 1


# Encode data into image

def encode():

    img = input("Enter image name(with extension) : ")

    image = Image.open(img, 'r')

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

    if (len(data) == 0):

        raise ValueError('Data is empty')

    newimg = image.copy()

    encode_enc(newimg, data)

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

    newimg.save(new_img_name, str(new_img_name.split(".")[1].upper()))


# Decode the data in the image

```

```

def decode():

    img = input("Enter image name(with extension) : ")

    image = Image.open(img, 'r')

    data = ""

    imgdata = iter(image.getdata())

    while (True):

        pixels = [value for value in imgdata.__next__()[0:3] +

imgdata.__next__()[0:3] +

imgdata.__next__()[0:3]]

        # string of binary data

        binstr = ""

        for i in pixels[0:8]:

            if (i % 2 == 0):

                binstr += '0'

            else:

                binstr += '1'

        data += chr(int(binstr, 2))

        if (pixels[-1] % 2 != 0):

            return data

```

Main Function

```
def main():
```

```
    a = int(input(":: Welcome to Steganography ::\n"
```

```
                "1. Encode\n2. Decode\n"))
```

```
    if (a == 1):
```

```
        encode()
```

```
    elif (a == 2):
```

```
        print("Decoded Word : " + decode())
```

```
    else:
```

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

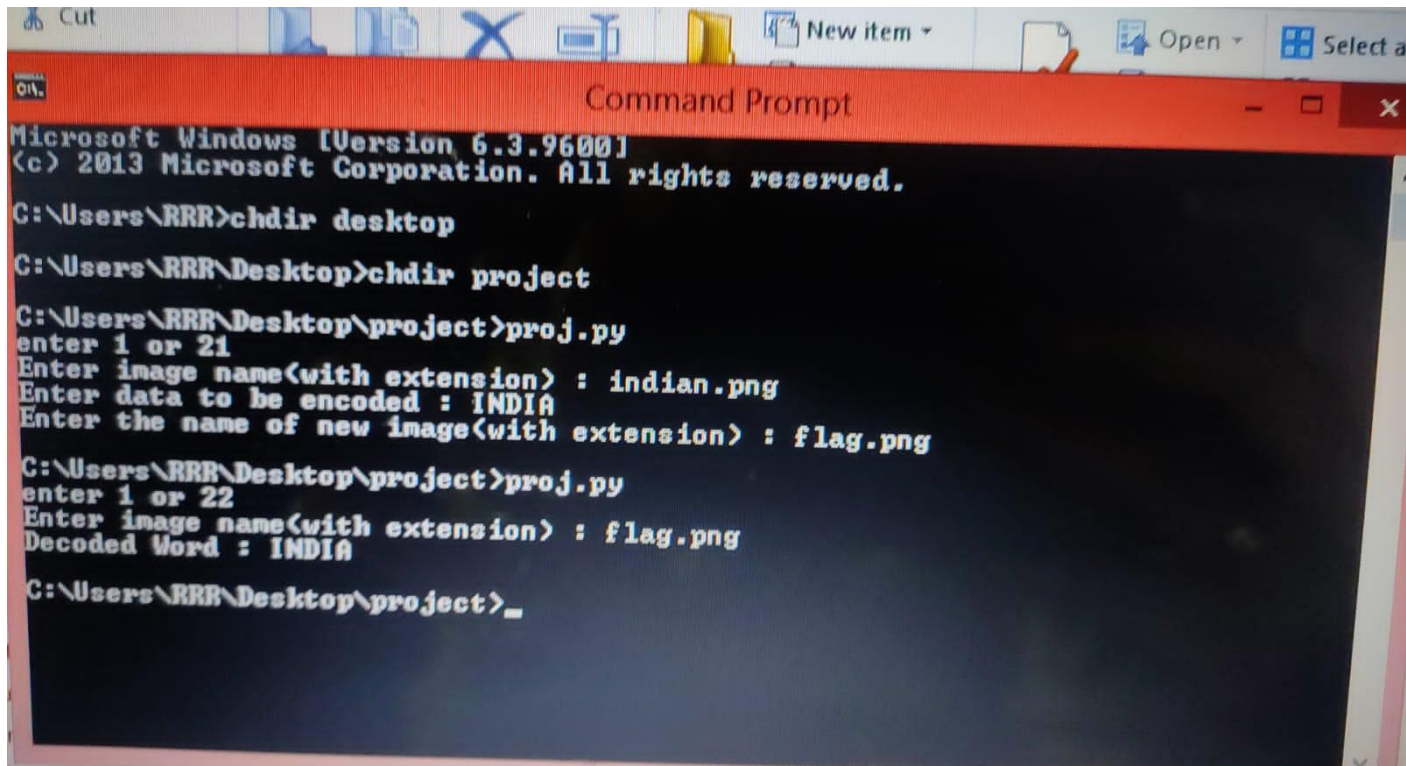
Driver Code

```
if __name__ == '__main__':
```

```
    # Calling main function
```

```
    main()
```

4.5 Output :



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\RRR>chdir desktop
C:\Users\RRR\Desktop>chdir project
C:\Users\RRR\Desktop\project>proj.py
enter 1 or 21
Enter image name(with extension) : indian.png
Enter data to be encoded : INDIA
Enter the name of new image(with extension) : flag.png
C:\Users\RRR\Desktop\project>proj.py
enter 1 or 22
Enter image name(with extension) : flag.png
Decoded Word : INDIA
C:\Users\RRR\Desktop\project>_
```

Figure 10: Output

5. SYSTEM TESTING

System testing also referred to as system-level tests or system integration testing, is the process in which a quality assurance(QA) team evaluates how the various components of an application interact together in the full, integrated system or application. System testing verifies that an application performance tasks as designed. This step, a kind of black box testing, focuses on the functionality of an application. System testing examines every component of an application to make sure that they work as a complete and unified whole. A QA team typically conducts system testing after it checks individual modules with functional or user story testing and then each component through integration testing.

5.1 Unit Testing:

In unit testing, various modules have been tested individually. This has been manually to test if expected result is actually seen on the screen. The following are test cases with the help of which the application has been tested

5.2 Performance Testing:

Performance testing is the practice of evaluating how a system performs in terms of responsiveness and stability under a particular workload. The performance of the Steganography application is measured as following:

- **Reliability:** Steganography application always produces the correct results irrespective of mobile device.
- **Maintenance:** The maintenance of Steganography application is very easy. Since, the code consists of only three modules, we can easily detect errors and bugs and fixes them easily.
- **Scalability:** Scalability is defined as increasing the potentiality of the software. Our application has worked well when we tested in another android devices. The application can be installed in many number of android devices (Android version: 5 or above).
- **Speed:** Steganography application works very fast and in smooth manner after installing the application. We kept the user interface very light with less number of modules.
- **Robustness:** Robustness testing is usually done to test the exceptional handling and quality of the software. So, when we give tasks list more than fifteen and stored in the firebase. Steganography application doesn't produce any false results.

- **Portability:** This application software is platform independent. Therefore, it can run on any mobile device whose version is greater than five and can be installed on iOS devices only

5.3 Compatibility Testing:

Compatibility testing is done to ensure the system functions in other devices or in other software environments without any error or malfunctions. Our Steganography functions very well similiarly on other devices.

6. CONCLUSION & FUTURE ENHANCEMENT

6.1 Conclusion:

This document gave an introduction about steganography and the meaning of hiding data; also, the problem research and the importance of Steganography in transmit the data over the network, for that reason in this document we propose a system to hiding and embedding the data inside images. Also, it provided what is the purpose and objectives of our project.

6.2 Future Scope:

1. The future work on this project is to improve the compression ratio of the image to text.
2. This project can be extended to a level such that it can be used for different types of image formats like .JPEG, .bmp, .tif etc,
3. The security using LSB algorithm is good, but we can improve the level to a certain extent by varying carriers, as well as using different keys for encryption and decryption.

7. REFERENCES

<https://matlabprojects.org/steganography-projects/>

2. https://www.researchgate.net/publication/314116270_Image_Steganography

<https://www.geeksforgeeks.org/image-based-steganography-using-python/amp/>

4. <https://www.m.mu.edu.sa/sites/default/files/content/2018/12/Steganography%20Using%20Images.pdf>

8. ANNEXURE

1. UDP – User Datagram Protocol.
2. ILT – Information and Communication Technology.
3. D/A – Digital to Analog.
4. A/D – Analog to Digital.
5. JPEG – Joint Photographic Experts Group.
6. LSB – Least Significant Bit.
7. PNG – Portable Network Graphics
8. XML Extensible Markup Language.
9. SRS – Software Requirement Specification.
10. CGI – Common Gateway Interface.
11. DIB – Data Interface Bus.
12. GUI – Graphical User Interface
13. EoF – End of File
14. DFD – Data Flow Diagram
15. UML – Unified Modeling Language.