

SECURING DATA WITH IMAGE ENCRYPTION

*Minor project report submitted
in partial fulfillment of the requirement for award of the degree of*

**Bachelor of Technology
in
Computer Science & Engineering**

By

C SAI KARTHIK	(20UECS0178)	(VTU16975)
T CHAKRADHAR	(20UECS0929)	(VTU15064)
P ANIRUDH	(20UECS1073)	(VTU21646)

*Under the guidance of
Dr .V. JEEVANATHAM,ME,PH.D.
ASSOCIATE PROFESSOR*



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF COMPUTING**

**VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF
SCIENCE & TECHNOLOGY**

(Deemed to be University Estd u/s 3 of UGC Act, 1956)

**Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA**

April, 2023

SECURING DATA WITH IMAGE ENCRYPTION

*Minor project report submitted
in partial fulfillment of the requirement for award of the degree of*

**Bachelor of Technology
in
Computer Science & Engineering**

By

**C SAI KARTHIK (20UECS0178) (VTU16975)
T CHAKRADHAR (20UECS0929) (VTU15064)
P ANIRUDH (20UECS1073) (VTU21646)**

*Under the guidance of
Dr.V.JEEVANATHAM,M.E,Ph.D.
ASSOCIATE PROFESSOR*



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF COMPUTING**

**VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF
SCIENCE & TECHNOLOGY**

(Deemed to be University Estd u/s 3 of UGC Act, 1956)

**Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA**

April, 2023

CERTIFICATE

It is certified that the work contained in the project report titled SECURING DATA WITH IMAGE ENCRYPTION by C.SAI KARTHIK (20UECS0178), T.CHAKRADHAR (20UECS0929), P.ANIRUDH (20UECS1073)” has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Signature of Supervisor

Dr.V.Jeevanatham

Associate professor

Computer Science & Engineering

School of Computing

Vel Tech Rangarajan Dr. Sagunthala R&D

Institute of Science & Technology

April, 2023

Signature of Head of the Department

Dr.M.S. Murali Dhar, M.E., Ph.D.

Computer Science & Engineering

School of Computing

Vel Tech Rangarajan Dr. Sagunthala R&D

Institute of Science & Technology

April, 2023

Signature of the Dean

Dr. V. Srinivasa Rao

Professor & Dean

Computer Science & Engineering

School of Computing

Vel Tech Rangarajan Dr. Sagunthala R&D

Institute of Science & Technology

April, 2023

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We 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)

C.SAI KARTHIK

Date: / /

(Signature)

T.CHAKRADHAR

Date: / /

(Signature)

P.ANIRUDH

Date: / /

APPROVAL SHEET

This project report entitled SECURING DATA WITH IMAGE ENCRPTION by C.SAI KARTHIK (20UECS0178), T.CHAKRADHAR (20UECS0929), P.ANIRUDH (20UECS1073) is approved for the degree of B.Tech in Computer Science & Engineering.

Examiners

Supervisor

Dr.V.JEEVANATHAM, M.E,Ph.D.,

Date: / /

Place:

ACKNOWLEDGEMENT

We express our deepest gratitude to our respected **Founder Chancellor and President Col. Prof. Dr. R. RANGARAJAN B.E. (EEE), B.E. (MECH), M.S (AUTO),D.Sc., Foundress President Dr. R. SAGUNTHALA RANGARAJAN M.B.B.S.** Chairperson Managing Trustee and Vice President.

We are very much grateful to our beloved **Vice Chancellor Prof. S. SALIVAHANAN**, for providing us with an environment to complete our project successfully.

We record indebtedness to our **Professor & Dean, Department of Computer Science & Engineering, School of Computing, Dr. V. SRINIVASA RAO, M.Tech., Ph.D.**, for immense care and encouragement towards us throughout the course of this project.

We are thankful to our **Head, Department of Computer Science & Engineering, Dr.M.S. MURALI DHAR, M.E., Ph.D.**, for providing immense support in all our endeavors.

We also take this opportunity to express a deep sense of gratitude to our Internal Supervisor **Dr.V.JEEVANATHAM,M.E,Ph.D.**, for his cordial support, valuable information and guidance, he helped us in completing this project through various stages.

A special thanks to our **Project Coordinators Mr. V. ASHOK KUMAR, M.Tech., Ms. C. SHYAMALA KUMARI, M.E.**, for their valuable guidance and support throughout the course of the project.

We thank our department faculty, supporting staff and friends for their help and guidance to complete this project.

C.SAI KARTHIK	(20UECS0178)
T.CHAKRADHAR	(20UECS0929)
P.ANIRUDH	(20UECS1073)

ABSTRACT

Steganography is the method of hiding the data where sharing of data is present, by hiding data in other data. A major issue for computer networks is to prevent important information from being disclosed to illegal users. Valuable multimedia content such as digital images, however, is vulnerable to unauthorized access while in storage and during transmission over a network. Daily everybody transfer different kinds of media such as images, texts, audios, videos, etc on Internet. There are many hackers who can hack our private data from our devices using many different hacking techniques. If one should want to send some confidential messages to other, there must be some technique for it. Encryption is a method of encrypting the text and images. Encrypted message can be decrypted using some key. Any hacker can find the key and decrypt it as he knows if he finds it is encrypted. We propose to build a secured image embedding algorithm that can be used to embed and send images to the intended receiver. LSB algorithm which is Steganography method used to embed the image for sending through the internet to the receiver which anyone can't find that it is embedded.

Keywords: Stego key, Encryption, Decryption, Image, Message, Steganography, LSB, Cyber Space.

LIST OF FIGURES

4.1	Architecture	11
4.2	Text Data Encryption	12
4.3	Interacting the System	13
4.4	Sender Receiver Interaction	14
4.5	Secret Text Image	15
4.6	Communication Diagram	16
4.7	System Flow Control	17
5.1	Choosing the Encode/Decode	22
5.2	Protecting the Data	23
5.3	Image selection	27
5.4	Test Image	30
6.1	Encode the data	36
6.2	Decrypting the data	37
8.1	Plagraism report	40
9.1	Poster	46

LIST OF ACRONYMS AND ABBREVIATIONS

ASCII	American Standard Code for Information Interchange
AES	Advanced Encryption Standard
BCES	Binary Count Encryption System
BER	Bit Error Rate
HMAC	Hash Message Authentication Code
LSB	Lest significant bit
RSA	Rivest Shamir Adlemen
RAM	Random acces memory
UML	Unified modeling language

TABLE OF CONTENTS

	Page.No
ABSTRACT	v
LIST OF ACRONYMS AND ABBREVIATIONS	vii
1 INTRODUCTION	1
1.1 Introduction	1
1.2 Aim of the project	1
1.3 Project Domain	2
1.4 Scope of the Project	2
2 LITERATURE REVIEW	3
3 PROJECT DESCRIPTION	8
3.1 Existing System	8
3.2 Proposed System	8
3.3 Feasibility Study	9
3.3.1 Economic Feasibility	9
3.3.2 Technical Feasibility	9
3.3.3 Social Feasibility	9
3.4 System Specification	10
3.4.1 Hardware Specification	10
3.4.2 Software Specification	10
3.4.3 Standards and Policies	10
4 METHODOLOGY	11
4.1 General Architecture	11
4.2 Design Phase	12
4.2.1 Data Flow Diagram	12
4.2.2 Use Case Diagram	13
4.2.3 Class Diagram	14
4.2.4 Sequence Diagram	15

4.2.5	Collaboration diagram	16
4.2.6	Activity Diagram	17
4.3	Algorithm & Pseudo Code	17
4.3.1	Algorithm	17
4.3.2	Pseudo Code	19
4.4	Module Description	20
4.4.1	Encryption Module	20
4.4.2	Decryption Module	20
4.4.3	Mapping Module	20
4.5	Steps to execute/run/implement the project	21
4.5.1	Execution	21
4.5.2	Run	21
4.5.3	Implementation	21
5	IMPLEMENTATION AND TESTING	22
5.1	Input and Output	22
5.1.1	Input Design	22
5.1.2	Output Design	23
5.2	Testing	23
5.3	Types of Testing	24
5.3.1	Unit testing	24
5.3.2	Integration testing	25
5.3.3	System testing	27
5.3.4	Test Result	30
6	RESULTS AND DISCUSSIONS	31
6.1	Efficiency of the Proposed System	31
6.2	Comparison of Existing and Proposed System	31
6.3	Sample Code	31
7	CONCLUSION AND FUTURE ENHANCEMENTS	38
7.1	Conclusion	38
7.2	Future Enhancements	38
8	PLAGIARISM REPORT	39

9	SOURCE CODE & POSTER PRESENTATION	41
9.1	Source Code	41
9.2	Poster Presentation	46
	References	46

Chapter 1

INTRODUCTION

1.1 Introduction

Today web is going towards the multimedia data in which image covers the highest percentage of it. But with the ever increasing growth of multimedia applications, security is an important aspect in communication and storage of images, and encryption is the way to ensure security. Image encryption techniques try to convert original image to another image that is hard to understand and to keep the image confidential between users, in other words, it's important that without decryption key no one can access the content. Image encryption has applications in internet communication, multimedia systems, medical imaging, telemedicine, military communication etc. The image encryption and decryption was done by using Steganography, it is a method of hiding secret data, by embedding it into an audio, video, image or text file.

Digital communication witnesses a clear and continuous development in many applications within the Internet. Hence, secure communication sessions must be provided. The security of data that is transmitted across a world wide network has become a key factor on the network performance measures. So, the confidentiality and integrity of transmitted data are needed to stop from accessing and using transmitted data. Steganography and Cryptography are 2 techniques that are provided for network security. The aim of this paper is to develop an approach to cover secret data in an image, by taking advantage of combination of cryptography and steganography.

1.2 Aim of the project

The main objective was to provide a image encryption mechanism which provides high security level. Less computational time and efficient way to deal with bulky, difficult and intractable data.

1.3 Project Domain

The project domain is mainly based on the networking and cybersecurity. We can hide the data in an Image for a secret communication.

1.4 Scope of the Project

The scope of the project is to limit unauthorized access and provide better security during message transmission.

Chapter 2

LITERATURE REVIEW

[1]Abdullah A Rashid, Khalid Ali Hussein. Image encryption algorithm based on the density and 6D logistic map. Inform. Sciences, 2023, 539: 198–215. year 2023] proposed Image encryption algorithm based on the density and 6D logistic map. One of the most difficult issues in the history of communication technology is the transmission of secure images. On the internet, photos are used and shared by millions of individuals for both private and business reasons. Utilizing encryption methods to change the original image into an unintelligible or scrambled version is one way to achieve safe image transfer over the network. Cryptographic approaches based on chaotic logistic theory provide several new and promising options for developing secure Image encryption methods. The main aim of this paper is to build a secure system for encrypting gray and color images. The proposed system consists of two stages, the first stage is the encryption process, in which the keys are generated depending on the chaotic logistic with the image density to encrypt the gray and color images, and the second stage is the decryption, which is the opposite of the encryption process to obtain the original image. The proposed method has been tested on two standard gray and color images publicly available. The test results indicate to the highest value of peak signal-to-noise ratio (PSNR), unified average changing intensity (UACI), number of pixel change rate (NPCR) are 7.7268, 50.2011 and 100, respectively. While the encryption and decryption speed up to 0.6319 and 0.5305 second respectively.

[2]Huda R Shakir, Sadiq A Mehdi, Anwar A Hattab. A new four-dimensional hyper-chaotic system for image encryption . Appl. Sof Comput. 94, 106164, year 2023. proposed A new four-dimensional hyper-chaotic system for image encryption. Currently, images are very important with the rapid growth of communication networks. Therefore, image encryption is a process to provide security for private information and prevent unwanted access to sensitive data by unauthorized individuals. Chaos systems provide an important role for key generation, with high random-

ization properties and accurate performance. In this study, a new four-dimensional hyper-chaotic system has been suggested that is used in the keys generation, which are utilized in the image encryption process to achieve permutation and substitution operations. Firstly, color bands are permuted using the index of the chaotic sequences to remove the high correlation among neighboring pixels. Secondly, dynamic S-boxes achieve the principle of substitution, which are utilized to diffuse the pixel values of the color image. The efficiency of the proposed method is tested by the key space, histogram, and so on. Security analysis shows that the proposed method for encrypting images is secure and resistant to different attacks. It contains a big key space of (2627) and a high sensitivity to a slight change in the secret key, a fairly uniform histogram, and entropy values nearby to the best value of 8. Moreover, it consumes a very short time for encryption and decryption.

[3]Noor Sattar Noor, Dalal Abdulmohsin Hammood, Ali Al-Naji, Javaan Chahl.A fast text-to-image encryption-decryption algorithm for secure network commu- nication Nonlinear Dyn. 94, 776–782 year 2022. proposed A fast text-to-image encryption-decryption algorithm for secure network communication. Data security is the science of protecting data in information technology, including authentication, data encryption, data decryption, data recovery, and user protection. To protect data from unauthorized disclosure and modification, a secure algorithm should be used. Many techniques have been proposed to encrypt text to an image. Most past studies used RGB layers to encrypt text to an image. In this paper, a Text-to-Image Encryption-Decryption (TTIED) algorithm based on Cyan, Magenta, Yellow, Key/Black (CMYK) mode is proposed to improve security, capacity, and processing time. The results show that the capacity increased from one to four times compared to RGB mode. Security was also improved due to a decrease in the probability of an adversary discovering keys. The processing time ranged between 0.001 ms (668 characters) and 31 s (25 million characters), depending on the length of the text. The compression rate for the encrypted file was decreased compared to WinRAR. In this study, Arabic and English texts were encrypted and decrypted.

[4]C. Xiuli, G. Zhihua, Y. Ke, et al., An image encryption scheme based on three-dimensional Brownian motion and chaotic system [J]. Chinese Physics B 26(2), 97–115,year 2020. Image encryption using chaotic maps.Images are widely used a data type for sharing information through the Internet. Designing and explor-

ing security mechanisms to protect the image data during communication are matters of great interest among researchers and industry experts. Encrypting the images is a commonly used mechanism of securing them from different types of attacks. In this work, various image encryption algorithms employing chaotic map techniques are compared. It includes various security parameters in respect of chaotic maps and encryption algorithms for images. Chaotic maps are verified using tools like bifurcation and Lyapunov diagrams. Image encryption algorithms need to satisfy a comprehensive set of security parameters and tests—visual analysis, key sensitivity, keyspace, correlation coefficient, and histogram analysis. Finally, a tabular comparison of different encryption schemes is done using ‘NPCR’ and ‘UACI.’ Another comparison provides the different schemes’ correlation coefficients for horizontal, vertical, and diagonal adjacencies.

[5]Luo YL, Zhou RL, Liu JX, Cao Y, Ding XM. A parallel image encryption algorithm based on the piecewise linear chaotic map and hyper-chaotic map. *Nonlinear r Dynam.*, 2020, 93(3): 1167–1184. proposed Double-image encryption method based on optical interference and logistic map. Investigated on as a generalization of the classical Fourier transform, introduced years ago in mathematics literature. The enhanced computation of fractional Fourier transform, the discrete version of FrFT came into existence DFrFT. This study of illustrates the advantage of discrete fractional Fourier transform (DFrFT) as compared to other transforms for steganography in image processing. The result shows same PSNR in both domain (time and frequency) but DFrFT gives an advantage of additional stego key. The order parameter of this transform. With the rapid development of internet and wide application of multimedia technology, people can communicate the digital multimedia information such as digital image, with others conveniently over the internet. In numerous cases, image data, transmitted over a network are expected not to be browsed or processed by illegal receivers. Consequently, the security of digital image has attracted much attention recently and many different methods for image encryption have been proposed, such as Optical systems are of growing interest for image encryption because of their distinct advantages of processing 2-dimensional complex data in parallel at high speed.

[6]G. S. Chandel, P. Patel, "A Review: Image Encryption with RSA and RGB ran- domized Histograms," *International Journal of Advanced Research*

in Computer and Communication Engineering (IJARCCCE), vol. 2, Issue 11, November 2020. proposed The stego-image quality has been improved by using bit-inversion technique. LSB method improving the PSNR of stegoimage. Through storing the bit patterns for which LSBs are inverted, image may be obtained correctly. For the improving the robustness of steganography, RC4 algorithm had been implemented to achieve the randomization in hiding message image bits into cover image pixels instead of storing them equentially. This method randomly disperses the bits of the message in the cover image and thus, harder for unauthorized people to extract the original message. The presented method shows good enhancement to Least Significant Bit technique in consideration to security as well as image quality. Among them the most widely used and highly successful optical encryption scheme is double random phase encoding. It can be shown that if these random phases are statistically independent white noise then the encrypted image is also a stationary white noise.

[7]Dr. P. Mahajan, A. Sachdeva, “A Study of Encryption Algorithms AES, DES and RSA for Security”, Global Journal of Computer Science and Technology Network, Web Security (GJCSTNWS), vo. 13 Issue. 15 Version 1.0 Year 2020.

proposed. This methodology provides an efficient and storage security mechanism for the protection of digital medical images. Authors proposed a viable steganography method using Integer Wavelet Transform to protect the MRI medical image into a single container image. The patient’s medical diagnosis image has been taken as secret image and Arnold transform was applied and scrambled secret image was obtained. In this case, the scrambled secret image was embedded into the dummy container image and Inverse IWT was taken to get a dummy secret image. It has been observed that the quality parameters are improved with acceptable PSNR compared to the existing algorithms. It can be shown that if these random phases are statistically independent white noise then the encrypted image is also a stationary white noise.

[8]L.Singh, Dr. R.K. Bharti, ”Comparative performance analysis of Cryptographic Algorithms,” International Journal of Advanced Research and Computer Science and Software Engineering (IJARCSSE), vol. 3, issue. 11, November 2021. implemented the optimal pixel change process has been applied after embedding the message. Authors employed the frequency domain to increase the robustness of our steganography method. Integer wavelet transform avoid the floating point precision problems of the wavelet filter. Result shows that the method outper-

forms adaptive steganography technique based on integer wavelet transform in terms of peak signal to noise ratio and capacity. Result shows very good Peak Signal to Noise Ratio, which is a measure of security. In this method the secret information is hidden in the middle bit-planes of the integer wavelet coefficients in high frequency sub-bands. Transform to protect the MRI medical image into a single container image. The patient's medical diagnosis image has been taken as secret image and Arnold transform was applied and scrambled secret image was obtained. In this case, the scrambled secret image was embedded into the dummy container image and Inverse IWT was taken to get a dummy secret image.

Chapter 3

PROJECT DESCRIPTION

3.1 Existing System

There are many existing systems which are using the RSA algorithm. RSA algorithm for this purpose user may submit his image encryption our system now gets the image and converts it into hex format before being encrypted. Then we use RSA algorithm to encrypt the image for sending through the internet user may now Sky select the intended user from among the list of users having our software installed. System sends the image in an encrypted format through an active internet connection. Even if the attacker gets the file he first has to decrypt using proper case which are not available to him. He then needs to decode the image into proper image format. When image reaches intended receiver, he must first enter required keys. On entering the keys, the software decrypt the entire image in original format and provide it to and receiver. Thus we encrypt and securely send images over the internet using secure RSA encryption, but by using RSA algorithm majorly most of the system security can be attacked by hacking by other user irrespective of the sender and receiver.

3.2 Proposed System

In this proposing system ,hiding the text data in an image using LSB steganography using a Steganokey which is only known to the sender and receiver if sender shares it with him. And also in our proposed system we can send the image from one device to another using their ipv4 address of the receiver and this is possible only if the systems are connected internet or any other network.

Advantages:

- 1.Highly secure as the secret pin is only known to sender and reciever.
- 2.Easy transmission over Internet using the Ipv4 address of the reciever.

3.3 Feasibility Study

The feasibility study is the must for any type of the project for the purpose of analysing it properly. There are mainly three important features which are included in the feasibility study.

They are:

1. Economic Feasibility.
2. Technical Feasibility.
3. Social Feasibility.

3.3.1 Economic Feasibility

This is an feasibility where the economic conditions will be checked with the users. We must be able to justify the expenditure and Research should be completed within the specified budget .which can be achieved because most of the new technologies are available freely for the usage of it.

3.3.2 Technical Feasibility

This feasibility is used for checking the technical requirements for the available system. This analysis will be replacing the old process where stealing of the data is possible. This is secured way and cracking that cannot be done easily. The technical resources available should not be much demand. It will be leading to increase the demand of the research in the market as well as within the client

3.3.3 Social Feasibility

This feasibility is used for verifying the system acceptance by the user. This process is helpful in making the user to adapt to the compatibility of the system efficiently .The user must be accepting all the requests and conditions as they are necessary for their process .Confidence level should be more to implement constructive ideas, which are accepted by the end-user of the system.

3.4 System Specification

3.4.1 Hardware Specification

- Pentium IV 2.4GHz
- Hard Disk-40GB
- RAM -256MB

3.4.2 Software Specification

- RAM -256MB
- Netbeans with apache
- Java JDK updated version Os

3.4.3 Standards and Policies

The purpose of this policy is to provide guidance that limits that use of encryption to those algorithms that have received substantial public review and has been proven to work efficiently. Additionally, this policy provides direction to ensure the federal regulations are followed and legal authority is granted for the dissemination and use of encryption technologies.

Chapter 4

METHODOLOGY

4.1 General Architecture

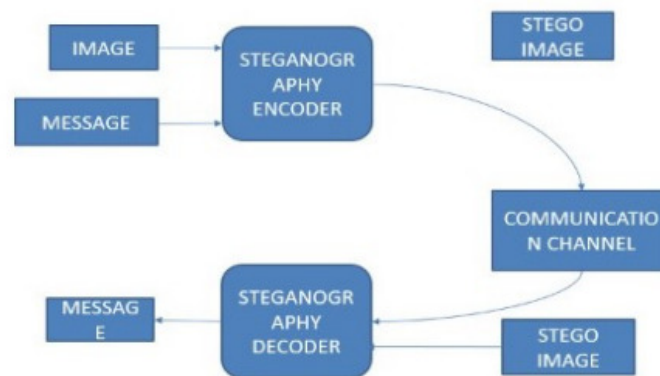


Figure 4.1: Architecture

In the architecture text message is embedded along with the image by the sender and encoding is done by using encoding algorithm so that the it is encrypted using steganokey know the encrypted image is sent to the receiver using IP address of a system which is connected to the Internet after receiving the image the receiver save the image and he decode the text from the image once the centre will stay connected to the receiver after the code in the actual message is shown to the receiver.

4.2 Design Phase

4.2.1 Data Flow Diagram



Figure 4.2: Text Data Encryption

In the text data encryption as can see the diagram we can understand the plane image is used to highlight the text data encrypted using this technique and then the stick no key is used to decrypt the encrypted image and receive reveals the text which is embedded in the image.

4.2.2 Use Case Diagram

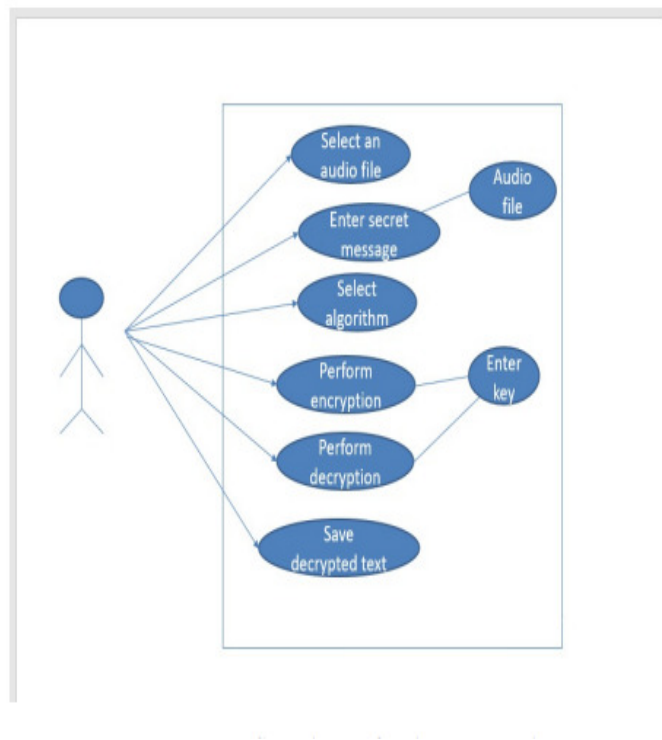


Figure 4.3: **Interacting the System**

In this use case diagram, the user selects a file to encrypt and chooses an encryption algorithm. The Encryption module encrypts the image using the selected algorithm and outputs the encrypted image. The user then saves the encrypted image to secure the data.

The encryption algorithm used can vary, but popular encryption algorithms include AES, Blowfish, and RSA. The use case diagram shows a simple flow of how encryption can be used to secure data with an image. However, it's important to note that proper encryption requires more than just choosing an algorithm and clicking a few buttons. It involves implementing appropriate security measures, including key management, data integrity checks, and access control, to ensure that the data remains secure.

4.2.3 Class Diagram

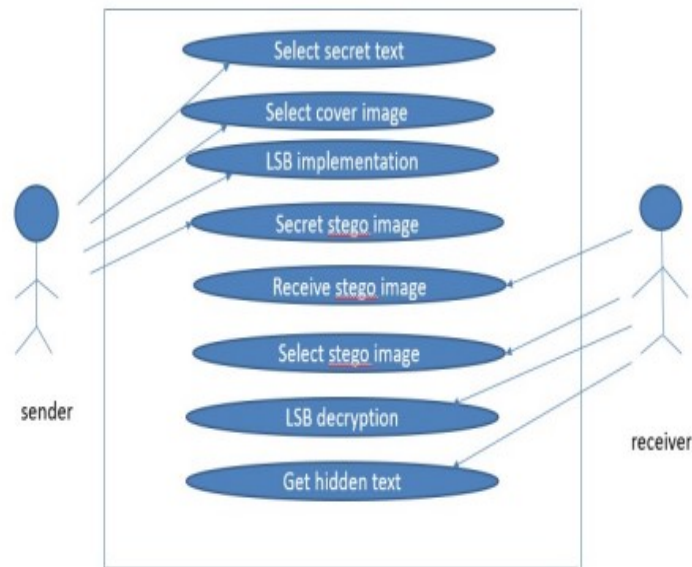


Figure 4.4: **Sender Receiver Interaction**

In class diagram as it simple is a representation of user interaction with the system first user right secret text then he selects cover image and data gets hidden image the user sends the go image to the receiver through image at the receiver side user selects the sticker image and a description on sticker image after that he can text hidden in the text.

4.2.4 Sequence Diagram

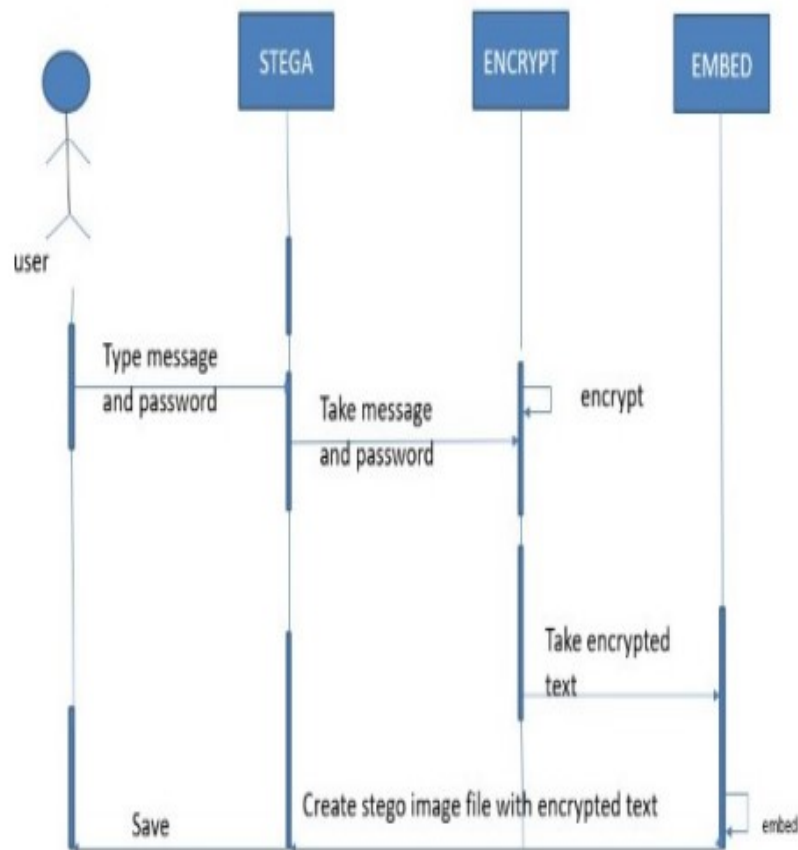


Figure 4.5: **Secret Text Image**

In secret text image In the case of image encryption, a sequence diagram can be used to illustrate the process of encrypting and decrypting an image using an encryption algorithm. The sequence diagram can show the flow of messages between the different objects involved in the process, such as the imageManager, the encryption-algorithm, and the encryptionkeymanagement.

The purpose of a sequence diagram is to provide a visual representation of the interactions between objects in a system. It can be used to communicate the design of the system to stakeholders, to aid in understanding of the system's behavior, and to identify potential issues or areas for improvement.

4.2.5 Collaboration diagram

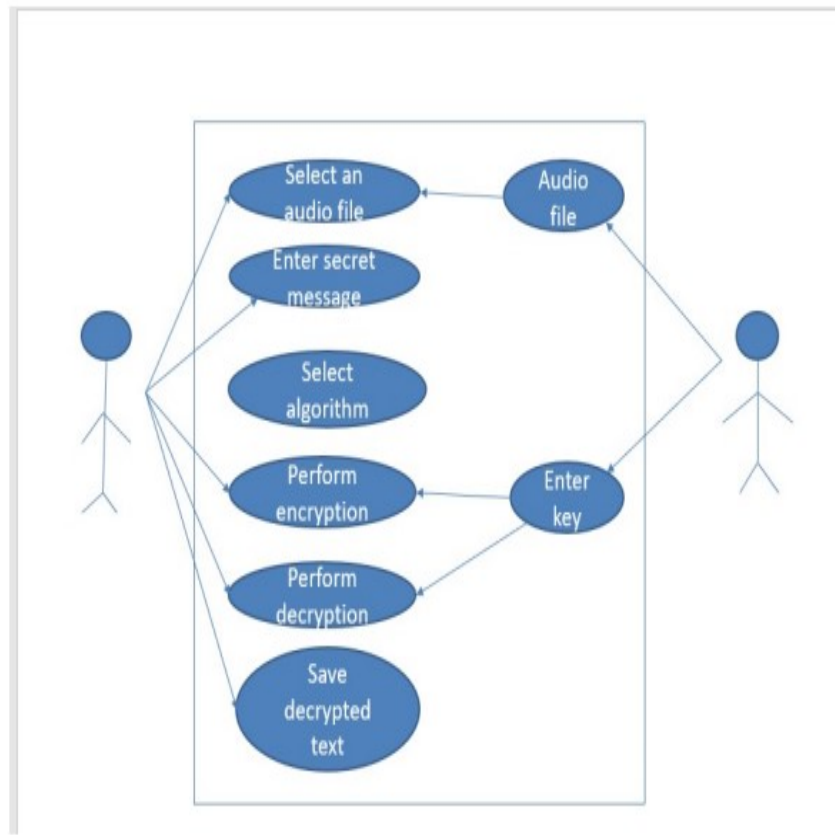


Figure 4.6: **Communication Diagram**

Description To secure secret information different medium methods are used and steganography Manual many individual steganography tools can be used to transfer data securely and in this this proposed tool developed using the last significant bit.

4.2.6 Activity Diagram

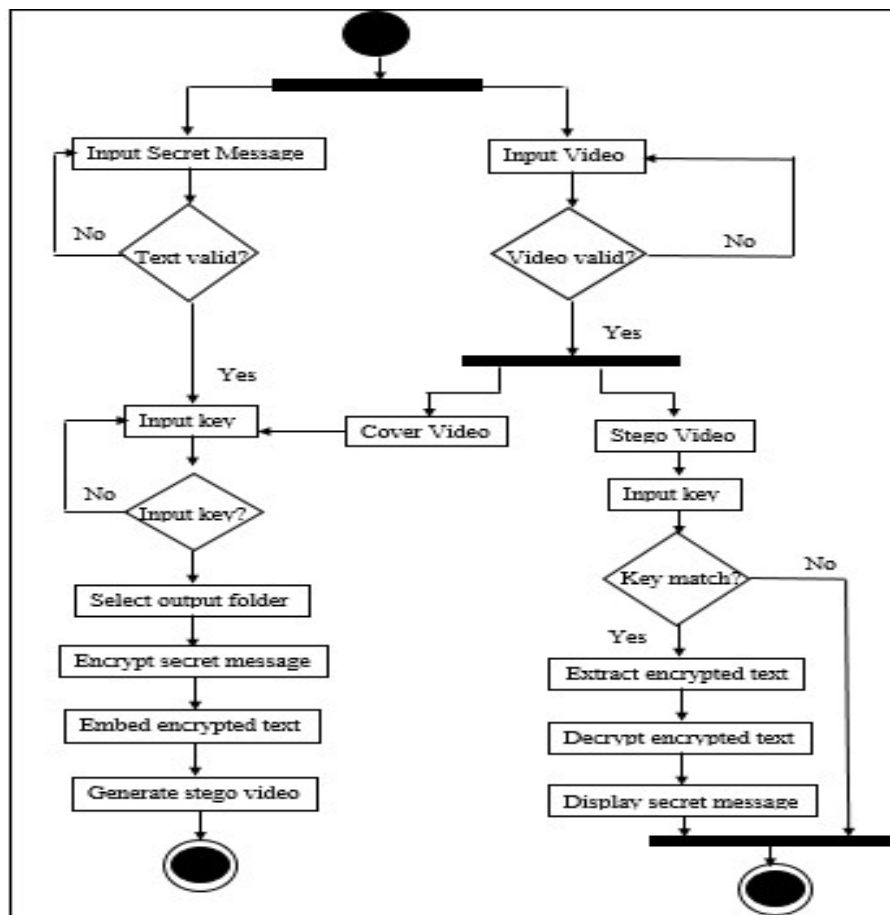


Figure 4.7: System Flow Control

In the system flow chart control gives the detailed information about the encryption and decryption in the hiding of data into the image using those above conditions.

4.3 Algorithm & Pseudo Code

4.3.1 Algorithm

The original image will be required to determine the bits that inverted in each pixel of stego image to recover the secret data. Partition each of the stego and original images into two equal parts and follow the following steps to obtain the secret data.

step 1: To know the maximum value follow the following steps from 2 to 6.

step 2: Compare the three LSBs of the first pixel in the second part of the stego image with the corresponding in the original image to determine the value of the remainder.

step 3: Compare the two LSBs of the first pixel in the first part of the stego image with the corresponding in the original image to determine the value of the quotient.

step 4: Multiple the value of this quotient by 8 and add the value of the remainder.

step 5: Compare the third, fourth, and fifth LSBs of the first pixel in the first part of the stego image with the corresponding in the original image to determine the value of the quotient.

step:6: Multiple the value of this quotient by 32 and add the value obtained in step 4.

step:7: Compare the fourth LSB of the first pixel in the second part of the stego image with the corresponding in the original image to determine how many bits are used for embedding for each pixel in the first part.

step:8: For each pixel in stego image, repeat the step 2 and compare the fourth LSB with the corresponding in the original image to know which inverted bits are inverted again.

step:9: After knowing the inverted bits that are inverted again, restore its value and apply the steps from 3 to 6 and subtract the result from the maximum value to obtain the original value of the secredata.

step:10: Repeat the previous step for all pixels to obtain the secret data.

The following is a simple example to further explain the proposed method: assume that the original pixels of the cover image in the first and second parts are 200 and 170 respectively.. The values of secret data are [250, 123, 125] where 250 and 123 are the maximum and minimum values respectively, thus

The value of R is 127.

The result of subtract 125 from 250 is 125, so the value of A=3, B=3, and C=5. B=3 means inverting the first and second LSBs of the original pixel in the first part. Also, A=3 means inverting the third and fourth LSBs of this pixel.

The value of the stego pixel in the first part will be 197. This value after inverting the second LSB again and apply the optimal LSBs method.

The value of C=5, then invert the first and third LSBs of the original pixel in the second part.

Invert the fourth LSB of this pixel to indicate that the second LSB is inverted again in the pixel in the first part.

The value of the stego pixel in the second part will be 167.

Finally, the stego pixels of the cover image in the first and second parts are 197 and 167 respectively.

4.3.2 Pseudo Code

Input

```
1 // Initialize variables
2 plaintext_data = "Your plaintext data"
3 image = "Your image file"
4 encrypted_image = "Path to save the encrypted image"
5 key = "Your encryption key"
6
7 // Convert the plaintext data to binary
8 binary_data = convert_to_binary(plaintext_data)
9
10 // Embed the binary data in the image using steganography
11 steg_image = embed(binary_data , image)
12
13 // Encrypt the steganographic image using a symmetric encryption algorithm
14 encrypted_steg_image = symmetric_encrypt(steg_image , key)
15
16 // Save the encrypted image to a file
17 save_image(encrypted_steg_image , encrypted_image)
18
19 // To decrypt the data:
20 // Load the encrypted image from file
21 encrypted_steg_image = load_image(encrypted_image)
22
23 // Decrypt the steganographic image using the symmetric encryption key
24 steg_image = symmetric_decrypt(encrypted_steg_image , key)
25
26 // Extract the binary data from the steganographic image using steganography
27 binary_data = extract(steg_image)
28
29 // Convert the binary data back to plaintext
30 plaintext_data = convert_to_plaintext(binary_data)
```

4.4 Module Description

4.4.1 Encryption Module

This module is responsible to encrypt both the secret key and secret data. The final output of this module is secret key and secret data bits in encrypted form. This module performs the following operations on secret key and secret data.

1. Select the secret data and a suitable secret key for encryption.
2. Convert the secret key into one-dimensional (1-D) array of bits.
3. Apply the bitxor operation on these bits with logical 1.
4. Shuffle these encrypted bits such that the bits with even and odd indices are interchanged.
5. If secret key bit = 1 Then perform bitxor operation of secret message bit with logical 1. Else Do not perform bitxor operation. End if
6. Repeat step 4 until all secret data bits are encrypted.

4.4.2 Decryption Module

The decrypt module is used to get the hidden information in an image file. It take the image file as an output, and give two file at destination folder, one is the same image file and another is the message file that is hidden it that.

4.4.3 Mapping Module

This module is responsible for mapping the secret encrypted data into the carrier image pixels. Before mapping, the carrier image channels are transformed and then a 1-1 mapping between secret data bits and image pixels is maintained. The end result of this module is a stego image, containing secret information.

4.5 Steps to execute/run/implement the project

4.5.1 Execution

In this first step in project when work is performed, and each and everything in this project is put into the action.

4.5.2 Run

Here in this step we are going to run the program files which are already put in action. This is the main important step, here we are going to get proper output without any errors in programs.

4.5.3 Implementation

This is the final step where after getting output screen we need to check with giving input data to process and by checking with existing dataset, it is going to give proper output.

Chapter 5

IMPLEMENTATION AND TESTING

5.1 Input and Output

5.1.1 Input Design



Figure 5.1: Choosing the Encode/Decode

To decode and encode the data ,we want to encode the data we have to select the encode then select the image from our device to add data and if we have to decrypt the data from image.

5.1.2 Output Design



Figure 5.2: Protecting the Data

In this it can encrypt and decrypt the data from the inserted image .if we have to hide the data then select encryption and add the data ,while we encrypt the data we have to select decrypt.

5.2 Testing

Testing is a process of executing programs with the intention of finding out the errors during the process of execution with the set of test is called test set and the output of the of the detector is evaluated to determine whether the project is performing or expected or not it makes a logical assumption that if all the parts of the module or correct then goal will be successfully achieved testing includes after the completion of the coding face the project is tested from the very beginning and also at each and every step by the entire different type of data in the testing phase some mistakes are found which did not come across to knowledge of program and while coding phase.

5.3 Types of Testing

5.3.1 Unit testing

Unit testing is a level of software testing where individual components of a software are tested. In Unit Testing the Encryption Decryption processes and Image Transmission process are Tested Individually.

Input

```
1  # Creating the button functions
2  def encode_image():
3      encode_wn = Toplevel(root)
4      encode_wn.title("Encode an Image")
5      encode_wn.geometry('600x220')
6      encode_wn.resizable(0, 0)
7      encode_wn.config(bg='AntiqueWhite')
8      Label(encode_wn, text='Encode an Image', font=("Comic Sans MS", 15), bg='AntiqueWhite').place(x
          =220, rely=0)
9
10     Label(encode_wn, text='Enter the path to the image(with extension):', font=("Times New Roman",
11         13),
12         bg='AntiqueWhite').place(x=10, y=50)
13     Label(encode_wn, text='Enter the data to be encoded:', font=("Times New Roman", 13), bg='
14         AntiqueWhite').place(
15         x=10, y=90)
16     Label(encode_wn, text='Enter the output file name (without extension):', font=("Times New Roman"
17         , 13),
18         bg='AntiqueWhite').place(x=10, y=130)
19
20     img_path = Entry(encode_wn, width=35)
21     img_path.place(x=350, y=50)
22
23     text_to_be_encoded = Entry(encode_wn, width=35)
24     text_to_be_encoded.place(x=350, y=90)
25
26     after_save_path = Entry(encode_wn, width=35)
27     after_save_path.place(x=350, y=130)
28
29     Button(encode_wn, text='Encode the Image', font=('Helvetica', 12), bg='PaleTurquoise', command=
30         lambda:
31         main_encryption(img_path.get(), text_to_be_encoded.get(), after_save_path.get())).place(x=220, y
32         =175)
```

Test result



5.3.2 Integration testing

Testing modules are combined into sub systems which are then tested. The goal is to see if the modules are properly integrated, and the emphasis being and the testing interfaces between the modules

Input

```
1 def decode_image():
2     decode_wn = Toplevel(root)
3     decode_wn.title("Decode an Image")
4     decode_wn.geometry('600x300')
5     decode_wn.resizable(0, 0)
6     decode_wn.config(bg='Bisque')
7
8     Label(decode_wn, text='Decode an Image', font=("Comic Sans MS", 15), bg='Bisque').place(x=220,
9         rely=0)
10
11     Label(decode_wn, text='Enter the path to the image (with extension):', font=("Times New Roman",
12         12),
13         bg='Bisque').place(x=10, y=50)
14
15     img_entry = Entry(decode_wn, width=35)
16     img_entry.place(x=350, y=50)
```

```
15
16 text_strvar = StringVar()
17
18 Button(decode_wn, text='Decode the Image', font=('Helvetica', 12), bg='PaleTurquoise', command=
    lambda:
19 main_decryption(img_entry.get(), text_strvar)).place(x=220, y=90)
20
21 Label(decode_wn, text='Text that has been encoded in the image:', font=("Times New Roman", 12),
    bg='Bisque').place(
22     x=180, y=130)
23
24 text_entry = Entry(decode_wn, width=94, text=text_strvar, state='disabled')
25 text_entry.place(x=15, y=160, height=100)
```

Test result

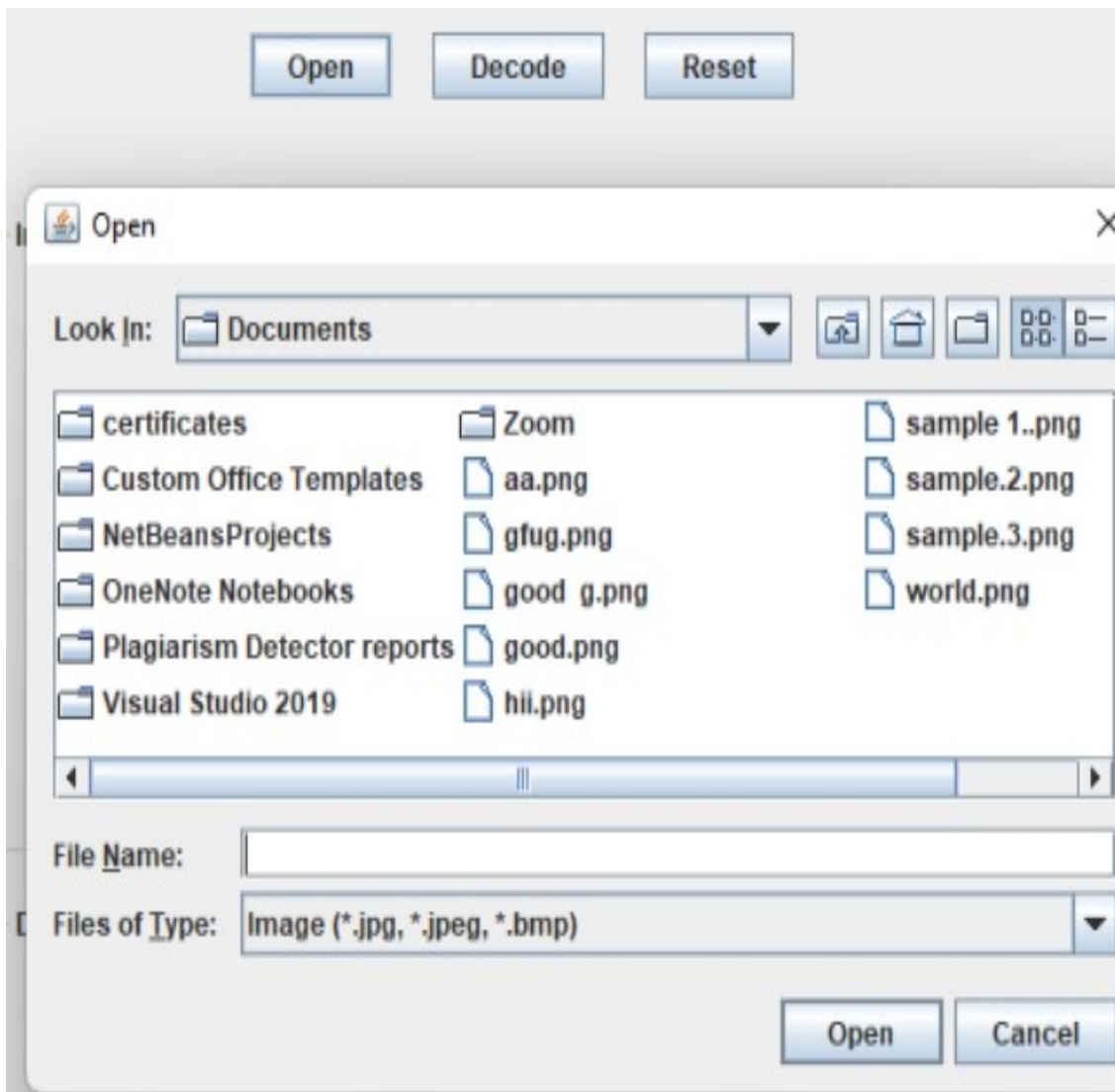


Figure 5.3: Image selection

In the above image we have to select the image from the device to hide the data in it, to choose the file go to encode and click on the selected image from the device .

5.3.3 System testing

Input

```
1 # Creating the basic Image Steganography functions using python
2 def generate_data(pixels , data):
3     # This function will convert the incoming data to 8-bit binary format using its ASCII values and
4     # return them
5     data_in_binary = []
```

```

6     for i in data:
7         binary_data = format(ord(i), '08b')
8         data_in_binary.append(binary_data)
9
10    length_of_data = len(data_in_binary)
11    image_data = iter(pixels)
12
13    for a in range(length_of_data):
14        pixels = [val for val in image_data.__next__()[ :3] + image_data.__next__()[ :3] + image_data.
15                  __next__()[ :3]]
16
17        for b in range(8):
18            if (data_in_binary[a][b] == '1') and (pixels[b] % 2 != 0):
19                pixels[b] -= 1
20            elif (data_in_binary[a][b] == '0') and (pixels[b] % 2 == 0):
21                if pixels[b] == 0:
22                    pixels[b] += 1
23                pixels[b] -= 1
24
25            if (length_of_data-1) == a:
26                if pixels[-1] % 2 == 0:
27                    if pixels[-1] == 0:
28                        pixels[-1] += 1
29                    else:
30                        pixels[-1] -= 1
31
32        pixels = tuple(pixels)
33
34        yield pixels[:3]
35        yield pixels[3:6]
36        yield pixels[6:9]
37
38    def encryption(img, data):
39        # This method will encode data to the new image that will be created
40        size = img.size[0]
41        (x, y) = (0, 0)
42
43        for pixel in generate_data(img.getdata(), data):
44            img.putpixel((x, y), pixel)
45            if size-1 == x:
46                x = 0; y += 1
47            else:
48                x += 1
49
50
51    def main_encryption(img, text, new_image_name):
52        # This function will take the arguments, create a new image, encode it and save it to the same
53        directory
54        image = Image.open(img, 'r')

```



```

54
55 if (len(text) == 0) or (len(img) == 0) or (len(new_image_name) == 0):
56     mb.showerror("Error", 'You have not put a value! Please put all values before pressing the
        button')
57
58 new_image = image.copy()
59 encryption(new_image, text)
60
61 new_image_name += '.png'
62
63 new_image.save(new_image_name, 'png')
64
65
66 def main_decryption(img, strvar):
67     # This function will decode the image given to it and extract the hidden message from it
68     image = Image.open(img, 'r')
69
70     data = ''
71     image_data = iter(image.getdata())
72
73     decoding = True
74
75     while decoding:
76         pixels = [value for value in image_data.__next__()[0:3] + image_data.__next__()[0:3] +
            image_data.__next__()[0:3]]
77
78         # string of binary data
79         binary_string = ''
80
81         for i in pixels[0:8]:
82             if i % 2 == 0:
83                 binary_string += '0'
84             else:
85                 binary_string += '1'
86
87         data += chr(int(binary_string, 2))
88         if pixels[-1] % 2 != 0:
89             strvar.set(data)

```

5.3.4 Test Result



Figure 5.4: **Test Image**

In the above image sourceimage is the normal image before the data embedded. Embedded image contains the data in to the image before the data embedded image and after data embedded image looks same as normal image.

Chapter 6

RESULTS AND DISCUSSIONS

6.1 Efficiency of the Proposed System

The embedding of text in an image is efficient in such a way that any third person cannot detect The Secret communication is going on except for the sender and receiver. the decoding is efficient process as it decode the text of any length which is embedded in the encrypted image. it has a secret pin for decoding which is called as stegano key known to sender and he says to the receiver.

6.2 Comparison of Existing and Proposed System

In the existing system did not use ipv4 address for transmission of images from source to destination and steganokey it is not mandatory in the existing system but it is mandatory in the proposed system.

6.3 Sample Code

```
1 # Importing the necessary modules
2 from tkinter import *
3 from tkinter import messagebox as mb
4 from PIL import Image
5
6 # Creating the basic Python Image Steganography functions
7 def generate_data(pixels, data):
8     # This function will convert the incoming data to 8-bit binary format using its ASCII values and
9     # return them
10     data_in_binary = []
11
12     for i in data:
13         binary_data = format(ord(i), '08b')
14         data_in_binary.append(binary_data)
15
16     length_of_data = len(data_in_binary)
17     image_data = iter(pixels)
```

```

17
18     for a in range(length_of_data):
19         pixels = [val for val in image_data.__next__()[3] + image_data.__next__()[3] + image_data.
20                     __next__()[3]]
21
22         for b in range(8):
23             if (data_in_binary[a][b] == '1') and (pixels[b] % 2 != 0):
24                 pixels[b] -= 1
25             elif (data_in_binary[a][b] == '0') and (pixels[b] % 2 == 0):
26                 if pixels[b] == 0:
27                     pixels[b] += 1
28                 pixels[b] -= 1
29
30             if (length_of_data-1) == a:
31                 if pixels[-1] % 2 == 0:
32                     if pixels[-1] == 0:
33                         pixels[-1] += 1
34                     else:
35                         pixels[-1] -= 1
36
37             pixels = tuple(pixels)
38
39             yield pixels[:3]
40             yield pixels[3:6]
41             yield pixels[6:9]
42
43 def encryption(img, data):
44     # This method will encode data to the new image that will be created
45     size = img.size[0]
46     (x, y) = (0, 0)
47
48     for pixel in generate_data(img.getdata(), data):
49         img.putpixel((x, y), pixel)
50         if size-1 == x:
51             x = 0; y += 1
52         else:
53             x += 1
54
55
56 def main_encryption(img, text, new_image_name):
57     # This function will take the arguments, create a new image, encode it and save it to the same
58     # directory
59     image = Image.open(img, 'r')
60
61     if (len(text) == 0) or (len(img) == 0) or (len(new_image_name) == 0):
62         mb.showerror("Error", 'You have not put a value! Please put all values before pressing the
63             button')
64
65     new_image = image.copy()

```

```

64     encryption(new_image, text)
65
66     new_image_name += '.png'
67
68     new_image.save(new_image_name, 'png')
69
70
71 def main_decryption(img, strvar):
72     # This function will decode the image given to it and extract the hidden message from it
73     image = Image.open(img, 'r')
74
75     data = ''
76     image_data = iter(image.getdata())
77
78     decoding = True
79
80     while decoding:
81         pixels = [value for value in image_data.__next__()[0:3] + image_data.__next__()[0:3] +
82                   image_data.__next__()[0:3]]
83
84         # string of binary data
85         binary_string = ''
86
87         for i in pixels[0:8]:
88             if i % 2 == 0:
89                 binary_string += '0'
90             else:
91                 binary_string += '1'
92
93         data += chr(int(binary_string, 2))
94         if pixels[-1] % 2 != 0:
95             strvar.set(data)
96
97 # Creating the button functions
98 def encode_image():
99     encode_wn = Toplevel(root)
100     encode_wn.title("Encode an Image")
101     encode_wn.geometry('600x220')
102     encode_wn.resizable(0, 0)
103     encode_wn.config(bg='AntiqueWhite')
104     Label(encode_wn, text='Encode an Image', font=("Comic Sans MS", 15), bg='AntiqueWhite').place(x
        =220, rely=0)
105
106     Label(encode_wn, text='Enter the path to the image(with extension):', font=("Times New Roman",
        13),
107           bg='AntiqueWhite').place(x=10, y=50)
108     Label(encode_wn, text='Enter the data to be encoded:', font=("Times New Roman", 13), bg='
        AntiqueWhite').place(
109           x=10, y=90)

```

```

110 Label(encode_wn, text='Enter the output file name (without extension):', font=("Times New Roman"
    , 13),
111       bg='AntiqueWhite').place(x=10, y=130)
112
113 img_path = Entry(encode_wn, width=35)
114 img_path.place(x=350, y=50)
115
116 text_to_be_encoded = Entry(encode_wn, width=35)
117 text_to_be_encoded.place(x=350, y=90)
118
119 after_save_path = Entry(encode_wn, width=35)
120 after_save_path.place(x=350, y=130)
121
122 Button(encode_wn, text='Encode the Image', font=('Helvetica', 12), bg='PaleTurquoise', command=
    lambda:
123 main_encryption(img_path.get(), text_to_be_encoded.get(), after_save_path.get())).place(x=220, y
    =175)
124
125
126 def decode_image():
127     decode_wn = Toplevel(root)
128     decode_wn.title("Decode an Image")
129     decode_wn.geometry('600x300')
130     decode_wn.resizable(0, 0)
131     decode_wn.config(bg='Bisque')
132
133     Label(decode_wn, text='Decode an Image', font=("Comic Sans MS", 15), bg='Bisque').place(x=220,
        rely=0)
134
135     Label(decode_wn, text='Enter the path to the image (with extension):', font=("Times New Roman",
        12),
136           bg='Bisque').place(x=10, y=50)
137
138     img_entry = Entry(decode_wn, width=35)
139     img_entry.place(x=350, y=50)
140
141     text_strvar = StringVar()
142
143     Button(decode_wn, text='Decode the Image', font=('Helvetica', 12), bg='PaleTurquoise', command=
        lambda:
144     main_decryption(img_entry.get(), text_strvar)).place(x=220, y=90)
145
146     Label(decode_wn, text='Text that has been encoded in the image:', font=("Times New Roman", 12),
        bg='Bisque').place(
147         x=180, y=130)
148
149     text_entry = Entry(decode_wn, width=94, text=text_strvar, state='disabled')
150     text_entry.place(x=15, y=160, height=100)
151
152

```

```

153 # Initializing the window
154 root = Tk()
155 root.title(' Image Steganography')
156 root.geometry('300x200')
157 root.resizable(0, 0)
158 root.config(bg='NavajoWhite')
159
160 Label(root, text=' Image Steganography', font=('Comic Sans MS', 15), bg='NavajoWhite',
161        wraplength=300).place(x=40, y=0)
162
163 Button(root, text='Encode', width=25, font=('Times New Roman', 13), bg='SteelBlue', command=
164        encode_image).place(
165        x=30, y=80)
166
167 Button(root, text='Decode', width=25, font=('Times New Roman', 13), bg='SteelBlue', command=
168        decode_image).place(
169        x=30, y=130)
170
171 # Finalizing the window
172 root.update()
173 root.mainloop()

```

Output



Figure 6.1: **Encode the data**

click on Encode and select the image then add the data or secret code into image, if we have to decrypt the data from the embedded image click on the Decode and then select encoded image.



Figure 6.2: **Decrypting the data**

The embedded image can be inserted into the decode option then the hidden data is displayed on the above box by clicking the decode option. To hide the data to the image we have to click on open and select the image give the secret data or code into the image.

Chapter 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Conclusion

The proposed method is developed to ensure the more image security during transmission by facilitating the victim is transfer also the processing image security mechanism had to be effective in the terms of elapsed time transferring the text message which is made to hide in the image using steganography algorithm.

7.2 Future Enhancements

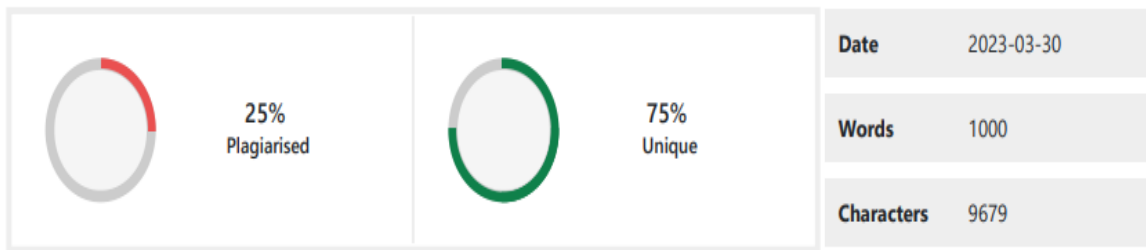
As a part of future work, the same enhanced encryption scheme can also be applied for encryption of audio, speech, and video signals with some necessary alterations. To handle color (RGB) image encryption, the scheme can be implemented in parallel to speed up the encryption process. The security experts should follow certain guidelines for building any image cryptosystem. The designer should develop cryptosystem and analyze it against possible cryptanalysis.

Chapter 8

PLAGIARISM REPORT



PLAGIARISM SCAN REPORT



Content Checked For Plagiarism

SCHOOL OF COMPUTING
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
1156CS601-MINOR PROJECT
WINTER SEMESTER 20-21
REVIEW - II
"SECURING DATA WITH IMAGE ENCRYPTION"

SUPERVISED BY PRESENTED BY
DR. V. JEEVANANTHAM
Assistant Professor 1. C. SAI KARTHIK (VTU16975) (20UECS0178)
2. T. CHAKRADHAR (VTU15064) (20UECS0929)
3. P. ANIRUDH REDDY (VTU21646) (20UECS1073)

Figure 8.1: Plagraism report

Chapter 9

SOURCE CODE & POSTER PRESENTATION

9.1 Source Code

```
1 # Importing the necessary modules
2 from tkinter import *
3 from tkinter import messagebox as mb
4 from PIL import Image
5
6 # Creating the basic Python Image Steganography functions
7 def generate_data(pixels, data):
8     # This function will convert the incoming data to 8-bit binary format using its ASCII values and
9     # return them
10     data_in_binary = []
11
12     for i in data:
13         binary_data = format(ord(i), '08b')
14         data_in_binary.append(binary_data)
15
16     length_of_data = len(data_in_binary)
17     image_data = iter(pixels)
18
19     for a in range(length_of_data):
20         pixels = [val for val in image_data.__next__()[ :3] + image_data.__next__()[ :3] + image_data.
21             __next__()[ :3]]
22
23         for b in range(8):
24             if (data_in_binary[a][b] == '1') and (pixels[b] % 2 != 0):
25                 pixels[b] -= 1
26             elif (data_in_binary[a][b] == '0') and (pixels[b] % 2 == 0):
27                 if pixels[b] == 0:
28                     pixels[b] += 1
29                 pixels[b] -= 1
30
31         if (length_of_data - 1) == a:
32             if pixels[-1] % 2 == 0:
33                 if pixels[-1] == 0:
34                     pixels[-1] += 1
35                 else:
```

```

34         pixels[-1] -= 1
35
36     pixels = tuple(pixels)
37
38     yield pixels[:3]
39     yield pixels[3:6]
40     yield pixels[6:9]
41
42
43 def encryption(img, data):
44     # This method will encode data to the new image that will be created
45     size = img.size[0]
46     (x, y) = (0, 0)
47
48     for pixel in generate_data(img.getdata(), data):
49         img.putpixel((x, y), pixel)
50         if size-1 == x:
51             x = 0; y += 1
52         else:
53             x += 1
54
55
56 def main_encryption(img, text, new_image_name):
57     # This function will take the arguments, create a new image, encode it and save it to the same
58     # directory
59     image = Image.open(img, 'r')
60
61     if (len(text) == 0) or (len(img) == 0) or (len(new_image_name) == 0):
62         mb.showerror("Error", 'You have not put a value! Please put all values before pressing the
63         button')
64
65
66     new_image = image.copy()
67     encryption(new_image, text)
68
69
70     new_image_name += '.png'
71
72     new_image.save(new_image_name, 'png')
73
74
75 def main_decryption(img, strvar):
76     # This function will decode the image given to it and extract the hidden message from it
77     image = Image.open(img, 'r')
78
79     data = ''
80     image_data = iter(image.getdata())
81
82     decoding = True
83
84     while decoding:

```

```

81     pixels = [value for value in image_data.__next__()[ :3] + image_data.__next__()[ :3] +
82                 image_data.__next__()[ :3]]
83
84     # string of binary data
85     binary_string = ''
86
87     for i in pixels[:8]:
88         if i % 2 == 0:
89             binary_string += '0'
90         else:
91             binary_string += '1'
92
93     data += chr(int(binary_string, 2))
94     if pixels[-1] % 2 != 0:
95         strvar.set(data)
96
97 # Creating the button functions
98 def encode_image():
99     encode_wn = Toplevel(root)
100     encode_wn.title("Encode an Image")
101     encode_wn.geometry('600x220')
102     encode_wn.resizable(0, 0)
103     encode_wn.config(bg='AntiqueWhite')
104     Label(encode_wn, text='Encode an Image', font=("Comic Sans MS", 15), bg='AntiqueWhite').place(x
        =220, rely=0)
105
106     Label(encode_wn, text='Enter the path to the image(with extension):', font=("Times New Roman",
        13),
107           bg='AntiqueWhite').place(x=10, y=50)
108     Label(encode_wn, text='Enter the data to be encoded:', font=("Times New Roman", 13), bg='
        AntiqueWhite').place(
109           x=10, y=90)
110     Label(encode_wn, text='Enter the output file name (without extension):', font=("Times New Roman"
        , 13),
111           bg='AntiqueWhite').place(x=10, y=130)
112
113     img_path = Entry(encode_wn, width=35)
114     img_path.place(x=350, y=50)
115
116     text_to_be_encoded = Entry(encode_wn, width=35)
117     text_to_be_encoded.place(x=350, y=90)
118
119     after_save_path = Entry(encode_wn, width=35)
120     after_save_path.place(x=350, y=130)
121
122     Button(encode_wn, text='Encode the Image', font=('Helvetica', 12), bg='PaleTurquoise', command=
        lambda:
123         main_encryption(img_path.get(), text_to_be_encoded.get(), after_save_path.get())).place(x=220, y
        =175)

```

```


124
125
126 def decode_image():
127     decode_wn = Toplevel(root)
128     decode_wn.title("Decode an Image")
129     decode_wn.geometry('600x300')
130     decode_wn.resizable(0, 0)
131     decode_wn.config(bg='Bisque')
132
133     Label(decode_wn, text='Decode an Image', font=("Comic Sans MS", 15), bg='Bisque').place(x=220,
134         rely=0)
135
136     Label(decode_wn, text='Enter the path to the image (with extension):', font=("Times New Roman",
137         12),
138         bg='Bisque').place(x=10, y=50)
139
140
141     img_entry = Entry(decode_wn, width=35)
142     img_entry.place(x=350, y=50)
143
144     text_strvar = StringVar()
145
146     Button(decode_wn, text='Decode the Image', font=('Helvetica', 12), bg='PaleTurquoise', command=
147         lambda:
148         main_decryption(img_entry.get(), text_strvar)).place(x=220, y=90)
149
150     Label(decode_wn, text='Text that has been encoded in the image:', font=("Times New Roman", 12),
151         bg='Bisque').place(
152         x=180, y=130)
153
154     text_entry = Entry(decode_wn, width=94, text=text_strvar, state='disabled')
155     text_entry.place(x=15, y=160, height=100)
156
157
158 # Initializing the window
159 root = Tk()
160 root.title('Image Steganography')
161 root.geometry('300x200')
162 root.resizable(0, 0)
163 root.config(bg='NavajoWhite')
164
165 Label(root, text='Image Steganography', font=('Comic Sans MS', 15), bg='NavajoWhite',
166     wraplength=300).place(x=40, y=0)
167
168 Button(root, text='Encode', width=25, font=('Times New Roman', 13), bg='SteelBlue', command=
169     encode_image).place(
170     x=30, y=80)
171
172 Button(root, text='Decode', width=25, font=('Times New Roman', 13), bg='SteelBlue', command=
173     decode_image).place(
174     x=30, y=130)

```



```
168  
169 # Finalizing the window  
170 root.update()  
171 root.mainloop()
```

9.2 Poster Presentation



Vel Tech
Rangarajan Dr. Sagunthala
R&D Institute of Science and Technology
Chennai-600 062

SECURING DATA WITH IMAGE ENCRYPTION

Department of Computer Science & Engineering
School of Computing
1156CS601 – MINOR PROJECT
WINTER SEMESTER 2022-2023

ABSTRACT

> With rapid progress in internet and digital imaging technology there are more and more ways

> to easily create, publish and distribute images. With rapid progress in internet and digital imaging technology there are more and more ways

> to easily create, publish and distribute images

> We use LSB algorithm which is Steganography method used to embed the image for sending through the internet to the receiver which anyone can find that it is embedded.

> Image encryption has applications in corporate world, health care etc.

TEAM MEMBER DETAILS

vtu15064/T. chakradhar
vtu16975/c.sai karthik
vtu21646/P.Anilrudh
<Student 1. 8790862389>
<Student 2. 9704230079>
<Student 3. 7995138808>
vtu15064@veltech.edu.in
vtu16975@veltech.edu.in
vtu21646@veltech.edu.in

INTRODUCTION



The use of chaotic methods in the encryption makes it more difficult for the attackers to decrypt the image. Every image to be transferred is first partitioned and then each part is encrypted and sent to the receiver. That is why a person with a single part or two parts of the image will not be able to use that image. It's important that without decryption key no one can access the content. Image encryption has applications in internet communication, multimedia systems, medical imaging, telemedicine, military communication etc. The image encryption and decryption was done by using Steganography. It is a method of hiding secret data, by embedding it into an audio, video, image or text file.

• MODULE 1

- Image uploading and Embedding with text.
- Step 1: User Interface of the Applications Main Menu. The frontend of the Applications user interface is designed with a menu having three options of Encryption, Decryption and Send Image. Each option in the menu performs the task of it respectively.
- Step 2: Encryption. When the User chooses for Encryption option from the main menu, The user will be taken to the Encryption phase where the User has to select an image and specify the secret message that to be embedded. Step 3: Stego Key. While encrypting User has to give any random pin which is called Stego key. This Stego key is used by the Receiver to Decode the message during Decryption.

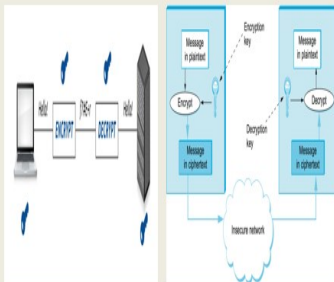
RESULTS

Transferring the text message is made to hide in the image using steganography algorithm

STANDARDS AND POLICIES

The purpose of this policy is to provide guidance that limits the use of encryption to those algorithms that have received substantial public review and have been proven to work effectively. Additionally, This policy provides direction to ensure that federal regulations are followed and legal authority is granted for the dissemination and use of encryption technologies.



CONCLUSIONS

The proposed scheme is developed to ensure the more image security during transmission by facilitating the victim's transfer. Also, the processing image security mechanism had to be effective in the terms of elapsed time transferring the text message which is made to hide in the image using steganography algorithm.

ACKNOWLEDGEMENT

- Project Supervisor DR.V.JEEVANANTHAM/ME.Ph.D
- Project supervisor Contact No: 9629194212
- Project supervisor Mail ID: drjeevananthamv@veltech.edu.in

Figure 9.1: Poster

References

- [1] Abdullah A Rashid, Khalid Ali Hussein. Image encryption algorithm based on the density and 6D logistic map. Inform. Sciences, 2023, 539: 198–215. year 2023.
- [2] Huda R Shakir, Sadiq A Mehdi, Anwar A Hattab. A new four-dimensional hyper-chaotic system for image encryption . Appl. Sof Comput. 94, 106164, year 2023.
- [3] Noor Sattar Noor, Dalal Abdulmohsin Hammood, Ali Al-Naji, Javaan Chahl. A fast text-to-image encryption-decryption algorithm for secure network communication Nonlinear Dyn. 94, 776–782 year 2022.
- [4] C. Xiuli, G. Zhihua, Y. Ke, et al., An image encryption scheme based on three-dimensional Brownian motion and chaotic system [J]. Chinese Physics B 26(2), 97–115, year 2020
- [5] Luo YL, Zhou RL, Liu JX, Cao Y, Ding XM. A parallel image encryption algorithm based on the piecewise linear chaotic map and hyper-chaotic map. Nonlinear r Dynam., 2020, 93(3): 1167–1184.
- [6] G. S. Chandel, P. Patel, "A Review: Image Encryption with RSA and RGB randomized Histograms," International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE), vol. 2, Issue 11, November 2020
- [7] Dr. P. Mahajan, A. Sachdeva, "A Study of Encryption Algorithms AES, DES and RSA for Security", Global Journal of Computer Science and Technology Network, Web Security (GJCSTNWS), vo. 13 Issue. 15 Version 1.0 Year 2020
- [8] L. Singh, Dr. R.K. Bharti, "Comparative performance analysis of Cryptographic Algorithms," International Journal of Advanced Research and Computer Science

