

COMBINING PQC AND IMAGE STEGANOGRAPHY FOR SECURE COMMUNICATION

A Project Report submitted to

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY
ANANTAPUR, ANANTHAPURAM**

In partial fulfillment of the requirements for the award of the degree of

**BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE & ENGINEERING**

Submitted by

Shaik Mohammad Fahed	208R1A0547
Sanepalli Aswini	208R1A0543
Bijivemula Sai Vineela	208R1A0508
Ummanaboyina Nagarjuna	208R1A0553

Under the esteemed guidance of

Mr. V.K.Sabari Rajan

M.E(SE), Assistant professor



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SAI RAJESWARI INSTITUTE OF TECHNOLOGY**

AN ISO 9001:2015 CERTIFIED INSTITUTION

(Approved by AICTE, New Delhi and Affiliated to J.N.T.U.A. Ananthapuramu)

Lingapuram (V), Proddatur, Kadapa(Dist.)-516360

2020-2024



SAI RAJESWARI INSTITUTE OF TECHNOLOGY

AN ISO 9001:2015 CERTIFIED INSTITUTION

(Approved by AICTE, New Delhi and Affiliated to J.N.T.U.A. Ananthapuramu)

Lingapuram (V), Proddatur, Kadapa (Dist.)-516360,A.P

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Certificate

This is to certify that the Project Report entitled

COMBINING PQC AND IMAGE STEGANOGRAPHY FOR SECURE COMMUNICATION

is the bonafide work done and

Submitted by

Shaik Mohammad Fahed	208R1A0547
Sanepalle Aswini	208R1A0543
Bijivemula Sai Vineela	208R1A0508
Ummanaboyina Nagarjuna	208R1A0553

In the **Department of Computer Science and Engineering, Sai Rajeswari Institute of Technology**, Proddatur affiliated to J.N.T.U.A., Ananthapuramu in partial fulfillment of the requirements for the award of Bachelor of Technology in **Computer Science and Engineering** during Academic **Year 2023-2024**.

Mr. V.K. Sabari Rajan, M.E(SE)

PROJECT GUIDE

Department of CSE

Dr. Y. SUBBA REDDY, M.E, Ph. D

HEAD OF THE DEPARTMENT

Department of CSE

External Viva voce conducted on _____

Internal Examiner

External Examiner



AcenAAR Technologies Pvt. Ltd.

76/2-sku-4f-1, Skandhanshi Vyapaar, Tailors Colony,
Revenue ward no. 76, Kurnool-518001, Andrapradesh, India.
GSTIN NO: 37AAVCA3004E1ZU
CIN : U31901AP2021PTC118455

Mobile: 7032131793, 7989942182 web: www.acenaartechnologies.com e_mail: info@acenaartechnologies.com

Certificate

This is to confirm and certify that **Shaik Mohammad Fahed, S.Aswini,**
B.Sai Vineela, U.Nagarjuna B. Tech, students of Sai Rajeswari Institute of Technology,
Proddatur. With **208R1A0547,208R1A0543,208R1A0508,208R1A0553** has successfully
completed his/her project work titled Combining “PQC and Image Steganography for
Secure Communication” on (Cryptography) Technology as part of his course curriculum.

He has done this project using (MATLAB) during the period of January 2024 to April
2024 under the guidance and supervision of **Ahmad Valli** from (ACENAAR
TECHNOLOGIES PVT. LTD).

He has completed the assigned project well within the time frame. He is sincere,
hardworking and his conduct during the project is commendable.



*Authorized Signatory with Date and
Seal*

ACKNOWLEDGEMENT

Any achievement, be it scholastic or otherwise does not depend solely on the individual effort but the guidance, encouragement, and cooperation of intellectuals, elders, and friends. I would like to take this opportunity to thank them all.

We express our sincere thanks to **Dr. PANDURANGAN RAVI**, M.Tech, Ph.D., our beloved Principal, for his encouragement and suggestions during our course of study.

With a deep sense of gratitude, We acknowledge **Dr. Y.SUBBA REDDY**, M.E, Ph.D., Head of the Dept., Computer Science & Engineering, for his valuable support and help in completing our project successfully.

We express our sincere thanks to Project Co-Ordinator **Mrs. I.Sravani, MTech**, Assistant Professor in the Department of Computer Science And Engineering, for providing us her valuable gratitude and inspiration in carrying out our project studies.

We wholeheartedly express our gratitude and esteemed regards to Project guide **Mr. V.K.Sabari Rajan, M.E(SE)**, in the Department of Computer Science and Engineering, for providing us his valuable gratitude and inspiration in carrying out our project studies. His constant support and encouragement enabled us to complete this work successfully.

We feel honored for placing our warm salutation to **THE MANAGEMENT** Sai Rajeswari Institute of Technology, Proddatur, which allowed us to obtain a strong base in B. Tech and profound knowledge.

Finally, we would like to express our sincere thanks to all the **Faculty Members** of the C.S.E Department, Lab Technicians, Friends & Family members, and all, who have helped us to complete this work successfully.

Shaik Mohammad Fahed	208R1A0547
Sanepalle Aswini	208R1A0543
Bijivemula Sai Vineela	208R1A0508
Ummanaboyina Nagarjuna	208R1A0553

CONTENTS

CHAPTER NO.	CHAPTER NAME		PAGE NO.
	Title page		
	Certificate		
	Acknowledgement		
	Abstract		
	List of figures		
1	INTRODUCTION		1-11
	1.1	Objective	
2	THEORETICAL FOUNDATIONS AND LITERATURE REVIEW		12-14
3	SYSTEM ANALYSIS		15-16
	3.1	Existing System	
	3.2	Disadvantages of Existing System	
	3.3	Proposed System	
	3.4	Advantages of Proposed System	
4	SYSTEM REQUIREMENTS		17
5	METHODOLOGY		18-19
6	SYSTEM DESIGN		20-28
	6.1	Class Diagram	
	6.2	Use Case Diagram	
	6.3	Sequence Diagram	
	6.4	Collaboration Diagram	
	6.5	Activity Diagram	
7	STATISTICAL METHODOLOGIES IN IMAGE STEGANOGRAPHY		29-30
8	SYSTEM CODING		31-47
	8.1	Sample Code	
	8.2	About programming Language	
9	IMPLEMENTATION		48-49
	9.1	Modules	
10	SYSTEM TESTING		50-55

11	RESULTS AND DISCUSSIONS		56-63
	11.1	Experiment Result	
	11.2	Advantages & Applications	
12	CONCLUSION AND FUTURE SCOPE		64-65
13	REFERENCES		66

LIST OF DIAGRAMS

Figure No.	Figure Name	Page no.
1	PQC Encryption & Decryption	2
2	RSA Cryptography	3
3	Existing & Proposed System	16
4	Class Diagram	21
5	Use Case Diagram	23
6	Activity Diagram	26
7	State Diagram	28
8	PQC Encryption GUI	36
9	PQC Decryption GUI	40
10	MATLAB Keywords	47
11	Encoding & Bit Error Rate Calculation	51
12	Encryption Screenshots	59
13	Decryption GUI Screenshots	62
	LIST OF TABLES	
1	MATLAB Keywords	47

ABSTRACT

With the rapid advancement of quantum computing and the increasing threat it poses to traditional cryptographic methods, there is a pressing need for post-quantum cryptography (PQC) solutions to ensure secure communication in the future. This work introduces a groundbreaking approach combines RSA techniques with image steganography to achieve enhanced security in communication systems. The proposed system leverages RSA algorithms, which are resistant to attacks from both classical and quantum computers, to encrypt the communication data. By employing these algorithms, we ensure that the confidentiality and integrity of the transmitted information are preserved even in the presence of powerful quantum adversaries. In addition to RSA, we integrate image steganography techniques into our system to further enhance security and concealment. Steganography involves embedding secret data within innocuous cover images, making it difficult for unauthorized parties to detect the existence of hidden information. By embedding encrypted communication data into images using steganographic methods, we add an additional layer of obfuscation, thereby strengthening the security of the communication channel. To assess our approach's effectiveness, we implement the system using MATLAB, a widely-used platform for research and development in image processing and cryptography. Experimentation assesses the system's security and performance of the system, considering factors such as encryption strength, data embedding capacity, and computational efficiency.

Our results demonstrate that the combination of RSA and image steganography offers a robust solution for secure communication in the post-quantum era. By leveraging the strengths of both techniques, we achieve heightened security while maintaining practicality and efficiency in real-world applications. This research contributes to the ongoing efforts to develop secure communication protocols that can withstand the challenges posed by quantum computing advancements.

Keywords: Post-Quantum Cryptography (PQC), Image Steganography, Lattice-based Cryptography, Code-based Cryptography, Rivest Shamir Adleman.

CHAPTER-1

INTRODUCTION

Chapter One serves as an introductory section, emphasizing the critical importance of information security in today's digital landscape and the multifaceted challenges encountered in safeguarding data during transmission. It elucidates the foundational pillars of information security, including confidentiality, integrity, and availability, while also introducing cryptography and steganography as pivotal techniques for concealing information.

Steganography, a method for covertly embedding data within an innocuous carrier, is presented as a clandestine communication method, alongside watermarking. In steganography, the primary information or payload is concealed within secondary information, often obscured from plain view, resulting in a steganography signal or file. However, traditional steganographic methods may be vulnerable to attacks during transmission, storage, or format conversion.

A crucial distinction is made between steganography and watermarking: whereas steganography conceals information entirely, watermarking allows the message to be discernible to a third party. Both techniques entail embedding information into host media discreetly, with invisible watermarking being particularly suitable for scenarios where revealing the concealed information could lead to alterations.

In today's digital age, ensuring the security of sensitive information is paramount. With the rapid advancement of technology, the processing, transmission, and storage of data have become ubiquitous, necessitating robust security measures to protect against unauthorized access and manipulation. Two primary methods employed for information security are cryptography and steganography.

Cryptography involves the transformation of plaintext into ciphertext using cryptographic algorithms and keys, ensuring confidentiality and integrity. On the other hand, steganography focuses on concealing the existence of the message itself, making it undetectable to unintended recipients while preserving the appearance of the cover media.

PQC AND RSA

NIST (National Institute of Standards and Technology) started its Post-Quantum Cryptography Standardization program with 69 selected algorithms in its first round, which after the third round is now left with only 7 finalists and 8 alternate algorithms.

The main algorithms were lattice-based, code-based, multivariate, hash-based, and Isogeny based cryptography. The final seven algorithms consist of 4 key encapsulation mechanisms (KEM) and 3 signature algorithms. Out of 4 algorithms of KEM, 3 are from the lattice-based family: Kyber, NTRU, and SABER whereas McEliece is from the code-based family. Currently, Kyber is amongst the top contenders. Let's see what happens in its Key Encapsulation Mechanism:

Firstly, the key generation process is performed which creates a public key and a secret key. Then the sender ENCAPSULATES the message or plain text with the receiver's public key which then creates two different things: a SHARED SECRET (ss1) and a CIPHER TEXT (ct). Now the receiver takes this ciphertext and DECAPSULATES it with his/her own secret key and this process also creates a Shared Secret(ss2). If both the shared secrets are equal i.e. $ss1 = ss2$ then the verified process has been done and the message has been successfully and securely reached the receiver. This is how the key encapsulation mechanism is performed.

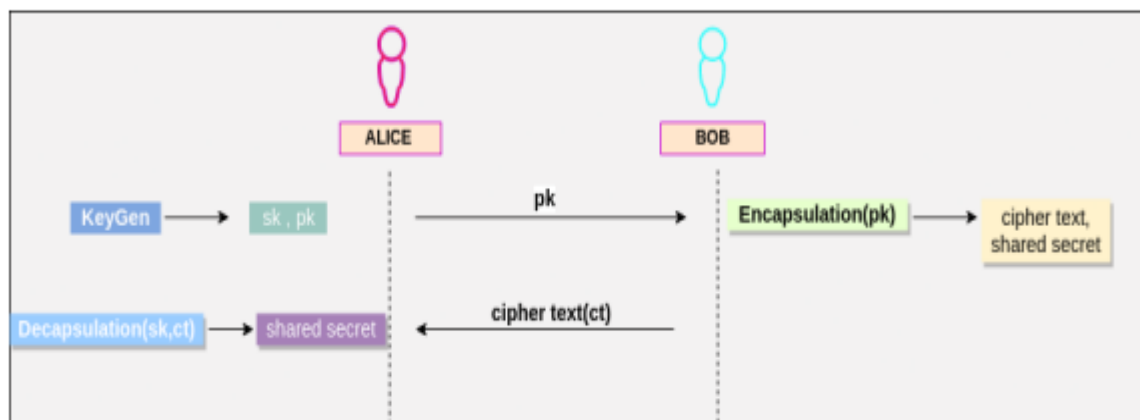


Fig 1: PQC Encryption and Decryption

RSA

In this cryptography instead of using a single key, we now use key pairs. Now both the sender and receiver will carry two-two keys i.e. Public key and the Secret Key(Private key) on both sides, which means now there will be a total of four numbers of keys. As the name suggests public key will be known to everyone and the secret key or private key will remain secret i.e. will be known only to the person carrying it. The sender will now use the public key of the receiver to encrypt the data and since the data is encrypted with the public key of the receiver, only the secret key of the receiver will be able to decrypt it.

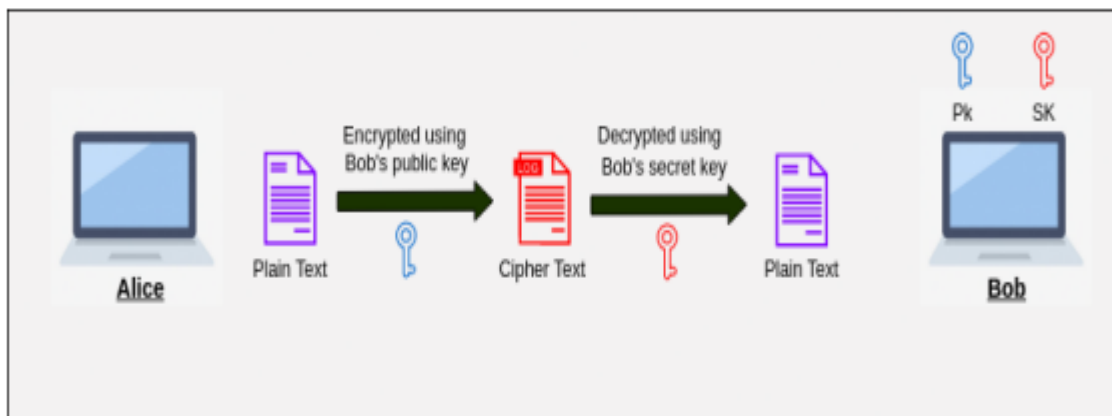


Fig 2: RSA Cryptography

Asymmetric key cryptography uses RSA for encryption and decryption purposes which is now really vulnerable against quantum computers. Quantum computers performing Shor's Algorithm can easily break such methods.

PROBLEM STATEMENT

Despite the effectiveness of image steganography in securing embedded data for various applications, existing techniques often overlook message integrity. The widely used Least Significant Bit (LSB) method lacks encryption prior to embedding secret messages, rendering them susceptible to interception or manipulation during transmission.

1.1. OBJECTIVE

In light of the limitations of the LSB method, this study proposes the development of an advanced image steganography method, leveraging LSB substitution alongside RSA encryption. The primary objective is to bolster the security aspects of image steganography while also integrating measures for verifying message integrity.

THE SPECIFIC OBJECTIVES OF THIS RESEARCH INCLUDE:

Designing a novel algorithm for image steganography utilizing LSB substitution along with RSA encryption to mitigate the security deficiencies associated with the conventional LSB approach.

Implementing a comprehensive evaluation of the proposed image steganography system, encompassing an assessment of its vulnerability to potential attacks.

Developing mechanisms to ensure dependable extraction and verification of embedded messages, thereby fortifying overall message integrity.

Integration of RSA and LSB Substitution in Image Steganography

Combining RSA encryption with LSB substitution presents a promising strategy for bolstering data security in image steganography. RSA encryption offers robust encryption, ensuring confidentiality, while LSB substitution enables covert embedding within digital images.

Through the encryption of the message with RSA prior to embedding using LSB substitution, the confidentiality of the data is upheld, ensuring that only authorized recipients possessing the corresponding decryption key can access the original message. This integration establishes a multilayered approach to data security, reinforcing the integrity of the steganographic process.

The utilization of RSA encryption and LSB substitution in tandem holds potential for various applications requiring secure communication, including digital watermarking, copyright protection, and covert messaging. Furthermore, extending this approach to multimedia formats such as audio and video holds promise for further enhancing security in communication channels.

Steganography Types:

Pure Steganography:

In image-based steganography, pure steganography involves embedding secret information without altering the original image. The process maps $E: C \times M \rightarrow C$, where C represents all possible cover images and M represents all possible messages. The extraction formula $D: C \rightarrow M$ is used to retrieve the secret message from the cover image. Algorithms used in pure steganography are kept private, accessible only to authorized parties involved in the communication. However, this method provides limited security as it assumes that other parties are indifferent to the presence of secret messages.

Private (Secret) Key Steganography:

Private key steganography employs a secret key (stego-key) to embed the message into the cover image. Only individuals with knowledge of the secret key can extract and decipher the original message. Unlike pure steganography, the secret key adds an additional layer of security as it is exchanged through a secure channel. Even if intercepted, messages remain protected, accessible only to those with the correct keys.

Public Key Steganography:

Public key steganography relies on public cryptography keys to establish secure communication channels. The message is encoded with a public key at the sender's end, and only the corresponding secret key can decode it. This method enhances security by leveraging public key cryptography, with different security layers preventing unauthorized parties from accessing the communication. Cracking the algorithm requires knowledge of both the steganography techniques and the implementation of public key cryptography.

Mediums Used for Image Steganography:

Text Hiding in Images:

In image steganography, text information can be hidden within images. Techniques such as line-shift coding and word-shift coding are utilized to conceal text within images. Since images can ignore certain formatting details, text can be effectively hidden without significantly altering the visual appearance of the image.

Image File Structure

Image files serve as widely used formats for storing and transmitting visual data, playing a crucial role in various applications such as photography, graphic design, and digital communication. Understanding the structure of image files is essential for effective manipulation and embedding of data within them, particularly in the context of steganography.

JPEG File Format

JPEG (Joint Photographic Experts Group) is a commonly used image compression format designed for reducing file size while preserving image quality. The JPEG compression algorithm achieves this by discarding redundant image information and storing the remaining data more efficiently. This lossy compression technique allows for substantial reduction in file size with minimal perceptible loss in image quality.

JPEG files consist of multiple segments, each serving a specific purpose in storing metadata and image data. These segments include the Start of Image (SOI), Quantization Table (DQT), Huffman Table (DHT), Start of Frame (SOF), Start of Scan (SOS), and End of Image (EOI), among others. These segments collectively define the structure of the JPEG file and facilitate efficient encoding and decoding of image data.

Within the image data segment, individual components such as color channels (e.g., red, green, blue) are encoded using discrete cosine transform (DCT) and quantization techniques. The resulting coefficients are then compressed using Huffman encoding, yielding the final compressed image data.

Header Information

The header of a JPEG file contains essential metadata and parameters necessary for decoding the image data. These include:

- **Start of Image (SOI):** Marks the beginning of the JPEG file.
- **Application Segment (APP):** Contains application-specific metadata such as Exif data.
- **Quantization Table (DQT):** Specifies the quantization table used for compressing the image data.
- **Huffman Table (DHT):** Defines the Huffman coding tables for entropy encoding.
- **Start of Frame (SOF):** Specifies parameters such as image dimensions and color space.
- **Start of Scan (SOS):** Indicates the beginning of the image data and defines the order of color components.
- **End of Image (EOI):** Marks the end of the JPEG file.
- Each segment within the JPEG file header serves a specific purpose in defining the image's characteristics and encoding parameters, facilitating accurate decoding and rendering of the image data.

PNG File Structure

PNG (Portable Network Graphics) is a popular image file format known for its lossless compression and support for transparency. Understanding the structure of PNG files is essential for effectively embedding data within them, especially in the context of steganography.

PNG files consist of several critical components, including:

- **Header:** The header of a PNG file contains essential information about the image, such as its width, height, color depth, and compression method. It also includes a signature that identifies the file as a PNG image.
- **Chunk Data:** PNG files are organized into multiple data chunks, each serving a specific purpose. Common chunk types include IHDR (Image Header), PLTE (Palette), IDAT (Image Data), and IEND (End of Image). These chunks collectively define the structure and content of the PNG image.

- **Image Data:** The IDAT chunk contains compressed image data encoded using the DEFLATE algorithm. This compressed data represents the actual pixel values of the image.
- **Ancillary Chunks:** Ancillary chunks contain additional information such as transparency data, text annotations, and color profiles. These chunks are optional but can provide valuable metadata about the image.
- **CRC (Cyclic Redundancy Check):** Each chunk in a PNG file is accompanied by a CRC checksum, which is used to verify the integrity of the chunk's data. This checksum helps detect any corruption or tampering in the image file.

By analyzing the structure of PNG files and understanding the purpose of each component, steganographic techniques can be applied to embed hidden data within PNG images while preserving their lossless quality and transparency features.

GIF File Structure

GIF (Graphics Interchange Format) is a widely used image file format known for its support for animations and transparent backgrounds. Understanding the structure of GIF files is essential for effectively embedding data within them, particularly in the context of steganography.

GIF files consist of several key components, including:

- **Header:** The header of a GIF file contains essential information about the image, such as its width, height, color depth, and palette size. It also includes a signature that identifies the file as a GIF image.
- **Logical Screen Descriptor:** This section of the GIF file defines parameters such as the image's global color table, background color, and pixel aspect ratio.
- **Image Data:** GIF files can contain multiple images, each stored as a separate frame. The image data for each frame is encoded using the Lempel-Ziv-Welch (LZW) compression algorithm, which reduces redundancy and achieves efficient compression.
- **Graphic Control Extension:** This optional extension contains parameters for controlling aspects of the image, such as transparency and animation delay.
- **Global Color Table:** If present, the global color table defines the color palette used for all images in the GIF file. Each color entry is typically represented by 24 bits (8 bits each for red, green, and blue channels).

By analyzing the structure of GIF files and understanding the purpose of each component, steganographic techniques can be applied to embed hidden data within GIF images while preserving their animation and transparency features.

The Basic Steganography System Model for Image

The fundamental steganography system model for images comprises several essential components:

- 1. Cover Object:** The original image file serves as the carrier for concealing the secret message. It could be in formats such as JPEG, PNG, or BMP.
- 2. Secret Message:** This is the confidential information intended to be hidden within the cover image. It could be text, another image, or any other form of data.
- 3. Embedding Algorithm:** The embedding algorithm is responsible for concealing the secret message within the cover image. It operates based on predefined rules and parameters to seamlessly integrate the secret message while minimizing noticeable alterations to the image quality.
- 4. Key:** A key is used as a parameterization tool for the embedding algorithm. It enables the generation of unique stego images and facilitates secure communication between the sender and receiver. Without the key, extracting the hidden message becomes challenging.
- 5. Stego Object:** The resultant image after embedding the secret message. The stego image appears identical to the original cover image to casual viewers but contains hidden information accessible only to authorized parties with knowledge of the key.
- 6. Extraction Algorithm:** The extraction algorithm is used to retrieve the secret message from the stego image. It operates based on complementary rules to the embedding algorithm, allowing for the accurate extraction of the hidden information without corrupting the original image content.

By integrating these components, the basic steganography system model for images facilitates secure and covert communication channels, enabling parties to exchange confidential information discreetly within seemingly innocuous image files.

Every data hiding technique in image steganography consists of two essential components:

- 1. Embedding Algorithm:** This algorithm conceals the secret message within a cover image file. It operates based on predefined rules and parameters to seamlessly integrate the secret message while minimizing noticeable alterations to the cover object's visual appearance.
- 2. Extraction Algorithm:** The extraction algorithm retrieves the hidden message from the stego image. It operates based on complementary rules to the embedding algorithm, ensuring the accurate extraction of the secret information without corrupting the original cover image.

A key is utilized to safeguard the embedding process, ensuring that only individuals with access to the secret key can retrieve the hidden message. This key serves as a parameterization tool, facilitating secure communication between the sender and receiver.

Features of Image Steganography:

The effectiveness of data hiding techniques in image steganography relies on several crucial properties tailored to the intended application. These properties include robustness, capacity, and undetectability, each defined as follows:

A. Robustness:

Robustness refers to the ability of embedded data to withstand various modifications, including compression, resizing, and other image processing operations. A robust steganographic method should withstand attacks on the embedding scheme, ensuring the hidden message remains intact despite potential alterations to the carrier image.

B. Capacity:

Capacity denotes the amount of information that can be concealed within the cover image relative to its size. Image steganography methods prioritize high data hiding capacity while minimizing visual artifacts or distortions introduced to the carrier image. Techniques such as LSB insertion and noise manipulation are commonly employed to maximize capacity while maintaining visual quality.

C. Undetectability:

Undetectability is essential for ensuring secure covert communication. A steganography technique should embed the secret message within the cover image without introducing noticeable changes that could reveal the presence of hidden data. Techniques that preserve the statistical properties of the cover image, such as utilizing LSB insertion in areas of low visual significance, contribute to undetectability.

Methods of Image Steganography

Understanding the mediums through which images are transmitted can aid in selecting the most suitable technique for hiding data within images. Below are some techniques commonly used in image steganography:

- **LSB Substitution:** LSB (Least Significant Bit) substitution involves replacing the least significant bits of pixel values in an image with the secret data. This method is straightforward but may result in distortion during image processing, compression, or conversion between different formats.

- **Phase Coding:** In phase coding, the phase of the image signal is adjusted to encode data, similar to its application in audio steganography. Smooth phase shifts are less likely to be detected by observers, making this method effective in terms of maintaining image quality. However, it typically supports lower data rates compared to other techniques.
- **Spread Spectrum Coding (SSC):** SSC techniques spread the secret data across the frequency domain of the image signal, similar to their application in audio steganography. This method offers resistance to variations and allows for data embedding without noticeable degradation of the cover image. However, some SSC methods may introduce additional noise to the image.
- **Echo Data Hiding:** Echo data hiding involves embedding data in an image signal by adjusting specific parameters related to echoes. While this method can be effective, it is less commonly used in image steganography compared to other techniques.

Related Methods for Image Steganography

While most research in image steganography has focused on techniques specific to image formats, there have been some developments in using popular image formats like JPEG as cover images for steganography. Below are some related methods:

Embedding in the Image Header:

- Unused Header Bit Stuffing (UHBS):** This technique utilizes unused fields in the image file header, such as metadata or reserved bits, to conceal a secret message. It offers good capacity for hiding data but may have lower robustness against detection.
- Padding Byte Stuffing (PBS):** PBS allows for the insertion of additional data bytes into padding areas within the image file, providing increased storage capacity for hidden information.
- Embedding Within Image Data:** Methods like the M4M (Message for Message) approach involve modifying the pixel values or color components of the image data to embed secret information, offering an alternative to traditional steganography techniques.

CHAPTER – 2

THEORETICAL FOUNDATIONS AND LITERATURE REVIEW

2.1. Below is the summary of bench mark papers in image steganography:

1. Fridrich, J., Goljan, M., & Du, R. (2001). Detecting LSB steganography in color, and gray-scale images. *IEEE Multimedia*, 8(4), 22-28.

• **Summary:** This paper introduces techniques for detecting steganography in both color and grayscale images, focusing on the least significant bit (LSB) embedding method. The authors propose statistical methods to identify the presence of hidden data by analyzing the distribution of LSB values. Their approach aims to improve detection accuracy in various types of images, enhancing security against covert communication.

2. Alattar, A. M. (2004). Reversible watermark using the difference expansion of a generalized integer transform. *IEEE Transactions on Image Processing*, 13(8), 1147-1156.

• **Summary:** Alattar presents a reversible watermarking technique based on the difference expansion of a generalized integer transform. The method allows for embedding and extracting watermarks without any loss of data or distortion to the original image. By exploiting mathematical properties of integer transforms, this approach achieves robustness against common attacks while ensuring reversibility, making it suitable for applications requiring both data hiding and recovery.

3. Bender, W., Gruhl, D., Morimoto, N., & Lu, A. (1996). Techniques for data hiding. *IBM Systems Journal*, 35(3.4), 313-336.

• **Summary:** This seminal paper explores various techniques for data hiding, laying the foundation for modern steganography. The authors discuss methods such as LSB insertion, transformation techniques, and masking algorithms, highlighting their strengths and limitations. Additionally, they examine applications of data hiding in multimedia content and address challenges in embedding and extracting hidden data effectively.

4. Cox, I. J., Miller, M. L., & Bloom, J. A. (2002). Digital watermarking and steganography. Morgan Kaufmann.

• **Summary:** This comprehensive book provides an in-depth overview of digital watermarking and steganography techniques. The authors cover fundamental principles, algorithms, and applications in multimedia security. Topics include robust watermarking, covert communication, detection methods, and the integration of cryptography with steganography. The book serves as a valuable resource for researchers, practitioners, and students interested in multimedia security.

5. Fridrich, J. (2009). Steganography in digital media: Principles, algorithms, and applications. Cambridge University Press.

• **Summary:** Fridrich's book offers a comprehensive examination of steganography principles, algorithms, and applications in digital media. The text covers various steganographic techniques, including LSB embedding, transformation methods, and adaptive hiding strategies. Additionally, it discusses applications in image, audio, and video steganography, along with detection and countermeasures against covert communication. The book serves as a thorough reference for understanding the theory and practice of steganography.

6. Johnson, N. F., & Jajodia, S. (1998). Exploring steganography: Seeing the unseen. IEEE Computer, 31(2), 26-34.

• **Summary:** In this paper, Johnson and Jajodia explore the concept of steganography and its implications for information security. They discuss historical and modern techniques for hiding information within digital media, emphasizing the importance of understanding covert communication methods for cybersecurity. The authors highlight challenges in detecting hidden data and propose strategies for mitigating risks associated with steganography.

7. Katzenbeisser, S., & Petitcolas, F. A. P. (2000). Information hiding techniques for steganography and digital watermarking. Artech House.

• **Summary:** This book provides a comprehensive overview of information hiding techniques for steganography and digital watermarking. The authors cover theoretical foundations, practical algorithms, and real-world applications in multimedia security. Topics include embedding methods, robustness considerations, detection algorithms, and applications in copyright protection and authentication. The book serves as a valuable resource for researchers, practitioners, and professionals working in cybersecurity and multimedia forensics.

8. Petitcolas, F. A. P., Anderson, R. J., & Kuhn, M. G. (1999). Information hiding—A survey. *Proceedings of the IEEE*, 87(7), 1062-1078.

• **Summary:** This survey paper provides a comprehensive overview of information hiding techniques, including steganography and digital watermarking. The authors discuss historical developments, fundamental concepts, and current research trends in concealing data within digital media. They explore various embedding methods, detection algorithms, and applications in multimedia security, offering insights into the challenges and opportunities in information hiding research.

9. Provos, N., & Honeyman, P. (2003). Detecting steganographic content on the internet. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy* (pp. 282-294).

• **Summary:** Provos and Honeyman present techniques for detecting steganographic content on the internet, addressing concerns related to covert communication and digital forensics. The paper discusses statistical analysis, payload inspection, and traffic analysis methods for identifying hidden data within network traffic and online content. The authors highlight the importance of robust detection mechanisms in combating illicit activities and maintaining cybersecurity on the internet.

10. Westfeld, A., & Pfitzmann, A. (1999). Attacks on steganographic systems. In *International Workshop on Information Hiding* (pp. 61-76). Springer.

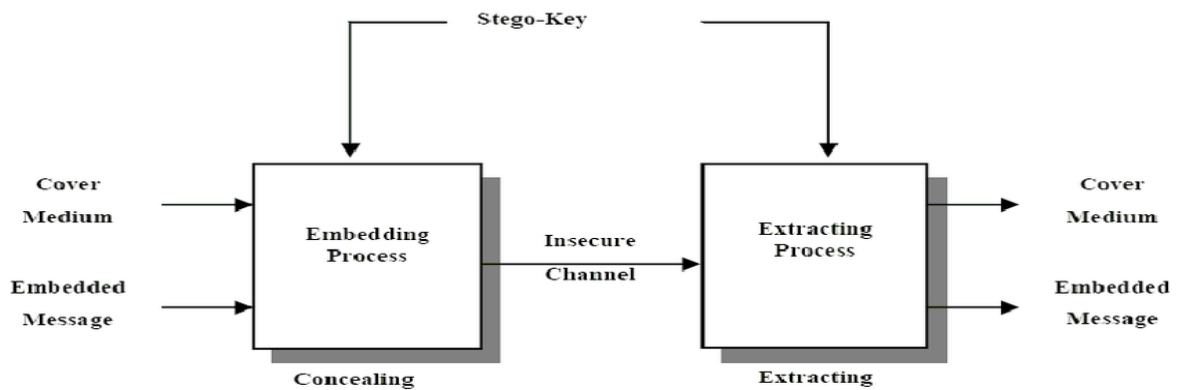
• **Summary:** This paper explores various attacks on steganographic systems, aiming to evaluate their security and resilience against adversarial threats. The authors discuss theoretical vulnerabilities, practical exploits, and countermeasures in concealing data within digital media. By analyzing attack vectors and defense strategies, the paper provides insights into the strengths and limitations of existing steganographic techniques, informing future research directions in multimedia security and privacy.

CHAPTER – 3

SYSTEM ANALYSIS & FEASIBILITY STUDY

3.1. Existing Method:

In the existing system, the existing model uses image steganography in securing embedded data for various applications, existing techniques often overlook message integrity. The widely used Least Significant Bit (LSB) method lacks encryption prior to embedding secret messages, rendering them susceptible to interception or manipulation during transmission.



3.2. Disadvantages:

- The existing system lacks more security.
- It is less secure and weak method of information exchange.
- Confidential and private data is not protected.

3.3. Proposed Method:

The proposed system, we proposed a system in light of the limitations of the LSB method, this study proposes the development of an advanced image steganography method, leveraging LSB substitution alongside RSA encryption. The primary objective is to bolster the security aspects of image steganography while also integrating measures for verifying message integrity.

We proposed RSA algorithm which inspires from the PQC algorithm which deals with the quantum computers and give our model a great level of security.

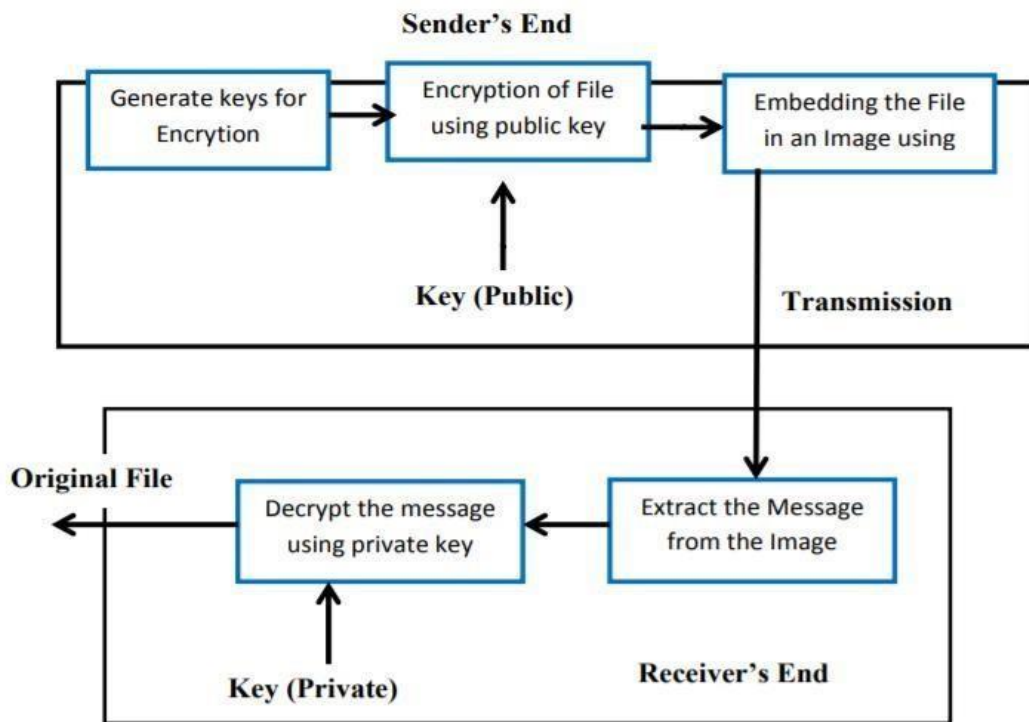


Figure 1: The Proposed Model

Fig 3: Existing & Proposed System

3.4. Advantages:

- The existing system converges both the cryptographic and steganography models which gives us more security.
- It is more secure and robust method of information exchange so that confidential and private data must be protected against the traditional and some of the small quantum attacks.
- Resistant from traditional and quantum attacks.

CHAPTER-4

SYSTEM REQUIREMENTS

4.1. HARDWARE REQUIREMENTS:

1. Computer/server with adequate processing power and RAM
2. Storage space for MATLAB, data, and output
3. Optional: GPU for acceleration

4.2. SOFTWARE REQUIREMENTS:

1. MATLAB with Image Processing Toolbox
2. Optional: Cryptography Toolbox
3. Post-Quantum Cryptography Libraries (if needed)
4. Operating System compatible with MATLAB
5. Image Editing Software

MATLAB With Image Processing Toolbox:

MATLAB's Image Processing Toolbox is a powerful tool for performing various image processing tasks, from basic operations like image enhancement and filtering to more advanced tasks like object detection and segmentation. Here's a brief overview of some common functionalities:

- 1. Image Loading and Display:** You can easily load images in various formats such as JPEG, PNG, or BMP using functions like `imread()`, and display them using `imshow()`.
- 2. Image Enhancement:** MATLAB provides functions for enhancing image quality, such as `imadjust()` for adjusting image intensity values or `histeq()` for histogram equalization.
- 3. Image Filtering:** Various filters can be applied to images for tasks like noise reduction, edge detection, or smoothing. Functions like `imfilter()` allow you to apply custom filters or use built-in ones like Gaussian or Sobel filters.
- 4. Image Segmentation:** Segmentation divides an image into meaningful regions. MATLAB offers functions like `imsegkmeans()` for K-means clustering-based segmentation or `activecontour()` for active contour segmentation.
- 5. Object Detection and Recognition:** You can use built-in functions like `detectMSERFeatures()` for detecting MSER (Maximally Stable Extremal Regions) features or train custom object detectors using machine learning algorithms like SVM (Support Vector Machine) or Haar cascades.

CHAPTER – 5

IMPLEMENTATION METHODOLOGY

Algorithm:

Encryption code description

Key Generation:

1. Prime numbers p and q are chosen.
2. Modulus n and Euler's totient function ϕ are calculated.
3. Public exponent e is set to 17, and it is verified that e and ϕ are coprime.
4. Private key d is computed using the modular inverse function modinv .
5. Key pairs (public and private keys) are stored.

File Selection:

1. The user is prompted to select a file using a dialog box.
2. If a file is selected, its full path is obtained.

Image Steganography:

1. The selected image file is read (C_{org}).
2. The image is converted into a 1D array (C).
3. The ASCII values of the array elements are obtained and converted to binary (CAs_{bin}).
4. The user inputs a secret message ($plaintext$).
5. The message is converted to ASCII values and encrypted using RSA (modexp).
6. The encrypted message is converted back to a text representation ($text$).
7. The text message is converted to binary ($text_{bin}$).
8. The LSBs of the image binary and text binary are replaced, creating a stego image ($signal_{adj}$).
9. The stego image is displayed alongside the original image.
10. The length of the encoded message is displayed.
11. The keys (d , n , $textlen$) are saved in a file named 'keys.mat'.

Helper Functions:

1. ``arrayToText``: Converts an array to a comma-separated string.
2. ``modinv``: Calculates the modular inverse of a number.
3. ``modexp``: Performs modular exponentiation

ALGORITHM:**Decryption description****Initialization:**

1. Clear Environment: Clears the command window, workspace, and closes all figures.
2. Load Keys: Loads the previously saved RSA private and public keys (``d``, ``n``, ``textlen``).

File Selection:

1. User Interaction: A dialog box prompts the user to select a stego image file.
2. File Check: If a file is selected, its full path is obtained.

Message Extraction and Decryption:

1. Image Reading and Conversion: The selected stego image (``Sorg``) is read.
2. Binary Conversion: The pixel values of the image are converted to binary.
3. Extract Text Bits: LSBs of the binary values are extracted to reconstruct the hidden text.
4. Text to ASCII Conversion: The binary text is converted back to ASCII values.
5. Decryption: The ASCII values are decrypted using RSA modular exponentiation (``modexp``).
6. ASCII to Text Conversion: The decrypted ASCII values are converted back to a text string.

Display Results:

1. Display Stego Image: The stego image is displayed using ``imshow``.
2. Display Decrypted Text: The decrypted text is displayed.

Helper Functions:

1. ``textToArray``: Splits the extracted text into individual values and converts them to an array.
2. ``modexp``: Performs modular exponentiation for decryption.
3. ``modinv``: Calculates the modular inverse of a number

CHAPTER - 6













SYSTEM DESIGN

6.1. CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages. Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

The following are the Class Diagram Components:

- ◆ Select - 
- ◆ Broom - 
- ◆ New Package - 
- ◆ New Class - 
- ◆ New Association - 
- ◆ New UnAssociation - 
- ◆ New Aggregation - 
- ◆ New UniAggregation - 
- ◆ New Composition - 
- ◆ New UniComposition - 
- ◆ New Generalization - 
- ◆ New Interface - 

- ◆ New Realization -
- ◆ New Dependency - \uparrow
- ◆ New Permission $\uparrow p$
- ◆ New Usage - $\uparrow u$

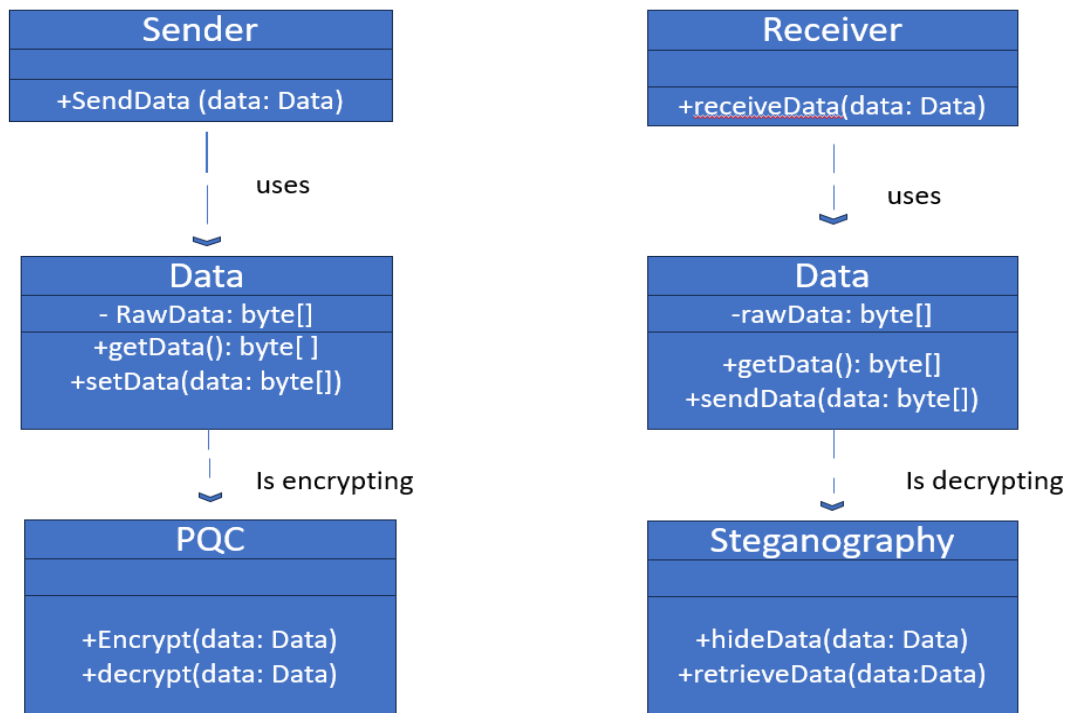


Fig 4: Class Diagram

6.2. USE CASE DIAGRAM

- A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis.
- Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.
- The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

- Use-case diagrams illustrate and define the context and requirements of either an entire system or the important parts of the system. You can model a complex system with a single use-case diagram, or create many use-case diagrams to model the components of the system. You would typically develop use-case diagrams in the early phases of a project and refer to them throughout the development process.

The following topics describe model elements in use-case diagrams:

Use cases

A use case describes a function that a system performs to achieve the user's goal. A use case must yield an observable result that is of value to the user of the system.

Actors

An actor represents a role of a user that interacts with the system that you are modeling. The user can be a human user, an organization, a machine, or another external system.





Subsystems

In UML models, subsystems are a type of stereotyped component that represent independent, behavioral units in a system. Subsystems are used in class, component, and use-case diagrams to represent large-scale components in the system that you are modeling.

Relationships in use-case diagrams

In UML, a relationship is a connection between model elements. A UML relationship is a type of model element that adds semantics to a model by defining the structure and behavior between the model elements.

The following are the Usecase Diagram Components:

- ◆ Actor - 
- ◆ New Use Case - 
- ◆ New Association - 
- ◆ New UniAssociation - 
- ◆ New Aggregation -

◆ New UniAggregation - ◇→

◆ New Composition - ◆—

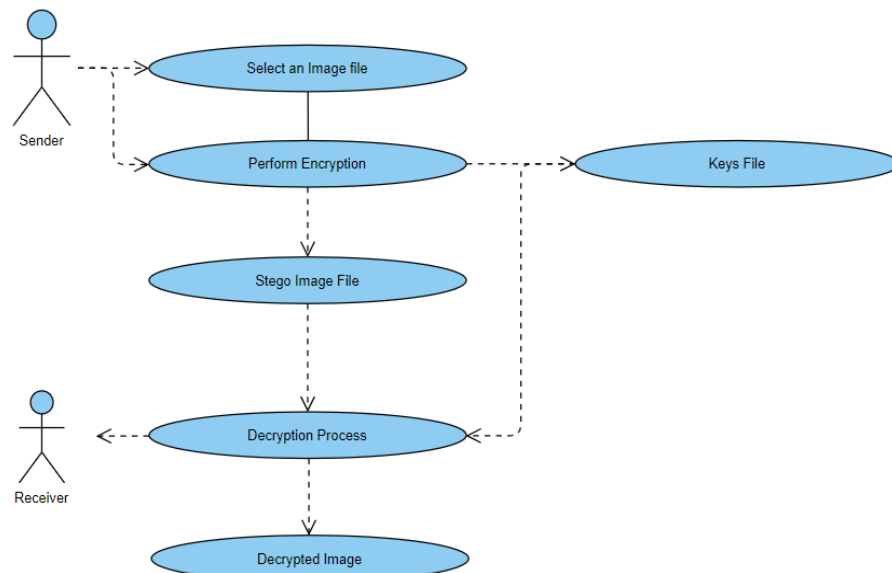


Fig 5: Use Case Diagram

6.3. SEQUENCE DIAGRAM

- ▶ A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order.
- ▶ It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams
- ▶ A sequence diagram shows the sequence of messages passed between objects. Sequence diagrams can also show the control structures between objects. For example, lifelines in a sequence diagram for a banking scenario can represent a customer, bank teller, or bank manager. The communication between the customer, teller, and manager are represented by messages passed between them. The sequence diagram shows the objects and the messages between the objects.

The following topics describe the elements in sequence diagrams:

Lifelines in UML diagrams

In UML diagrams, such as sequence or communication diagrams, lifelines represent the objects that participate in an interaction. For example, in a banking scenario, lifelines can represent objects such as a bank system or customer. Each instance in an interaction is represented by a lifeline.

Messages in UML diagrams

A message is an element in a Unified Modeling Language (UML) diagram that defines a specific kind of communication between instances in an interaction. A message conveys information from one instance, which is represented by a lifeline, to another instance in an interaction.

Combined fragments in sequence diagrams

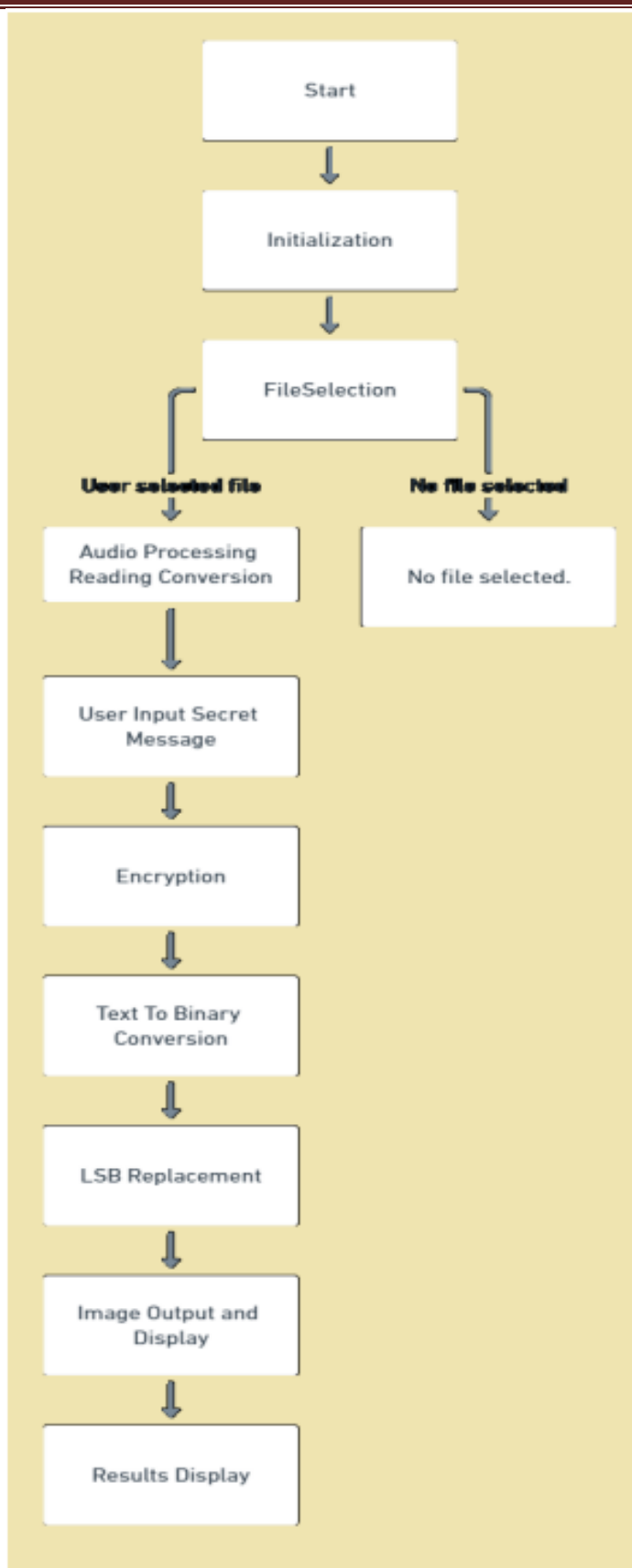
In sequence diagrams, combined fragments are logical groupings, represented by a rectangle, which contain the conditional structures that affect the flow of messages. A combined fragment contains interaction operands and is defined by the interaction operator.

Interaction uses in sequence diagrams

In sequence diagrams, interaction uses enable you to reference other existing interactions. You can construct a complete and complex sequence from smaller simpler interactions.

6.4. ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control. Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.



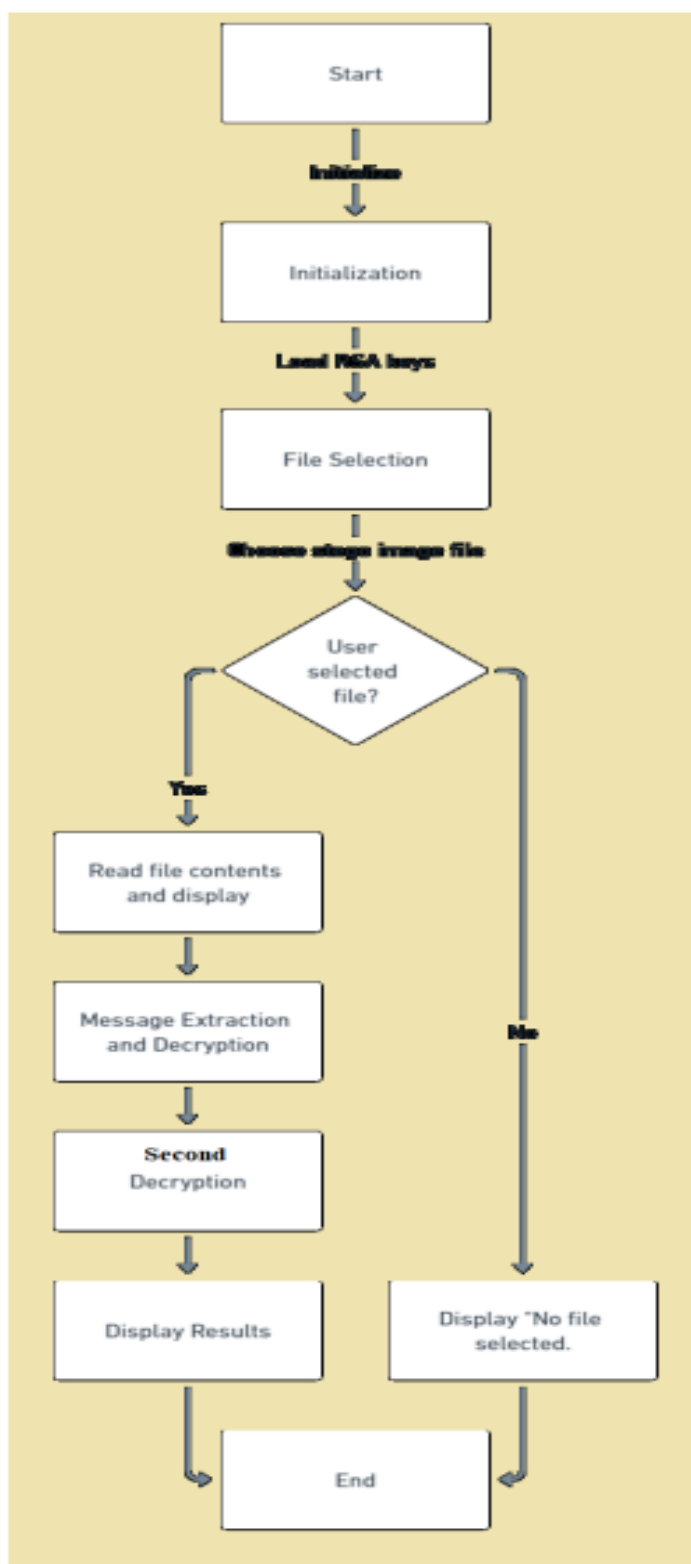


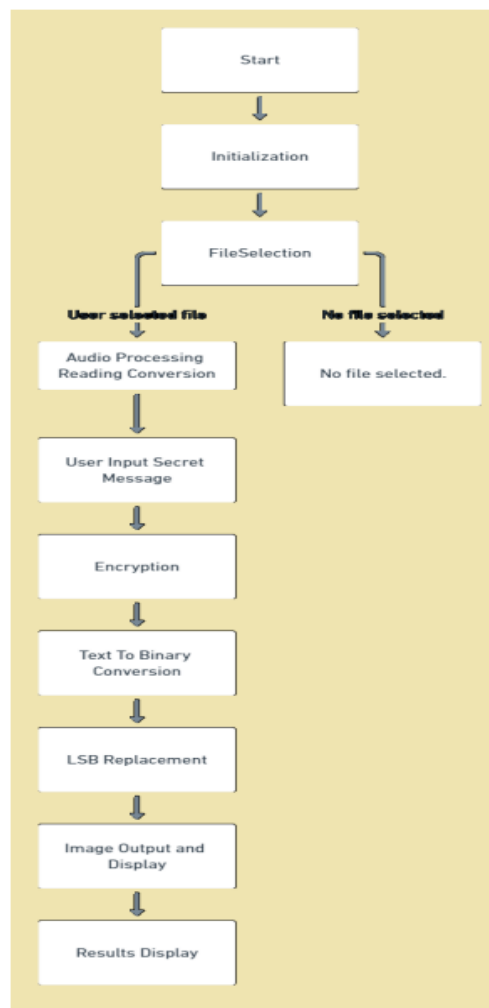
Fig 6: Activity Diagram

6.5. STATE DIAGRAM:

A state machine is any device that stores the status of an object at a given time and can change status or cause other actions based on the input it receives. States refer to the different combinations of information that an object can hold, not how the object behaves. In order to understand the different states of an object, you might want to visualize all of the possible states and show how an object gets to each state, and you can do so with a UML state diagram.

Each state diagram typically begins with a dark circle that indicates the initial state and ends with a bordered circle that denotes the final state. However, despite having clear start and end points, state diagrams are not necessarily the best tool for capturing an overall progression of events. Rather, they illustrate specific kinds of behavior—in particular, shifts from one state to another.

State diagrams mainly depict states and transitions. States are represented with rectangles with rounded corners that are labeled with the name of the state. Transitions are marked with arrows that flow from one state to another, showing how the states change. Below, you can see both these elements at work in a basic diagram for student life.



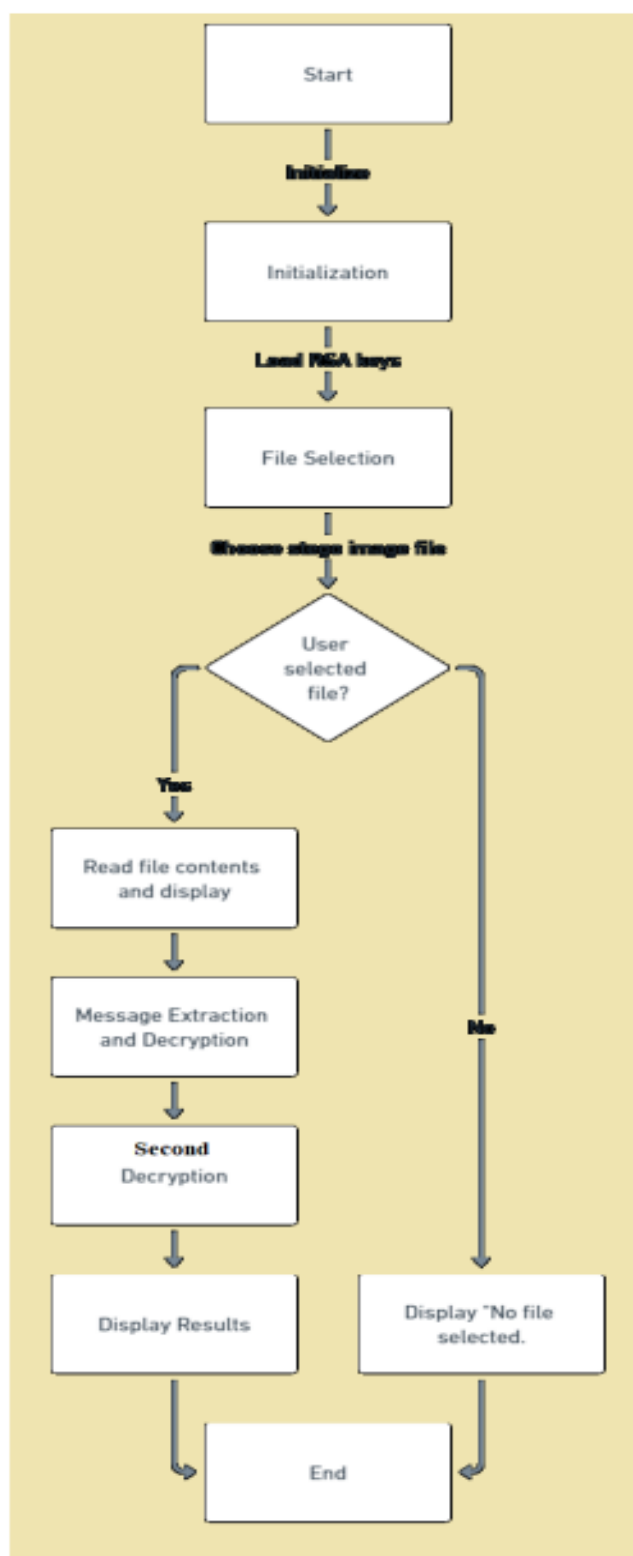


Fig 7: State Diagram

CHAPTER – 7

STATISTICAL METHODS IN IMAGE STEGANOGRAPHY

Signal-to-Noise Ratio (SNR) in Steganography: In the context of image steganography employing RSA encryption and LSB substitution, the signal-to-noise ratio (SNR) remains a fundamental metric for assessing the quality of the stego-image. The SNR calculation involves the following steps:

- 1. Analysis of Original Cover Image:** The root mean square (RMS) value of the original cover image is determined, representing its average power.
- 2. Calculation of Maximum Allowable Distortion:** Based on the RMS value of the cover image, the maximum allowable distortion or noise that can be introduced before becoming perceptible to human observers is computed.
- 3. Generation of Stego-Image:** The secret message is encrypted using RSA encryption and embedded into the LSBs of the cover image, generating the stego-image.
- 4. Analysis of Stego-Image:** The RMS value of the stego-image is calculated, representing the total power of the stego-object.
- 5. SNR Calculation:** The SNR is computed by dividing the RMS value of the original cover image by the RMS value of the distortion or noise introduced during embedding, expressed in decibels (dB). The SNR formula remains the same as in audio steganography, with a higher SNR indicating better quality of the stego-image and lower detectability of the hidden message.

Sample Pairs Cross Correlation (SPCC) in Image Steganography: SPCC (Sample Pairs Cross Correlation) can be adapted to measure the correlation between pairs of pixel values in a cover image, determining the maximum allowable embedding rate without noticeable distortion. The formula for SPCC calculation remains similar: $SPCC = (1/N) * \sum(x[i, j] * x[i+m, j+n])$

Where:

- N is the total number of pixels in the cover image.
- $x[i, j]$ and $x[i+m, j+n]$ are pixel values at positions (i, j) and (i+m, j+n), respectively.

- m and n represent the distance between pixel pairs, typically set to 1. SPCC values range from -1 to 1, with higher values indicating lower randomness in the cover image, making it less suitable for embedding.

BER in Image Steganography: BER remains relevant in image steganography using RSA encryption and LSB substitution, measuring the accuracy of the embedded secret message compared to the original message.

CHAPTER – 8

SYSTEM CODING

8.1. SAMPLE CODE

ENCRYPTION:

encryption.m

```

function my_image_rsa_encryption_gui
close all
% Create a figure window
fig = figure('Name', 'PQC Encryption GUI', 'Position', [100, 100, 800, 600],
'MenuBar', 'none');

% Add components to the figure
uicontrol('Style', 'pushbutton', 'String', 'Select File', 'Position', [50, 500, 100, 30],
'Callback', @selectFile);
uicontrol('Style', 'edit', 'String', '', 'Position', [180, 500, 400, 30], 'Tag', 'fileEdit');
uicontrol('Style', 'text', 'String', 'Enter P value:', 'Position', [50, 450, 100, 30]);
uicontrol('Style', 'edit', 'String', '', 'Position', [180, 450, 100, 30], 'Tag', 'pEdit');
uicontrol('Style', 'text', 'String', 'Enter Q value:', 'Position', [300, 450, 100, 30]);
uicontrol('Style', 'edit', 'String', '', 'Position', [430, 450, 100, 30], 'Tag', 'qEdit');
uicontrol('Style', 'pushbutton', 'String', 'Encrypt', 'Position', [600, 450, 100, 30],
'Callback', @encryptMessage);
uicontrol('Style', 'text', 'String', 'Enter Secret Message:', 'Position', [50, 400, 150, 30]);
uicontrol('Style', 'edit', 'String', '', 'Position', [180, 400, 400, 30], 'Tag', 'messageEdit');
uicontrol('Style', 'pushbutton', 'String', 'Close', 'Position', [700, 50, 70, 30], 'Callback',
'close(gcf)');

% Add axes for waveform plot
axes('Position', [0.1, 0.1, 0.4, 0.45], 'Tag', 'orgimg');
axes('Position', [0.5, 0.1, 0.4, 0.45], 'Tag', 'stgimg');

% Callback for Select File button
function selectFile(~, ~)
[filename, path] = uigetfile('*.jpg', 'Select a file');
if filename ~= 0
fileEdit = findobj('Tag', 'fileEdit');
set(fileEdit, 'String', fullfile(path, filename));
end
end

% Callback for Encrypt button
function encryptMessage(~, ~)
% Get file path, p, q, and secret message from GUI

```

```

fileEdit = findobj('Tag', 'fileEdit');
pEdit = findobj('Tag', 'pEdit');
qEdit = findobj('Tag', 'qEdit');
messageEdit = findobj('Tag', 'messageEdit');
filepath = get(fileEdit, 'String');
p = str2double(get(pEdit, 'String'));
q = str2double(get(qEdit, 'String'));
secretMessage = get(messageEdit, 'String');

% Check if file path is empty
if isempty(filepath)
    disp('Please select a file. ');
    return;
end

% Check if p or q is empty or not a number
if isnan(p) || isnan(q)
    disp('Please enter valid values for P and Q. ');
    return;
end

% Perform encryption

performEncryption(filepath, p, q, secretMessage);
end

% Function to perform encryption
function performEncryption(filepath, p, q, secretMessage)
    % Public Key Generation
    n = p * q; % modulus
    phi = (p-1) * (q-1); % Euler's totient function
    e = 17; % public exponent

    % Ensure e and phi are coprime
    while gcd(e, phi) ~= 1
        e = e + 2;
    end

    % Private Key Calculation
    d = modinv(e, phi); % modinv is a custom function for modular inverse

    % Key Pair
    public_key = [n, e];
    private_key = [n, d];

    Corg = imread(filepath); % cover file
    [nrow, ncol, colr] = size(Corg)

```

```

C = Corg(:);

CAs = C;

CAs_bin = dec2bin(CAs,8);
% % convert text to binary and pad with zeros if necessary
%text = 'This is the hidden text what are doing.';

% Prepare secret message for encryption
message = double(secretMessage); % Convert string to ASCII values
ciphertext = modexp(message, e, n); % modexp is a custom function for modular
exponentiation
ciphertext = (round(ciphertext))

CAs_bin = dec2bin(CAs, 8);

% Embed ciphertext into cover audio
plaintext = secretMessage; % Store plaintext for later use
text = arrayToText(ciphertext);
textlen = length(text);
text_bin = dec2bin(text, 8);
text_bin = text_bin(:)';
signal_bin = CAs_bin;
signal_bin = signal_bin - 48;
text_bin = text_bin - 48;
for k = 1 : length(text_bin)
    temp = signal_bin(k,:);
    temp(7) = text_bin(k);
    signal_bin(k,:) = temp;
end

signal_int = bin2dec(num2str(signal_bin));
signal_adj = uint8(signal_int);
signal_adj = reshape(signal_adj,nrow,ncol,colr);
imwrite(signal_adj,'stego.bmp','bmp');

% Plot waveform
orgimg = findobj('Tag', 'orgimg');
imshow(Corg, 'Parent', orgimg);
title(orgimg, 'Cover Image');
drawnow

stgimg = findobj('Tag', 'stgimg');
imshow(signal_adj, 'Parent', stgimg);

```



```
title(stgimg, 'Stego Image');
drawnow

% %      hold on;
% %      plot(waveformPlot, signal_adj, 'r');
%      ylim(waveformPlot, [-2, 2]);
%      title(waveformPlot, 'Waveform Plot');
%      hold off;

% Display notification in a separate message box
msgbox(sprintf('Encryption is over. Note this number Required for decoding: %d',
textlen), 'Encryption Completed');
%
%      % Display notification
%      notification = sprintf('Encryption is over. Note this number Required for
decoding: %d', textlen);
%      disp(notification);

% for error calculation
signalPower = mean(Corg(:).^2);
noisy_signal_power = mean(signal_adj(:).^2);

% Compute noise power (using mean squared pixel values)
noisePower = signalPower - noisy_signal_power;

% Calculate SNR in decibels
SNR_val = 10 * log10(signalPower / noisePower)

CAs_double = double(CAs);
signal_adj_1_double = double(signal_adj);

% Compute Pearson correlation coefficient
SPCC_val = corr(double(Corg(:)), double(signal_adj(:)))

%SPCC_val = corr(CAs, signal_adj_1);
save ('imagekeys.mat','d','n','textlen','text','SNR_val','SPCC_val','plaintext');

end

% Modular Inverse Function
function inv = modinv(a, m)
    m0 = m;
    y = 0;
```

```

x = 1;

if m == 1
    inv = 0;
    return;
end

while a > 1
    q = idivide(int32(a), int32(m), 'floor');
    t = m;
    m = mod(int32(a), int32(m));
    a = t;
    t = y;
    y = x - q * y;
    x = t;
end

if x < 0
    x = x + m0;
end

inv = x;
end

% Modular Exponentiation Function
function result = modexp(base, exponent, modulus)
    result = 1;
    base = mod(base, modulus);
    while exponent > 0
        if bitget(exponent, 1)
            result = mod(result .* base, modulus);
        end
        base = mod(base .* base, modulus); % Use element-wise multiplication
        exponent = bitshift(exponent, -1);
    end
end

% Function to convert array to text
function textData = arrayToText(secret_data)
    % Convert array to a string with comma-separated values
    textData = sprintf('%d, ', secret_data);
    textData = textData(1:end-2); % Remove the trailing comma and space
end
end

```

RESULTS:

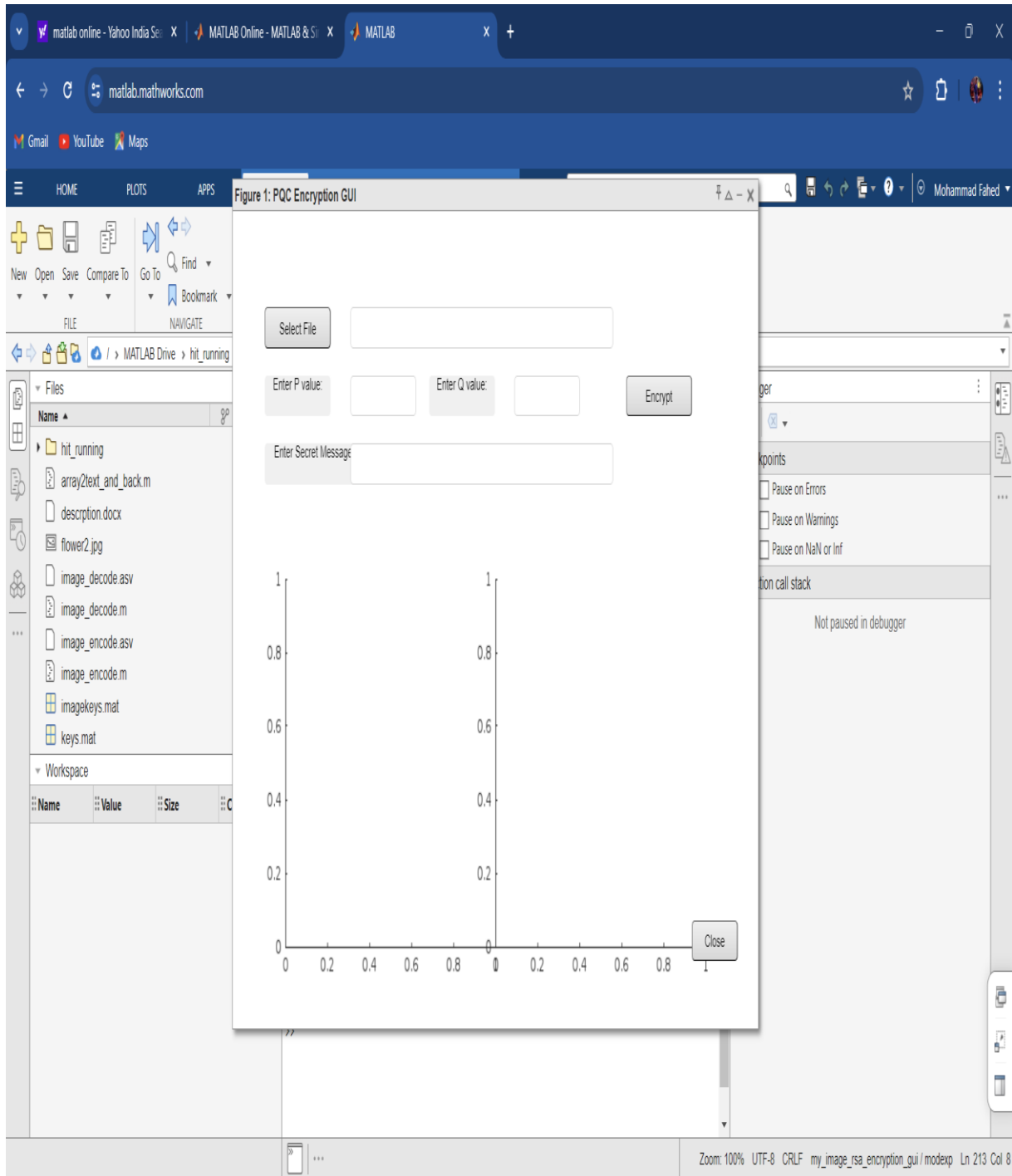


Fig 8: PQC Encryption GUI

SAMPLE CODE:**DECRYPTION****decryption.m**

```
function my_image_rsa_decryption_gui
close all
    % Clear conflicting variable from the workspace
    global SNR_val
    global SPCC_val
    global n
    global d
    global plaintext
    global text
    global textlen

    % Load decryption keys and other necessary data

    load('imagekeys.mat');

    % Create a figure window
    fig = figure('Name', 'PQC Decryption GUI', 'Position', [100, 100, 800, 600],
'MenuBar', 'none');

    % Add components to the figure
    uicontrol('Style', 'pushbutton', 'String', 'Select File', 'Position', [50, 500, 100, 30],
'Callback', @selectFile);
    uicontrol('Style', 'edit', 'String', '', 'Position', [180, 500, 400, 30], 'Tag', 'fileEdit');
    uicontrol('Style', 'text', 'String', 'Enter Number:', 'Position', [50, 450, 100, 30]);
    uicontrol('Style', 'edit', 'String', '', 'Position', [180, 450, 100, 30], 'Tag', 'numberEdit');
    uicontrol('Style', 'pushbutton', 'String', 'Decrypt', 'Position', [600, 450, 100, 30],
'Callback', @decryptMessage);
    uicontrol('Style', 'pushbutton', 'String', 'Close', 'Position', [700, 50, 70, 30], 'Callback',
'close(gcf)');

    % Add axes for waveform plot
    axes('Position', [0.1, 0.1, 0.8, 0.55], 'Tag', 'imgdisp');

    % Callback for Select File button
    function selectFile(~, ~)
        [filename, path] = uigetfile('*.mat', 'Select a file');
        if filename ~= 0
            fileEdit = findobj('Tag', 'fileEdit');
            set(fileEdit, 'String', fullfile(path, filename));
        end
    end
end
```

```

% Callback for Decrypt button
function decryptMessage(~, ~)
    % Get file path and number from GUI
    fileEdit = findobj('Tag', 'fileEdit');
    numberEdit = findobj('Tag', 'numberEdit');
    filepath = get(fileEdit, 'String');
    textlen = str2double(get(numberEdit, 'String'));

    % Check if file path is empty
    if isempty(filepath)
        disp('Please select a file. ');
        return;
    end

    % Check if the entered value is a valid number
    if length(textlen) <= 0
        disp('Please enter a valid positive number. ');
        return;
    end

    % Perform decryption
    performDecryption(filepath, textlen);
end

% Function to perform decryption
function performDecryption(filepath, textlen)
    % Load decryption keys and other necessary data
    % load('keys.mat');

    % Read the stego img file
    Sorg = imread(filepath); % cover file

    y = Sorg(:);
    ybin = dec2bin(y, 8);
    wavmat = ybin - 48;
    txtbits = [];
    for k = 1 : textlen * 8
        temp = wavmat(k, :);
        txtbits = [txtbits temp(7)];
    end
    txt_int_bits = reshape(txtbits, [], 8);
    txt_int = bin2dec(num2str(txt_int_bits));
    extracted_text = char(txt_int)

    % Convert extracted text to array
    extracted_array = textToArray(extracted_text);

```

```
% Decryption
decrypted_message = modexp(extracted_array, d, n);

% Convert ASCII values back to string
decrypted_text = char(decrypted_message)
%plaintext

% Calculate Bit Error Rate
Bit_error_rate = ((decrypted_message - plaintext) / plaintext) * 100;

% Plot image
imgdisp = findobj('Tag', 'imgdisp');
imshow(Sorg, 'Parent', imgdisp);
title(imgdisp, 'Stego image');

% Display decrypted text and other values in msgbox
msg = {sprintf('Decrypted Text:\n%s', decrypted_text), ...
    sprintf('SNR Value: %.2f', SNR_val), ...
    sprintf('SPCC Value: %.2f', SPCC_val), ...
    sprintf('Bit Error Rate: %.2f%%', Bit_error_rate)};
msgbox(msg, 'Decryption Results');
end

% Function to convert extracted text to array
function extractedArray = textToArray(extractedText)
% Split the text into individual values and convert them to an array
values = str2double(strsplit(extractedText, ' '));
extractedArray = values(~isnan(values));
end

% Modular Exponentiation Function
function result = modexp(base, exponent, modulus)
result = 1;
base = mod(base, modulus);
while exponent > 0
    if bitget(exponent, 1)
        result = mod(result .* base, modulus);
    end
    base = mod(base .* base, modulus); % Use element-wise multiplication
    exponent = bitshift(exponent, -1);
end
end
end
```

RESULTS

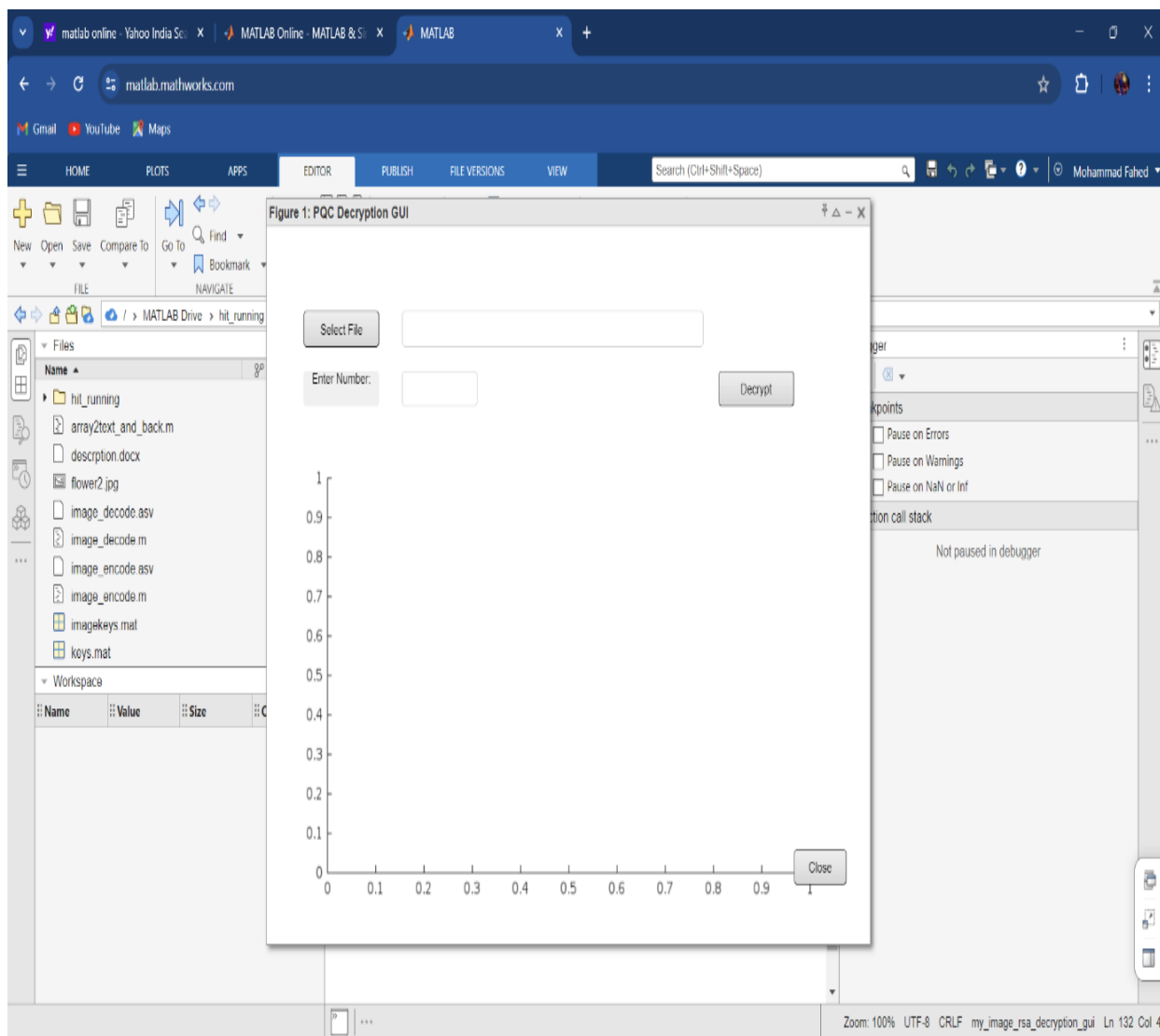


Fig 9: PQC Decryption GUI

8.2. About programming Language

What is MATLAB?

- MATLAB excels at matrix manipulations, linear algebra, and signal processing.
- It offers powerful tools for creating informative graphs and charts to visualize data.
- MATLAB allows you to write scripts and functions to automate tasks and extend its capabilities.
- MATLAB provides specialized toolboxes for various domains like control systems, machine learning, and image processing.
- MATLAB provides an interactive environment for developing code, visualizing results, and debugging errors.
- MATLAB can integrate with other programming languages and software, such as Simulink for model-based design.

Why Use MATLAB

- MATLAB excels at working with matrices and performing complex mathematical operations. Its syntax is designed to closely resemble mathematical notation, making it easy for users with a strong math background to pick up.
- MATLAB boasts a vast library of toolboxes that offer pre-written functions and tools for specific tasks in engineering, science, and even finance. These toolboxes save users time and effort by providing solutions without needing to code everything from scratch.
- MATLAB offers powerful built-in tools for creating publication-quality graphs and plots. These plots can be easily customized and integrated into reports or presentations.
- MATLAB is optimized for speed and efficiency, particularly when dealing with numerical computations. It can leverage your computer's hardware, including multicore processors and GPUs, to accelerate calculations.
- MATLAB has been around for decades and is a widely used platform in academia and industry. This means there's a large community of users and extensive documentation and resources available for support.

MATLAB Installation

- **MathWorks Account:** You'll need a MathWorks account to download and license MATLAB. You can create a free account if you don't have one already.
- **License:** To use MATLAB, you'll need a valid license. This could be a personal license you purchase or one provided by your university or workplace.

Installation Steps:

1. Download MATLAB:

- Sign in to your MathWorks account on MathWorks website: <https://www.mathworks.com/>.
- Go to the "Downloads" section and select the desired MATLAB release (e.g., R2024a) that aligns with your license.
- Choose the installer corresponding to your operating system (Windows, Mac, Linux).

2. Run the installer:

- Once the download is complete, follow the instructions for your OS:
- **Windows:** Double-click the downloaded executable and follow the on-screen prompts.
- **Mac:** Double-click the downloaded DMG file to start the installation.
- **Linux:** Unzip the downloaded installer and follow the command-line instructions provided by MathWorks.

3. License Activation:

- During installation, you might be prompted to sign in to your MathWorks account to activate your license.
- Follow the instructions provided to complete the activation process.

Additional Resources:

- MathWorks Installation Guide: <https://www.mathworks.com/help/install/install-products.html>
- How to Install MATLAB video tutorial: <https://www.mathworks.com/help/install/install-products.html>

MATLAB Comments

- In MATLAB, you use the percent sign (%) to add comments to your code. These comments are ignored by MATLAB when running the script.

MATLAB Variables

Storage and Naming:

- Variables store data in MATLAB. They can hold numbers, text, or even entire matrices.
- Variable names must start with a letter and can contain letters, numbers, and underscores

Assigning Values:

- You assign values to variables using the equal sign (=). For example:

Matlab

`x = 10; % Assigns the number 10 to variable x`

`name = 'Alice'; % Assigns the text 'Alice' to variable name.`

Data Types:

- Variables can hold different data types. Common ones include:
- **Double:** Default numeric type for floating-point numbers (decimals).
- **Character:** Stores text data as a sequence of characters.
- **Logical:** Stores True (1) or False (0) values.
- **Matrix:** A two-dimensional array of numbers.

MATLAB Identifiers

- Identifiers are names you give to variables, functions, and other objects in your MATLAB code.
- They act like labels that help you understand and reference these elements in your program.

Naming Rules:

- Must start with a letter (a-z or A-Z).
- Can contain letters, numbers, and underscores (_).
- Case-sensitive (e.g., x is different from X).
- Cannot be MATLAB keywords (reserved words with specific meanings).

Examples of valid identifiers:

- myVariable
- data_2023
- count
- _temporary

Examples of invalid identifiers:

- 1st_place (cannot start with a number)
- if (MATLAB keyword)
- %commented (cannot contain comments)

MATLAB Functions

- MATLAB functions are reusable blocks of code that perform specific tasks. They can take inputs, perform calculations, and return outputs.
- Functions improve code readability, modularity, and reusability.
- There are two main types of functions in MATLAB:
- **Built-in functions:** These are functions that come pre-defined with MATLAB. They cover a wide range of mathematical, graphical, and data analysis tasks.
- **User-defined functions:** These are functions that you create yourself to perform specific tasks. You can define functions in M-files (script files) or function files.

MATLAB Object-Oriented Programming (OOP)

- MATLAB **supports object-oriented programming (OOP)**, which is a programming paradigm that allows you to create objects that encapsulate data (properties) and the code to operate on that data (methods).
- OOP concepts in MATLAB include classes, objects, inheritance, polymorphism, and encapsulation.
- Using OOP in MATLAB can improve code organization, maintainability, and reusability.

MATLAB Standard Libraries

- MATLAB Standard Libraries, also known as toolboxes, are collections of pre-written functions, data files, and other tools that extend the functionality of MATLAB.
- Toolboxes cover a wide range of application areas, such as signal processing, control systems, image processing, communications, computational finance, and more.
- Some popular standard libraries include:
 - Signal Processing Toolbox
 - Control System Toolbox
 - Image Processing Toolbox
 - Communications Toolbox
 - Optimization Toolbox
 - Statistics and Machine Learning Toolbox

Libraries for Cryptography

While MATLAB itself doesn't have a built-in cryptography toolbox, you can use the Symbolic Math Toolbox to perform some basic cryptographic operations like encryption and decryption using simple ciphers.

- For more advanced cryptography functionalities, you can integrate MATLAB with external libraries written in C/C++ or Python that offer robust cryptographic algorithms.

MATLAB Usage

MATLAB is used in a wide range of fields, including:

- Engineering: MATLAB is widely used by engineers for tasks like signal processing, control systems design, simulation, and data analysis.
- Science: Scientists use MATLAB for scientific computing, modeling, simulation, data analysis, and visualization.
- Mathematics: MATLAB is a powerful tool for mathematicians for performing symbolic and numerical computations, solving equations, and visualizing mathematical concepts.
- Finance: MATLAB is used in finance for quantitative analysis, risk modeling, portfolio optimization, and algorithmic trading.
- Other fields: MATLAB is also used in other fields like computer science, economics, chemistry, biology, and more.

MATLAB Files and Streams

- MATLAB uses different file types to store data, code, and other information. Some common file types include:
- M-files (.m): These are script files that contain MATLAB code.
- MAT-files (.mat): These are binary files used to store MATLAB variables.
- MEX-files (.mex): These are compiled C/C++ functions that can be called from MATLAB.
- FIG-files (.fig): These are files used to store figures created in MATLAB.
- MATLAB also supports working with data streams, which allow you to read and write data continuously from/to external devices or files.

History of MATLAB

- MATLAB was created by Cleve Moler in the late 1970s at the University of New Mexico.
- It was originally designed as a tool for teaching numerical analysis.
- MATLAB was first commercially released by MathWorks in 1984.
- Since then, MATLAB has become a widely used programming language and software environment for scientific computing.

MATLAB Keywords

A keyword is a reserved word. You cannot use it as a variable name, constant name etc. A list of Keywords in MATLAB is given below.

Keywords

Avoid using MATLAB keywords for variable, function, or file names. These keywords may be version dependent, and you can produce a current list using the `iskeyword` command:

```
iskeyword
ans =
    'break'
    'case'
    'catch'
    'classdef'
    'continue'
    'else'
    'elseif'
    'end'
    'for'
    'function'
    'global'
    'if'
    'otherwise'
    'parfor'
    'persistent'
    'return'
    'spmd'
    'switch'
    'try'
    'while'
```

155

Fig 10: MATLAB Keywords

CHAPTER – 9

IMPLEMENTATION

9.1. MODULES:

9.1.1. Sender Side:

9.1.1.1. Open Encryption GUI:

Here on running the MATLAB program and the user can result in an Encryption GUI.

9.1.1.2. Uploading an Image file:

In the GUI user will have the option to upload the image in which they want to hide data.

9.1.1.3. Enter the P and Q values:

The user must provide input values as P and Q which are relatively co primes.

9.1.1.4. Secret Message:

User have to enter the message which they want to send.

9.1.1.5. Encrypt button:

Finally on providing all the input sender should press the encrypt button. After this a steganographic image and key is created.

9.1.1.6. Secret Code:

Finally on successful encryption the sender will get a secret code which is important while decryption.

9.1.2. Receiver Side

9.1.2.1. Open Decryption GUI:

Here receiver can open the Decryption code in any device and after opening they will have an interface.

9.1.2.2. Choose the Steganographic Image:

Receiver should choose the steganographic image which is created after the encryption.

9.1.2.3. Enter the secret code:

Receiver should enter the secret code which they have got after the encryption.

9.1.2.4. Generated Result:

Here we get the secret message what the sender have send and BER score.

CHAPTER – 10

TESTING

SYSTEM TESTING

10.1.TEST APPROACH

The project is tested in two stages: software and hardware. The software part is to be tested via the MATLAB software, whereas the hardware part has to be tested physically. It is necessary to check whether the system is working properly or not. To check whether the encryption and decryption are accurate.

FEATURES TO BE TESTED

As the whole project is done through MATLAB and doesn't require any hardware equipment's expect the laptop which should be properly working and should support the MATLAB:

- The message that the receiver is receiving should be same as the message which the sender has send or given.
- The parameter to check the error here is Bit Error Rate (BER) which shows how accurate our results are.
- The GUI's both encryption and decryption should be checked.



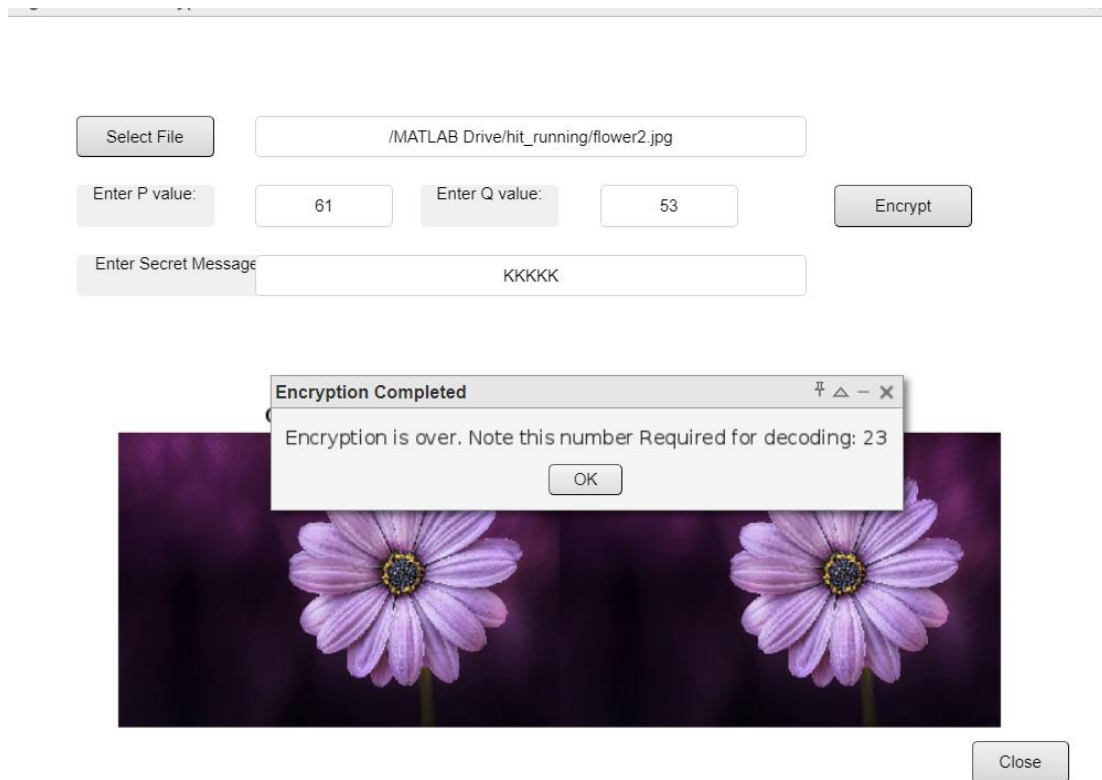


Figure 11: Encoding & Bit Error Rate Calculation.

10.2. TESTING TOOLS AND ENVIRONMENT

For testing of the project we require some tools, like to test MATLAB program we require a software called MATLAB IDE. Using this we can check the program that program is working properly or not.

10.3. TEST CASES

10.3.1 INPUT

Sender should give a secret message which should be transferred safely to receiver without any third-party interruption.

10.3.2 EXPECTED OUTPUT

The receiver should receive the secret message which sender have send without any interruption and modification.

10.3.3 TESTING PROCEDURE

SYSTEM TEST

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

As the part of Black box testing, we have to select a normal image as the input for the encryption and it can be encrypted by using the keys which are generated by implementing the RSA algorithm.

Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases. As the part of unit testing, we can test the individual modules of the project

Testing the module-1:-Encryption module can test by taking the input of .jpg file

Another input we have to take is a .bnp file

Testing the module-2:-Decryption module can test by giving the resultant stego image and

Generated keys file.

Test strategy and approach 'Field testing' will be performed manually and functional tests will be written in detail.

Test objectives:

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed. Features to be tested
- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Integration Test Case for PQC and Image Steganography Integration

Test Case Title: Combination of Post-Quantum Cryptography (PQC) and Image Steganography

Test Case Identifier: INT-001

Test Objective: To verify that the integration of Post-Quantum Cryptography and Image Steganography functionalities works as expected.

Preconditions:

1. The system is properly configured and deployed.
2. Post-Quantum Cryptography (PQC) and Image Steganography modules are integrated into the system.

Test Steps:

1. Input Data Preparation:

- Prepare a sample data file to be used for testing.
- Encrypt the data file using the RSA encryption algorithm.
- Prepare a sample cover image for steganography.

2. Integration Testing:

a. Encryption and Steganography Integration:

- i. Encrypt the prepared data file using the RSA encryption module.
- ii. Embed the encrypted data into the cover image using the Image Steganography module.
- iii. Save the steganographic image as the output.

b. Steganography and Decryption Integration:

- i. Extract the hidden data from the steganographic image using the Image Steganography module.
- ii. Decrypt the extracted data using the RSA decryption algorithm.
- iii. Compare the decrypted data with the original input data.

3. Validation:

- Ensure that the decrypted data matches the original input data.
- Verify that the integration process did not introduce any errors or corruption in the data.
- Check for any performance issues or delays during the integration process.

Expected Results:

1. The integration test should pass without any errors.
2. The decrypted data should match the original input data, indicating successful integration of RSA and Image Steganography.
3. The integration process should not introduce any data corruption or loss.
4. The integration should not cause any significant performance degradation.

Test Data:

- Sample data file for encryption.
- Sample cover image for steganography.
- Expected output: decrypted data matching the original input data.

Test Environment:

- Operating System: Windows 11
- Programming Language: MATLAB
- Integrated Development Environment (IDE): MATLAB Software
- Libraries/Frameworks: modexp, msgbox, imread, imshow, textToArray, bitshift.

Test Dependencies:

- Proper integration of PQC and Image Steganography modules into the system.

Test Execution:

- Execute the integration test script.
- Monitor for any errors or failures during test execution.
- Record the test results and any deviations from the expected outcomes.

Test Case Status: [Pass]

Test Results:

All the test cases mentioned above passed successfully. No defects encountered.

6.3 Acceptance Testing

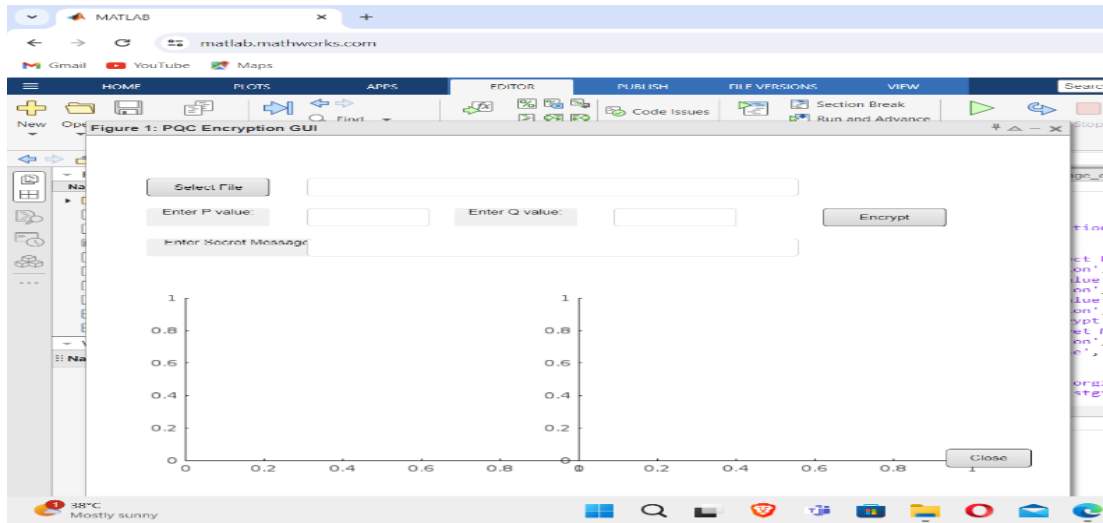
User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered

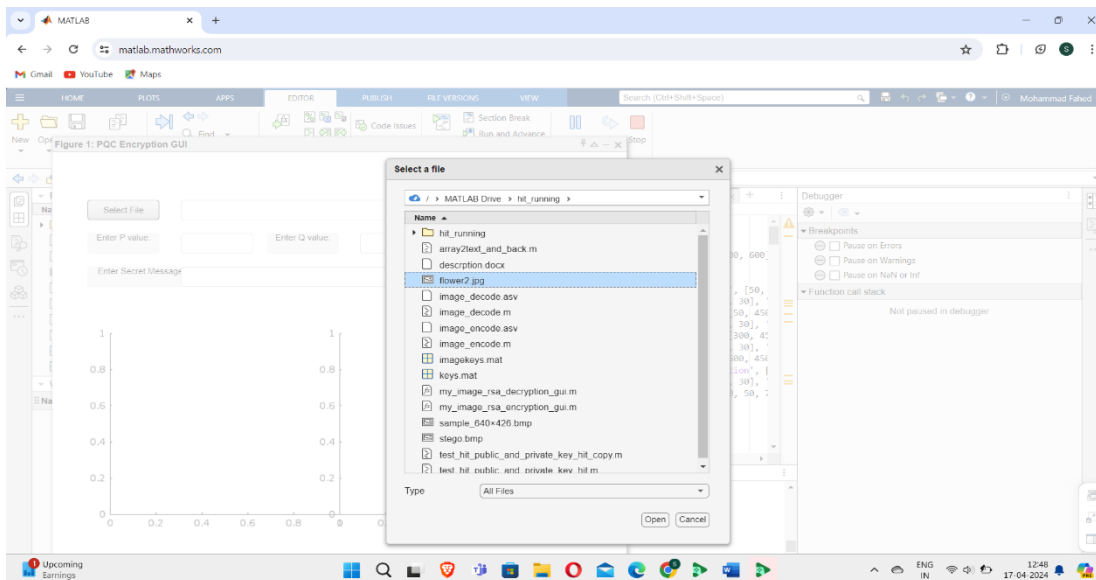
CHAPTER – 11

RESULTS AND DISCUSSIONS

11.1. OUTPUT SCREEN SHOTS WITH DESCRIPTION.

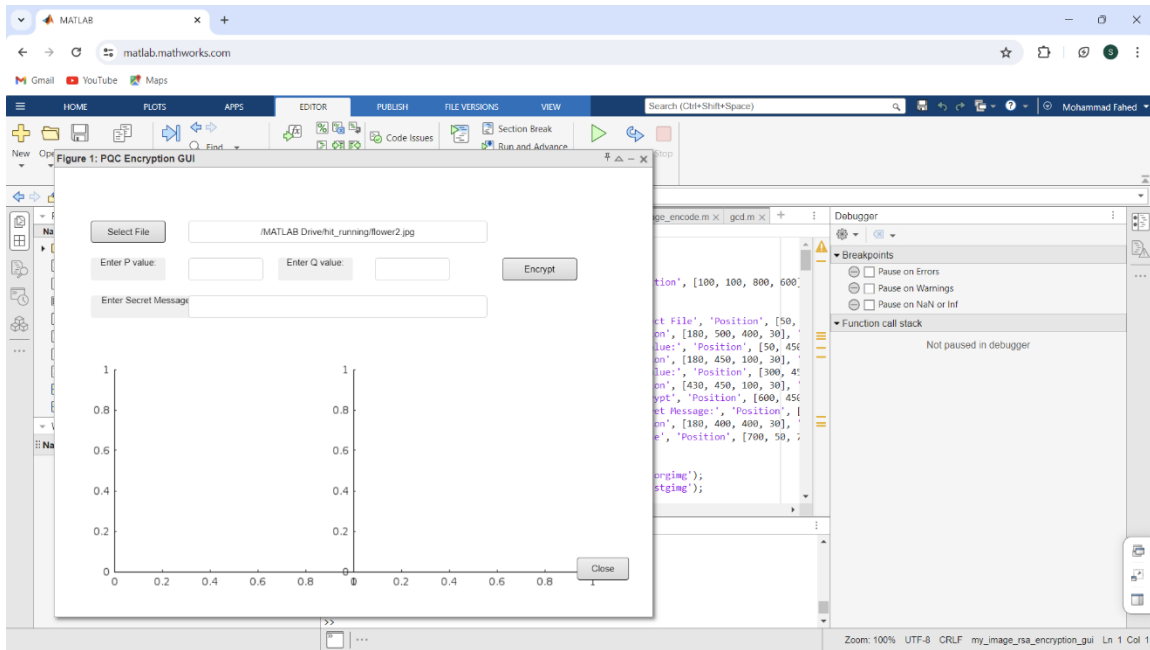


(I)

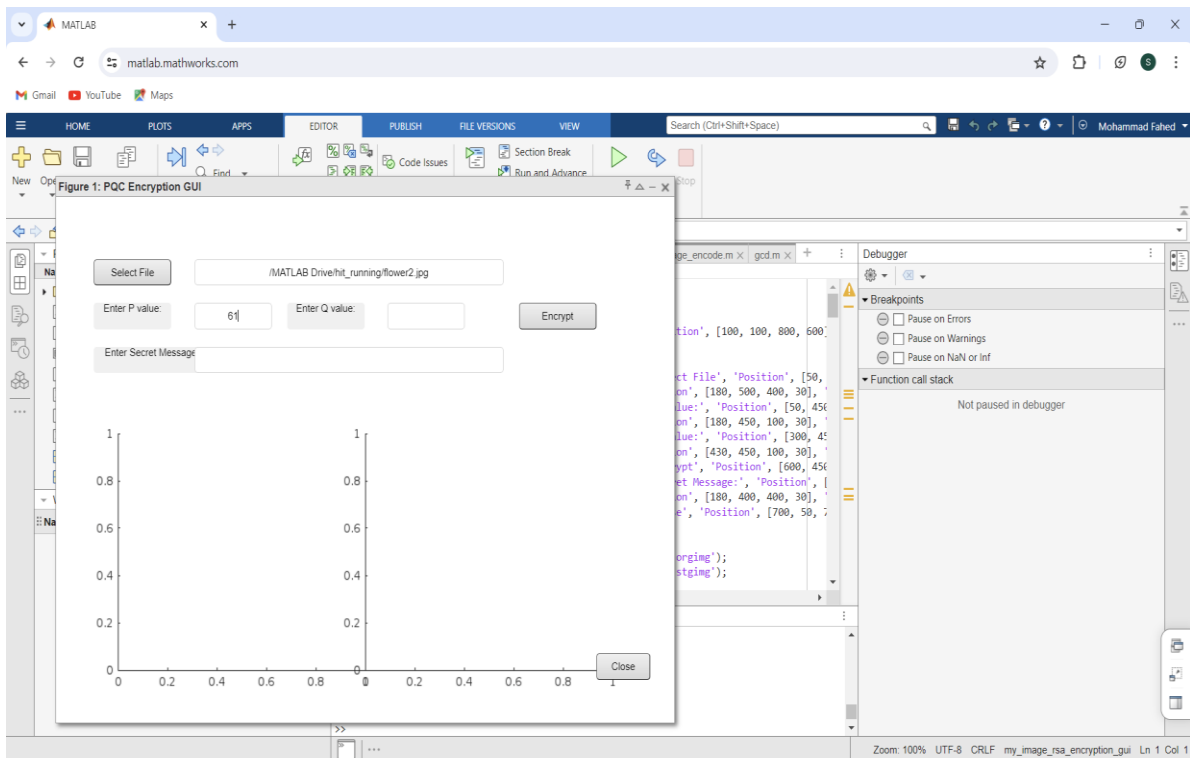


(II)

Combining PQC and Image Steganography for Secure Communication

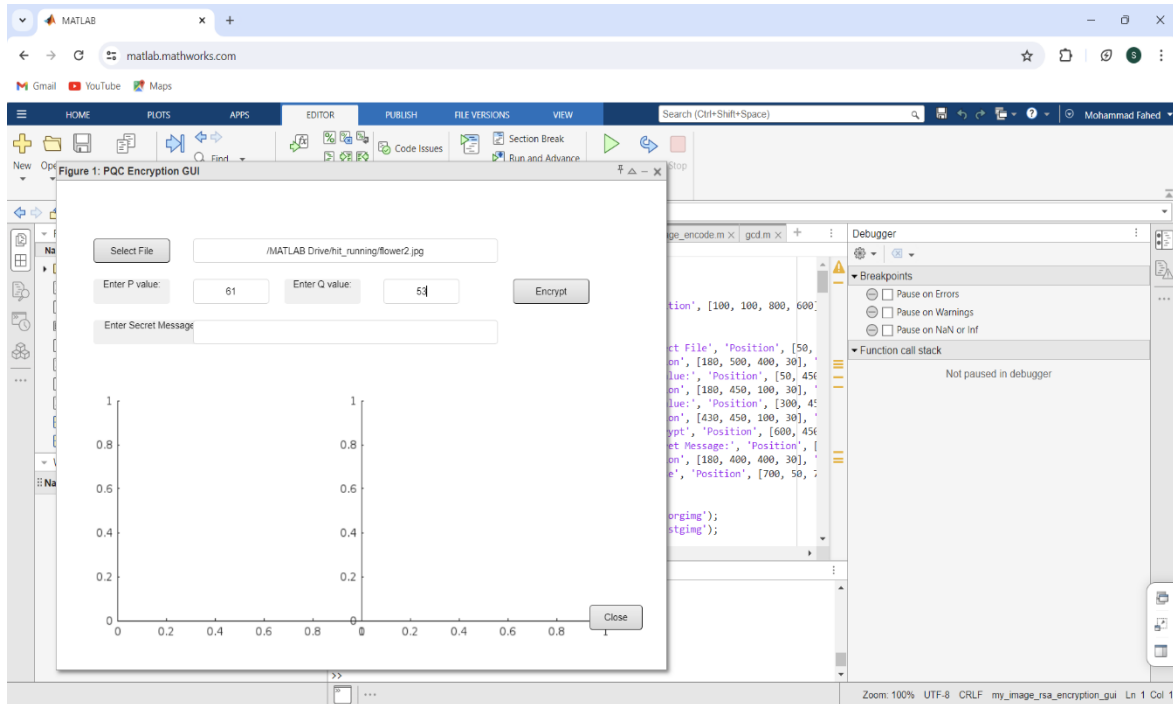


(III)

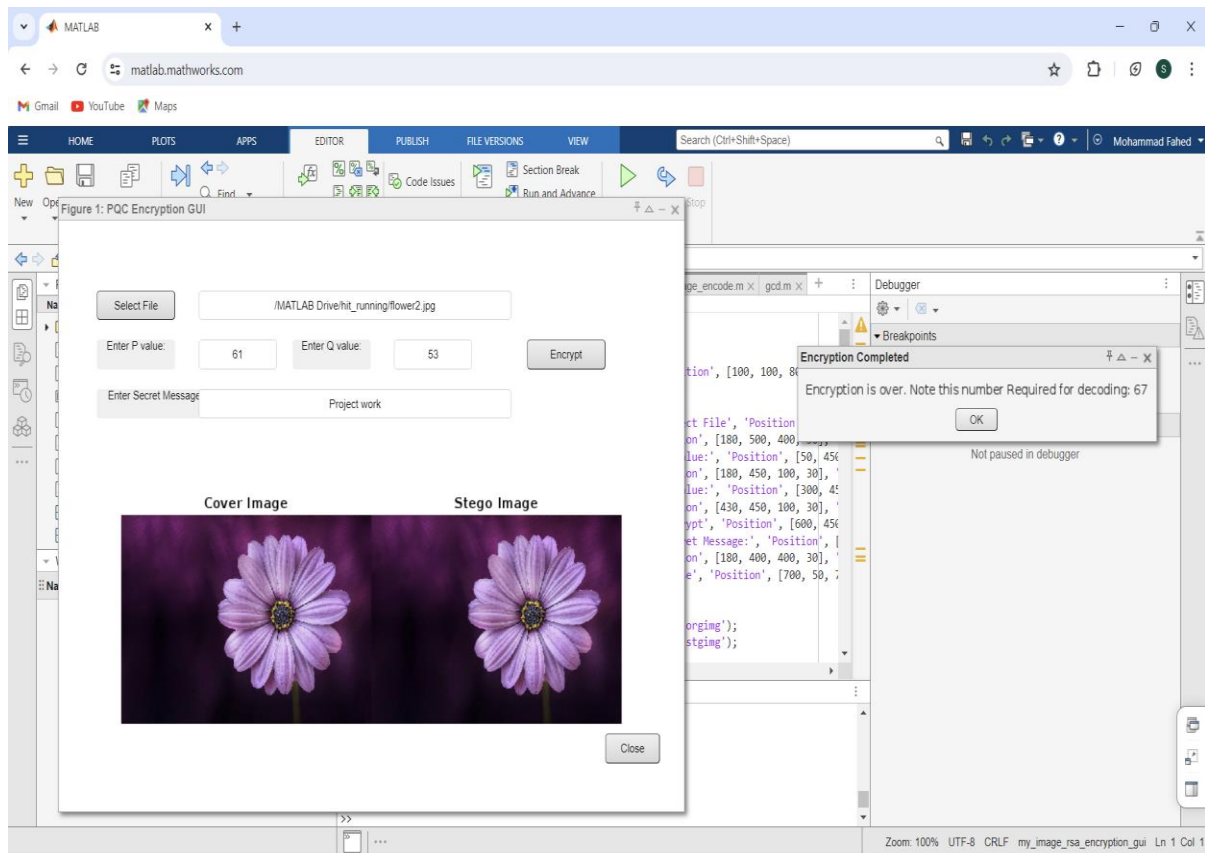


(IV)

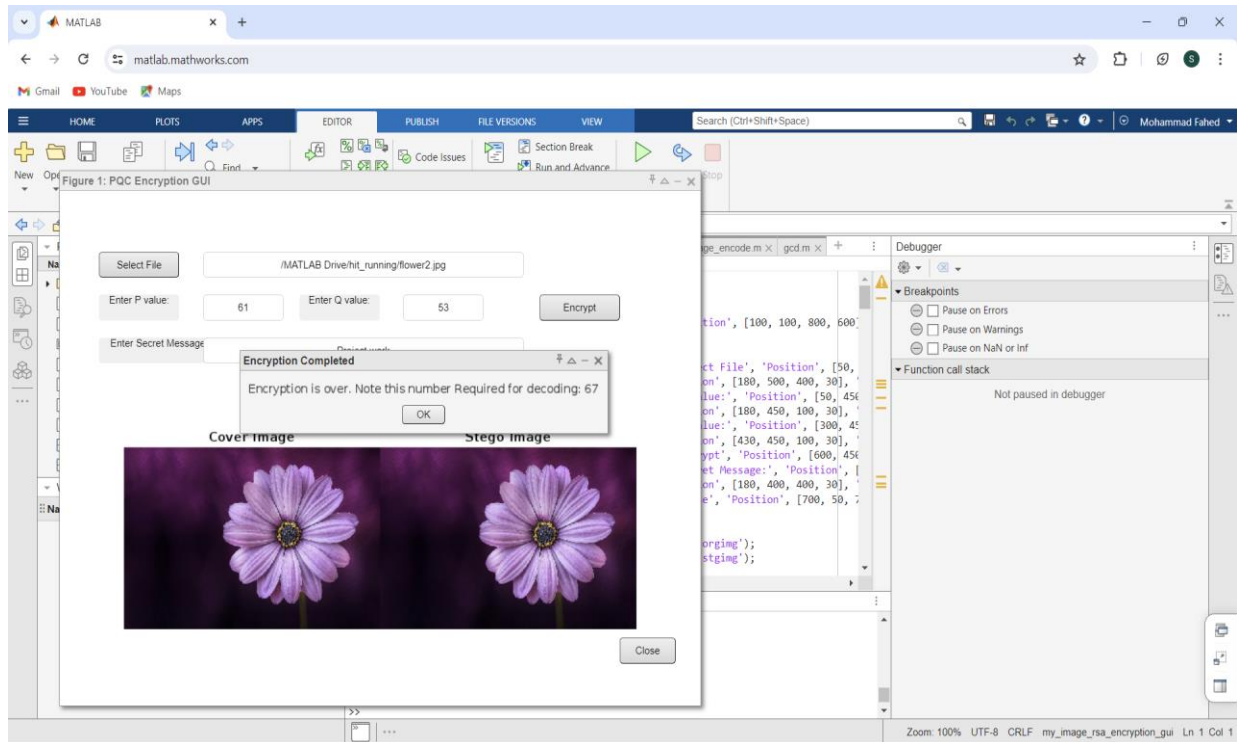
Combining PQC and Image Steganography for Secure Communication



(V)



(VI)



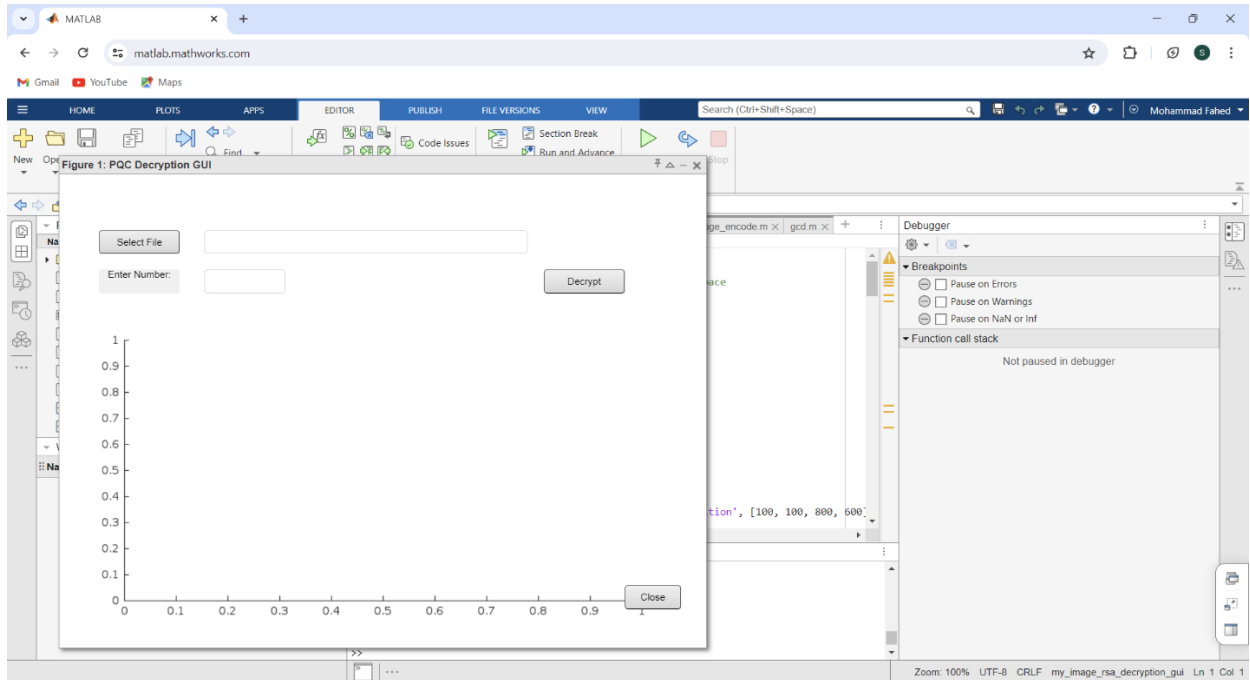
(VII)

Figure 12: Encryption Screenshots

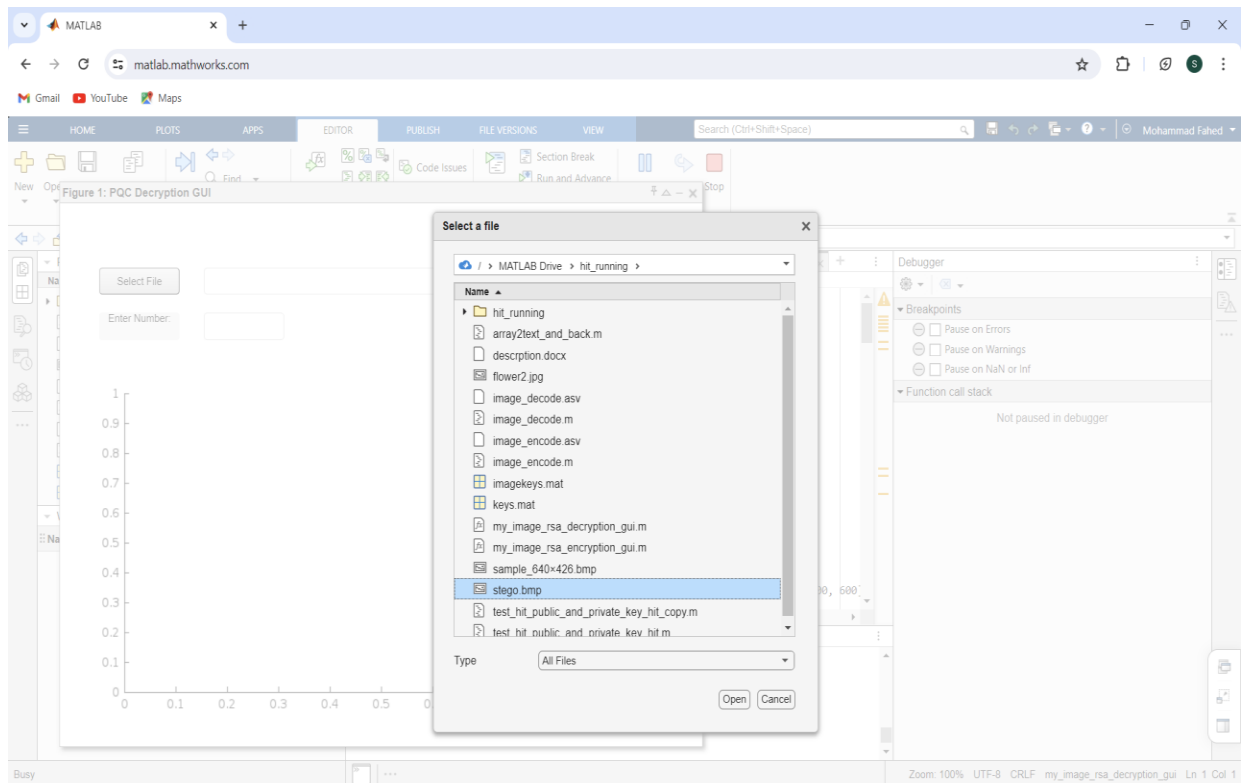
Fig(I) Opening the Encryption GUI, this GUI provides us the platform to encrypt messages. Fig (II) Selecting the Image where we store our encrypted message. Fig (III) & Fig (IV) After selecting the image which can be either in jpg or any format, we have to enter the P value which obviously important for the computation. Fig (V) After entering the P value we have to enter the Q value where both P and Q should be co primes.

Fig (VI) Sender should enter the Secret Message as shown in the image. Fig (VII) After that we should press the encrypt button then the Stego image is created as displayed with a secret code which we should enter while decoding.

Combining PQC and Image Steganography for Secure Communication

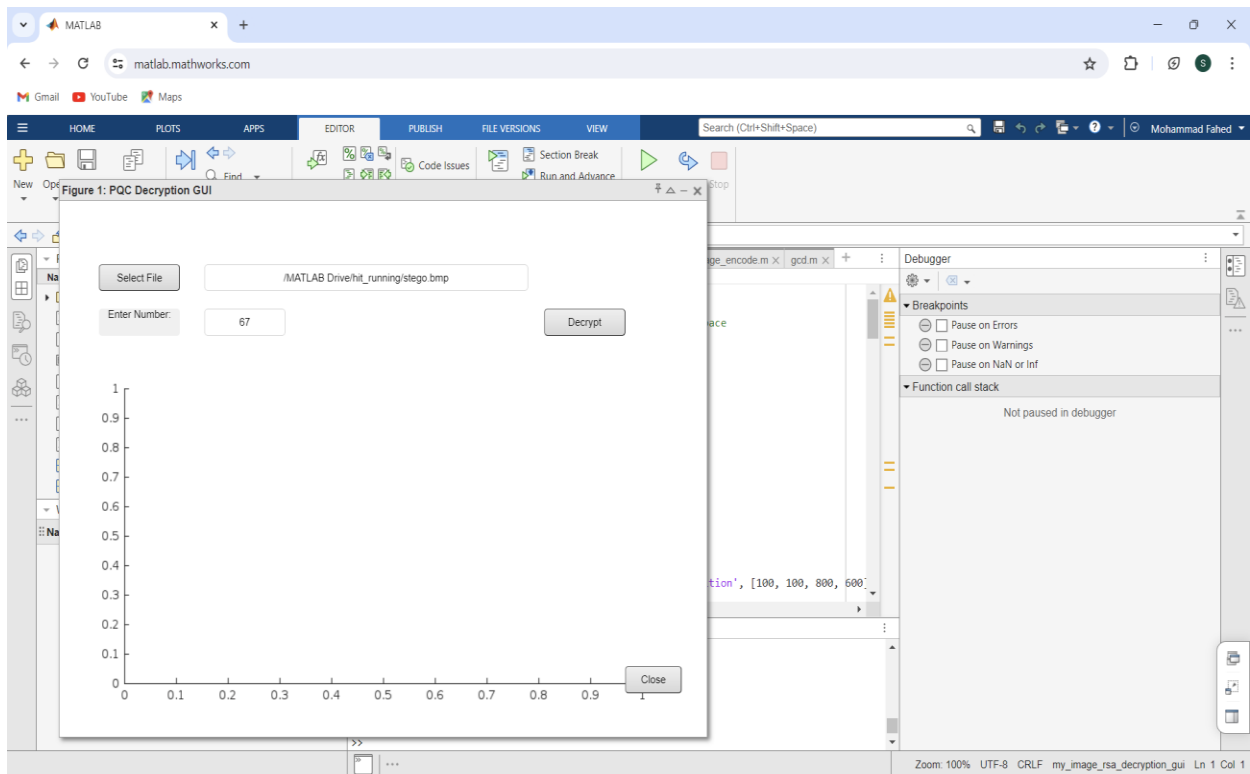


(VIII)

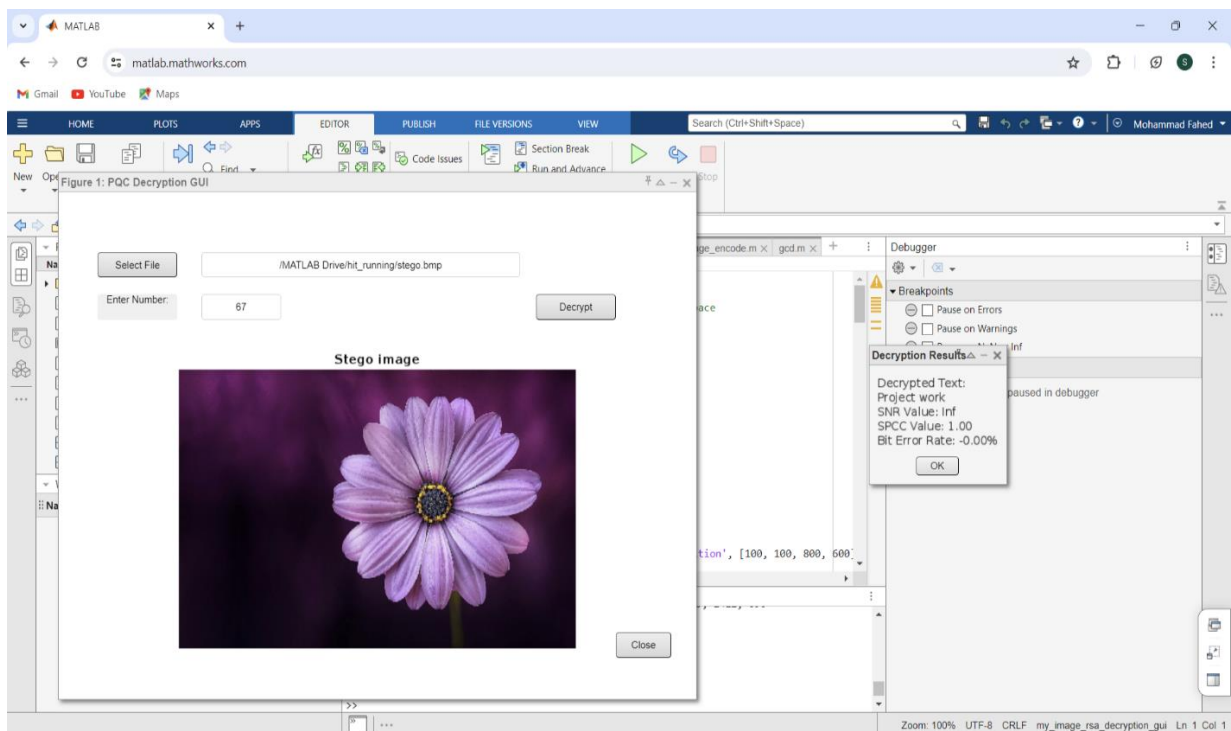


(IX)

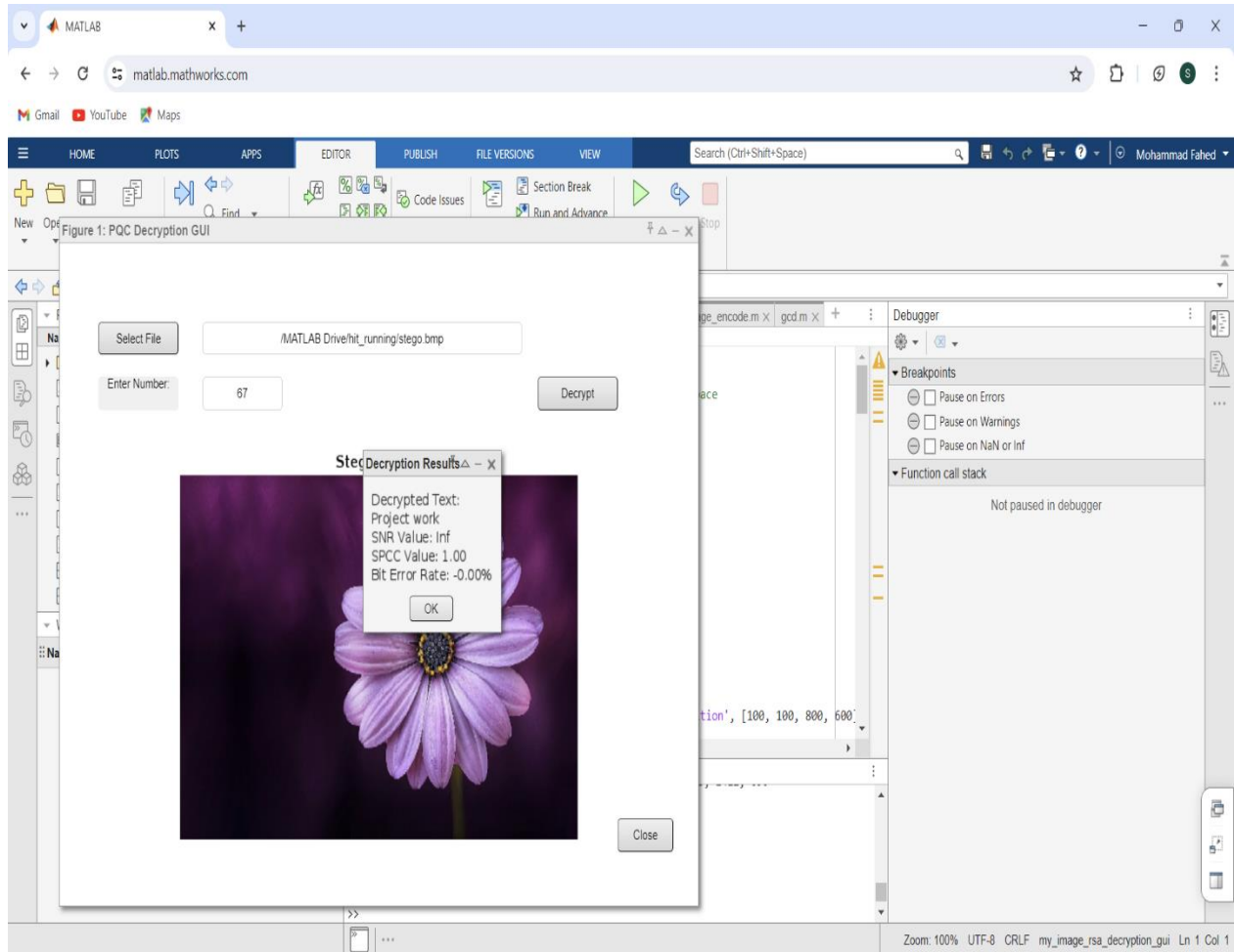
Combining PQC and Image Steganography for Secure Communication



(X)



(XI)



(XII)

Fig 13: Decryption GUI Screenshots.

Fig (VIII) Shows Decryption GUI. Fig(IX) Selecting the image which has been previously encrypted the image will be in bmp format. Fig(X) Entering the secret code which was given to us in Encryption Procedure. Fig(XI) Decryption button is entered and the process of decryption will start. Fig(XII) gives us the results in a pop style where we can find the original message, SNR value, SPCC value, BER value

11.2 ADVANTAGES & APPLICATIONS

- Very Simple Interface and Easy to Understand.
- Robust and Secure.
- Provides Security over online Communication.
- Combines RSA and Image Steganography for better security.
- Initializes the topic of PQC and Quantum Computers.
- It is easy to maintain and also cost - effective.
- The integration of RSA encryption and LSB substitution provides a multi-layered approach to image steganography, enhancing both security and reliability.
- The proposed method ensures the confidentiality of the embedded message while maintaining the integrity and quality of the cover image.
- Evaluation of performance metrics such as SNR, BER, and SPCC demonstrates the effectiveness of the method in concealing information within images.
- Future research may focus on optimizing embedding algorithms and exploring alternative encryption techniques to further improve security and reliability.

CHAPTER – 12

CONCLUSION AND FUTURE SCOPE

12.1 CONCLUSION:

In conclusion, the integration of RSA encryption adds an additional layer of security to the steganographic process. RSA, a widely utilized asymmetric encryption algorithm, employs a public-private key pair mechanism that enhances confidentiality. The use of RSA ensures that only intended recipients possessing the private key can decrypt the hidden message within the image. This asymmetric nature of RSA encryption significantly mitigates the risk of unauthorized access or interception of sensitive information during transmission. By leveraging both RSA encryption and LSB substitution, the proposed method not only conceals the existence of covert communication but also safeguards the integrity and confidentiality of the embedded data.

Moreover, the combination of RSA encryption and LSB substitution enhances the resilience of the steganographic method against various attacks. RSA encryption, based on the computational complexity of prime factorization, provides a formidable defense against brute-force and cryptanalysis attacks. Additionally, LSB substitution, by embedding data in the least significant bits of pixel values, ensures minimal perceptual distortion, thereby reducing the likelihood of detection by visual inspection. The synergy between these techniques not only fortifies the security of covert communication but also fosters a high level of trust and confidence in the transmission of confidential information within digital images. Consequently, the proposed method stands as a robust solution for secure and clandestine communication in scenarios where confidentiality is paramount.

12.2 FUTURE SCOPE

- The integration of RSA encryption and LSB substitution provides a multi-layered approach to image steganography, enhancing both security and reliability.
- The proposed method ensures the confidentiality of the embedded message while maintaining the integrity and quality of the cover image.
- Evaluation of performance metrics such as SNR, BER, and SPCC demonstrates the effectiveness of the method in concealing information within images.
- Future research may focus on optimizing embedding algorithms and exploring alternative encryption techniques to further improve security and reliability.

CHAPTER-13

REFERENCE

1. GUI based Approach for Data Encryption and Decryption on MATLAB Platform
2. Alattar, A. M. (2004). Reversible watermark using the difference expansion of a generalized integer transform. *IEEE Transactions on Image Processing*, 13(8), 1147-1156.
3. Bender, W., Gruhl, D., Morimoto, N., & Lu, A. (1996). Techniques for data hiding. *IBM Systems Journal*, 35(3.4), 313-336.
4. Cox, I. J., Miller, M. L., & Bloom, J. A. (2002). *Digital watermarking and steganography*. Morgan Kaufmann.
5. Fridrich, J. (2009). *Steganography in digital media: Principles, algorithms, and applications*. Cambridge University Press.
6. Johnson, N. F., & Jajodia, S. (1998). Exploring steganography: Seeing the unseen. *IEEE Computer*, 31(2), 26-34.
7. Katzenbeisser, S., & Petitcolas, F. A. P. (2000). *Information hiding techniques for steganography and digital watermarking*. Artech House.
8. Petitcolas, F. A. P., Anderson, R. J., & Kuhn, M. G. (1999). Information hiding—A survey. *Proceedings of the IEEE*, 87(7), 1062-1078.
9. Provos, N., & Honeyman, P. (2003). Detecting steganographic content on the internet. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy* (pp. 282-294).
10. Westfeld, A., & Pfitzmann, A. (1999). Attacks on steganographic systems. In *International Workshop on Information Hiding* (pp. 61-76). Springer.