

A
Major Project
On
**SECURING DATA IN IOT USING CRYPTOGRAPHY AND
STEGANOGRAPHY TECHNIQUES**

(Submitted in partial fulfilment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY

In
COMPUTER SCIENCE AND ENGINEERING

By

N. SAI VARDHAN REDDY (167R1A05A0)

ERRABELLI SWETHA (177R1A05M3)

NESE YASHWANTH SAI VENKATESHULU (177R1A05M3)

Under the guidance of
K. PRAVEEN KUMAR
(ASSISTANT PROFESSOR)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CMR TECHNICAL CAMPUS
UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently affiliated to JNTUH, Approved by AICTE, New Delhi)

Recognized Under Section 2(f) & 12(B) of the UGC Act. 1956,

Kandlakoya (V), Medchal Road, Hyderabad-501401.

2018-2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

This is to certify that the project entitled “**SECURING DATA IN IOT USING CRYPTOGRAPHY AND STEGANOGRAPHY TECHNIQUES**” being submitted by **N. SAI VARDHAN REDDY (167R1A05A0), ERRABELLI SWETHA(177R1A0513) & NESE YASHWANTH SAI VENKATESHULU (177R1A05M3)** in partial fulfilment of the requirements for the award of the degree of B. Tech in Computer Science and Engineering of the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by him/her under our guidance and supervision during the year 2021-2022. It is certified that they have completed the project satisfactorily.

The results embedded in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

INTERNAL GUIDE
Mr. K. Praveen Kumar
Assistant Professor

DIRECTOR
Dr. A. Raji Reddy

Head of the department
Dr. K. Srujan Raju

EXTERNAL EXAMINER

Submitted on viva voce Examination on _____

ACKNOWLEDGEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to my guide **K. Praveen Kumar**, Assistant Professor for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to the Project Review Committee (PRC): **Mr. A. Uday Kiran, Mr. J. Narasimha Rao, Dr. T. S. Mastan Rao, Mrs. G. Latha, Mr. A. Kiran Kumar**, for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Srujan Raju**, Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy**, Director for being cooperative throughout the course of this project. We also express our sincere gratitude to Sri. **Ch. Gopal Reddy**, Chairman for providing excellent infrastructure and a nice environment throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in completion of this project.

N. SAI VARDHAN REDDY (167R1A05A0)
ERRABELLI SWETHA (177R1A0513)
NESE YASHWANTH SAI VENKATESHULU (177R1A05M3)

ABSTRACT

Internet of Things (IOT) is a domain where the transfer of data is taking place every single second. The security of these data is a challenging task; however, security challenges can be mitigated with cryptography and steganography techniques. These techniques are crucial when dealing with user authentication and data privacy. In the proposed work, the elliptic Galois cryptography protocol is introduced and discussed. In this protocol, a cryptography technique is used to encrypt confidential data that came from different medical sources. Next, a Matrix XOR encoding steganography technique is used to embed the encrypted data into a low complexity image. The proposed work also uses an optimization algorithm called Adaptive Firefly to optimize the selection of cover blocks within the image. Based on the results, various parameters are evaluated and compared with the existing techniques. Finally, the data that is hidden in the image is recovered and is then decrypted.

LIST OF FIGURES

FIGURE NO	NAME OF FIGURE	PAGE NO
Fig no. 3.1.	System Architecture	8
Fig no. 3.2.	Data Flow Diagram	9
Fig no. 3.3.	Use Case Diagram	10
Fig no. 3.4.	Class Diagram	11
Fig no. 3.5.	Sequence Diagram	12

LIST OF SCREENSHOTS

SCREENSHOT NO.	SCREENSHOT NAME	PAGE NO.
5.1. Screenshot	Login Page	80
5.2. Screenshot	Encryption	80
5.3. Screenshot	IoT Reciever	81
5.4. Screenshot	Reciever	82
5.5. Screenshot	Decryption	82

TABLE OF CONTENTS

	Page No.
ABSTRACT	i
LIST OF FIGURES	ii
LIST OF SCREENSHOTS	iii
1. INTRODUCTION	1
1.1. INTERNET OF THINGS (IOT)	1
1.2. SECURITY OF IOT NETWORK	2
1.3. STEGANOGRAPHY	2
2. SYSTEM ANALYSIS	3
2.1. EXISTING SYSTEM	4
2.1.1. DISADVANTAGES OF EXISTING SYSTEM	4
2.2. PROPOSED SYSTEM	5
2.2.1. ADVANTAGES OF PROPOSED SYSTEM	5
2.3. HARWARE AND SOFTWARE REQUIREMENTS	5
2.3.1. HARDWARE REQUIREMENTS	5
2.3.2. SOFTWARE REQUIREMENTS	6
3. ARCHITECTURE	7
3.1. SYSTEM ARCHITECTURE	8
3.2. DATA FLOW DIAGRAM	9
3.3. USE CASE DIAGRAM	10
3.4. CLASS DIAGRAM	11
3.5. SEQUENCE DIAGRAM	12
4. IMPLEMENTATION	13
4.1. ELLIPTIC GALLOIS CRYPTOGRAPHY AND STEGANOGRAPHY PROTOCOL	14
4.1.1. ELLIPTIC GALLOIS CRYPTOGRAPHY	14
4.1.2. MATRIX XOR	14
4.1.3. OM-XOR STEGANOGRAPHY TECHNIQUE	15

4.2. SAMPLE CODE	15
4.2.1. ENCRYPTION OF TEXT INTO IMAGE	15
4.2.2. DECRYPTION OF TEXT FROM IMAGE	34
4.2.3. IOT ROUTER	46
4.2.4. LOGIN	58
4.2.5. RECIEVER DECRYPTION	63
4.3. PARAMETER EVALUATION	75
4.3.1. MEAN SQAURE ERROR	75
4.3.2. PEAK SIGNAL TO NOISE RATIO	76
4.3.3. CARRIER CAPACITY	76
4.3.4. TIME COMPLEXITY	77
5. SCREENSHOTS	79
5.1. LOGIN PAGE	80
5.2. ENCRYPTION	80
5.3. IOT ROUTER	81
5.4. RECIEVER	82
5.5. DECRYPTION	82
6. TESTING	83
6.1. INTRODUCTION TO TESTING	84
6.2. TYPES OF TESTING	84
6.2.1. UNIT TESTING	84
6.2.2. INTEGRATION TESTING	84
6.2.3. FUNCTIONAL TESTING	85
6.2.4. SYSTEM TESTING	85
6.2.5. WHITE BOX TESTING	85
6.2.6. BLACK BOX TESTING	85
6.3. TEST CASES	86
6.3.1. USER LOGIN	86
6.3.2. ADMIN LOGIN	87

6.3.3. VIEW PROFILE	87
7. CONCLUSION	89
8. ACKNOWLEDGEMENT	91
8. BIBLIOGRAPHY	93

1. INTRODUCTION

1. INTRODUCTION

1.1. INTERNET OF THINGS (IOT)

Internet of Things (IOT) is a network of connected vehicles, physical devices, software, and electronic items that facilitate data exchange. The purpose of IOT is to provide the IT- infrastructure for the secure and reliable exchange of “Things”. The foundation of IOT mainly consists of the integration of sensors/actuators, radio frequency identification (RFID) tags, and communication technologies. The IOT explains how a variety of physical items and devices can be integrated with the Internet to permit those objects to cooperate and communicate with each other to reach common goals. The IOT consists mostly of little materials that are associated together to facilitate collaborative calculating situations. Constraints of the IOT include energy budget, connectivity, and computational power.

1.2. SECURITY OF IOT NETWORK

Although IOT devices have made life easier, little attention has been given to the security of these devices. Currently, the focus of developers is to increase the capabilities of these devices, with little emphasis on the security of the devices. The data that is transferred over the IOT network is vulnerable to attack. This data is needed to be secured to protect the privacy of the user. If there is no data security, then there is a possibility of data breach and thus, personal information can be easily hacked from the system. Some of the important concepts of IOT involve identification and authentication. These concepts are inter-related to each other as cryptographic functions that are necessary to ensure that the information is communicated to the correct device and if the source is trusted or not. With the lack of authentication, a hacker can easily communicate to any device.

Whenever two devices communicate with each other, there is a transfer of data between them. The data can also be very sensitive and personal. Therefore, when this sensitive data is moving from device to device over the IOT network, then there is a need for encryption of the data. Encryption also helps to protect data from intruders. The data can be easily encrypted with the help of cryptography, which is the process of converting simple text into unintelligible text. The primary objectives of cryptography are confidentiality, integrity, non-repudiation, and authentication. Elliptic curve cryptography (ECC) is one of the cryptographic algorithms that are used in the proposed work. ECC is a public key cryptographic technique based on the algebraic structure of elliptic curves over finite fields.

1.3. STEGANOGRAPHY

In addition, to the cryptographic techniques, another method, named steganography is used in the proposed work which helps to provide additional security to the data. Steganography hides encrypted messages in such a way that no one would even suspect that an encrypted message even exists in the first place. In modern digital steganography, encryption of data occurs using typical cryptographic techniques. Next, a special algorithm helps to insert the data into redundant data that is part of a file format, such as a JPEG image. The proposed work uses Matrix XOR steganography to provide additional security. The image block is optimized with the help of Adaptive Firefly algorithm in which the encrypted data is hidden in a selected block from a huge image block.

2. SYSTEM ANALYSIS

2. SYSTEM ANALYSIS

2.1. EXISTING SYSTEM

Daniels introduced security microvisor middleware, which uses software virtualization and assembly level code verification to provide memory isolation and custom security.

Banerjee presented energy-efficient datagram transport layer security (EEDTLS), which is a low energy variant of datagram transport layer security (DTLS) that had the same security strength but a lower energy requirement.

Monogram proposed a system in which medical sensor devices are embedded in the human body to collect clinical measurements of patients. Significant changes in respiratory rate, blood pressure, heart rate, blood sugar, and body temperature that exceed standard levels are detected by the sensors, which generate an alert message containing relevant health information that is sent to the doctor, with the help of a wireless network. This system uses a vital management security mechanism to protect large amounts of data in the industry.

Sun proposed Cloud Eyes, a cloud-based antimalware system. The proposed system provided efficient and trusted security services to the devices in the IOT network.

UKIL studied the requirements of embedded security, provided methods and solutions for resisting cyber-attacks, and provided technology for tamper proofing the embedded devices based on the concept of trusted computing.

Yang proposed the lightweight break-glass access control (LIBAC) system in which medical files can be encrypted in two ways:

- 1) Attribute-Based Access
- 2) Break-glass access.

In standard situations, a medical worker can decrypt and access data if the attribute set satisfies the access policy of a medical file. In an emergency, a break-glass access mechanism is used that can bypass the access policy of the medical file so that emergency medical care workers or rescue workers can access the data in a timely fashion.

2.1.1. DISADVANTAGES OF EXISTING SYSTEM:

- There is no effective secret key used for data hiding.
- Less secure cryptographic techniques have been used.

2.2. PROPOSED SYSTEM

The proposed system proposes the elliptic Galois cryptography (EGC) protocol for protection against data infiltration during transmission over the IOT network. In the proposed work, different devices in the IOT network transmit data through the proposed protocol as a part of the controller. The encrypted algorithm within the controller encrypts the data using the EGC protocol and then the encrypted and secured message is hidden in layers of the image, with help from the steganography technique.

The image can then be easily transferred throughout the Internet such that an intruder cannot extract the message hidden inside the image. Initially, the EGC technique encrypts confidential data. Subsequently, the encoded secret message is inserted within the image by the XOR steganography technique. Next, an optimization algorithm called the Adaptive Elliptic Galois Cryptography: ECC, commonly known as the public key encryption technique, is based on elliptic curve theory. The keys are generated by using the properties of elliptic curve equations instead of traditional methods. The proposed work uses EGC. For improving the efficiency of calculations and to reduce the complexities of rounding errors, the elliptic curve over the Galois field is used. The value of the Galois field must be greater than one.

2.2.1. ADVANTAGES OF PROPOSED SYSTEM:

- All the fireflies are unisex so that all fireflies are attracted to each other.
- Attractiveness between the fireflies is proportional to their brightness; thus, a less bright firefly will move toward a brighter one. With increased distance between fireflies, both the attractiveness and brightness decrease.
- The brightness of a firefly is determined by the landscape of the objective function. Two important issues persist in the Firefly algorithm:
 - a) formulation of the attractiveness and
 - b) the variation of light intensity.

2.3. HARDWARE AND SOFTWARE REQUIREMENTS

2.3.1. HARDWARE REQUIREMENTS:

Hardware interfaces specifies the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

- Processor :Intel Core i3 5th gen @ 3.0GHz
- Hard Disk :40GB
- RAM :4GB

2.3.2. SOFTWARE REQUIREMENTS:

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements:

- Operating System :Windows XP, 7 or 8
- Coding Language :Java – AWT, Swings, Networking
- Database :MySQL/MS Access
- IDE :Eclipse Galileo
- Development Kit :JDK 1.6

3. ARCHITECTURE

3. SYSTEM ARCHITECTURE

3.1. SYSTEM ARCHITECTURE

This system architecture describes how the application is going to function. This describes how the user’s request is taken as input and how the output is delivered to them. The detailed architecture is explained below.

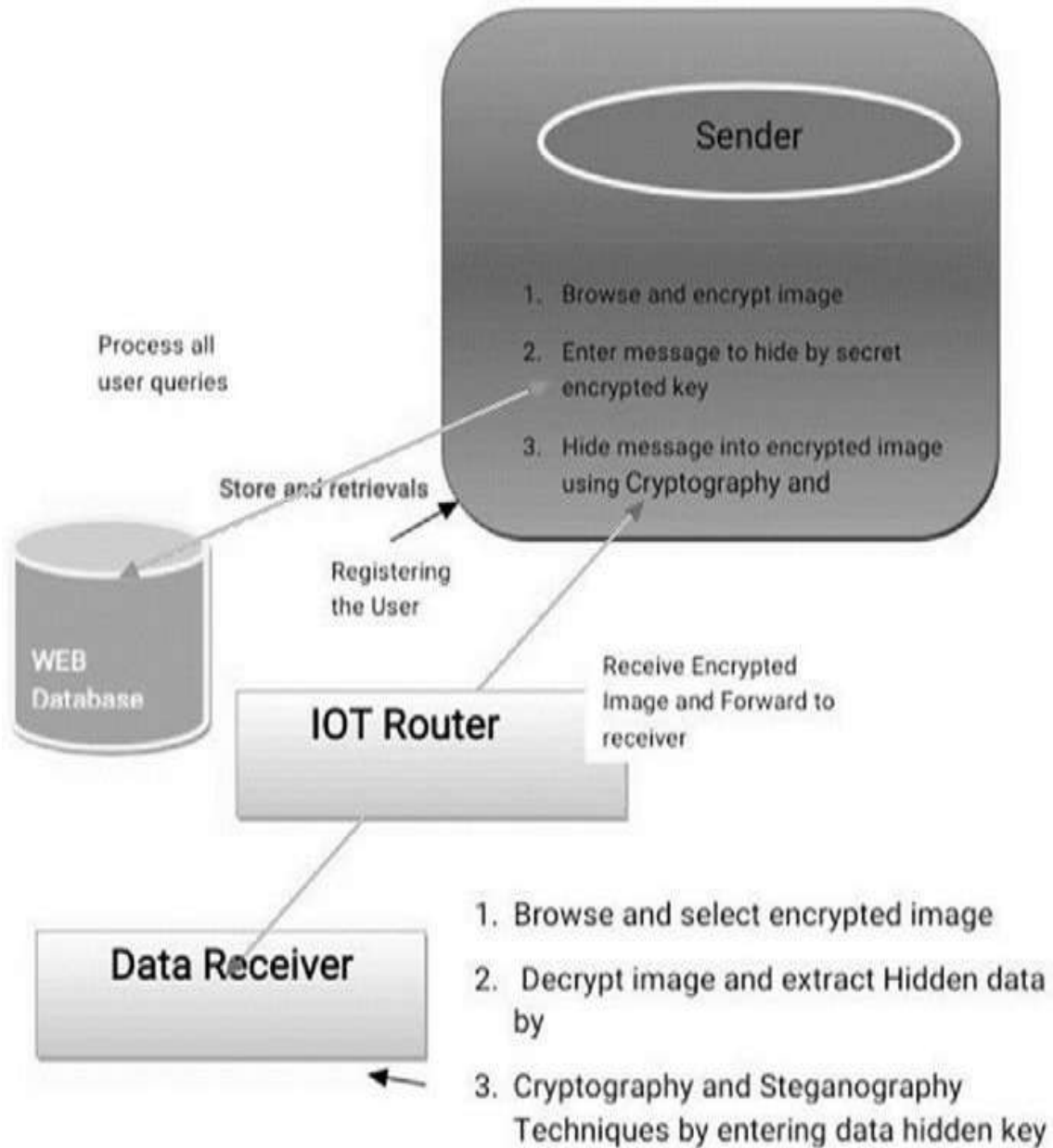


Fig 3.1. System Architecture

3.2. DATA FLOW DIAGRAM

The sender stores an encrypted message in a picture which is stored in the database. This image can be viewed separately by others, and it would seem no different than a normal picture since the encrypted data is not visible. This encrypted image can then be shared over the internet. The receiver can then extract the encrypted message by using the encryption key.

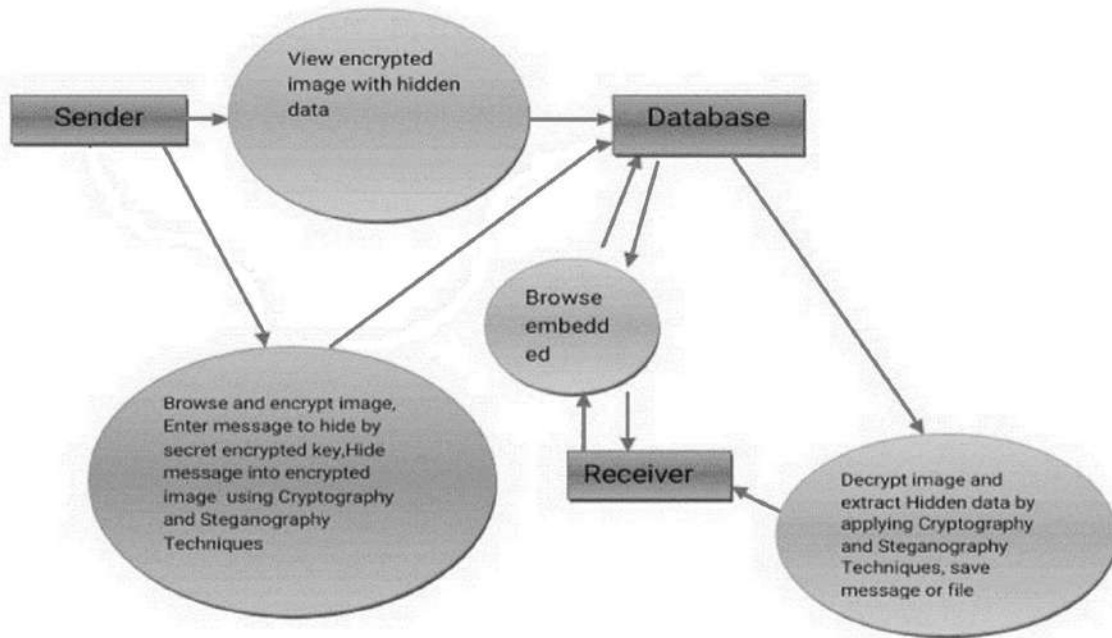


Fig 3.2. Data Flow Diagram

3.3. USE CASE DIAGRAM

The operations an admin can do are writing the encrypted message, viewing the image with the encrypted message, send it and also decrypt it. The user can only perform actions like browsing the encrypted image, decrypting the image and decrypting the message hidden in the image.

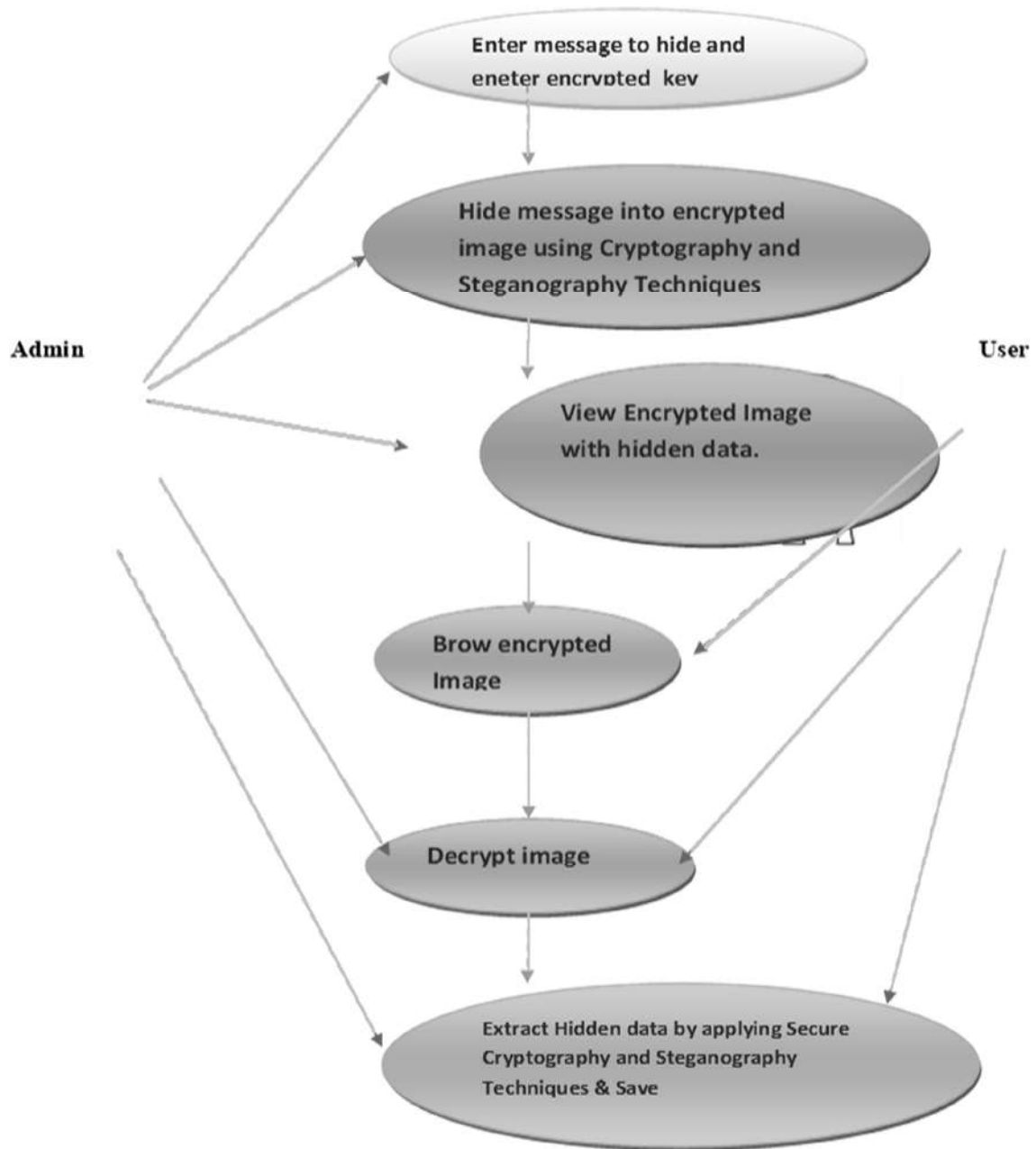


Fig 3.3. Use Case Diagram

3.4. CLASS DIAGRAM

The Sender class contains all the methods used for browsing, encrypting, and hiding data using cryptography and steganography techniques.

The IOT router class has the methods for receive and forwarding messages from the sender to the receiver.

The Login class has the methods for login, register and password reset operations.

The Data Receiver class has the methods for browsing, decrypting, extracting hidden data by applying cryptography and steganography techniques, saving message and files.

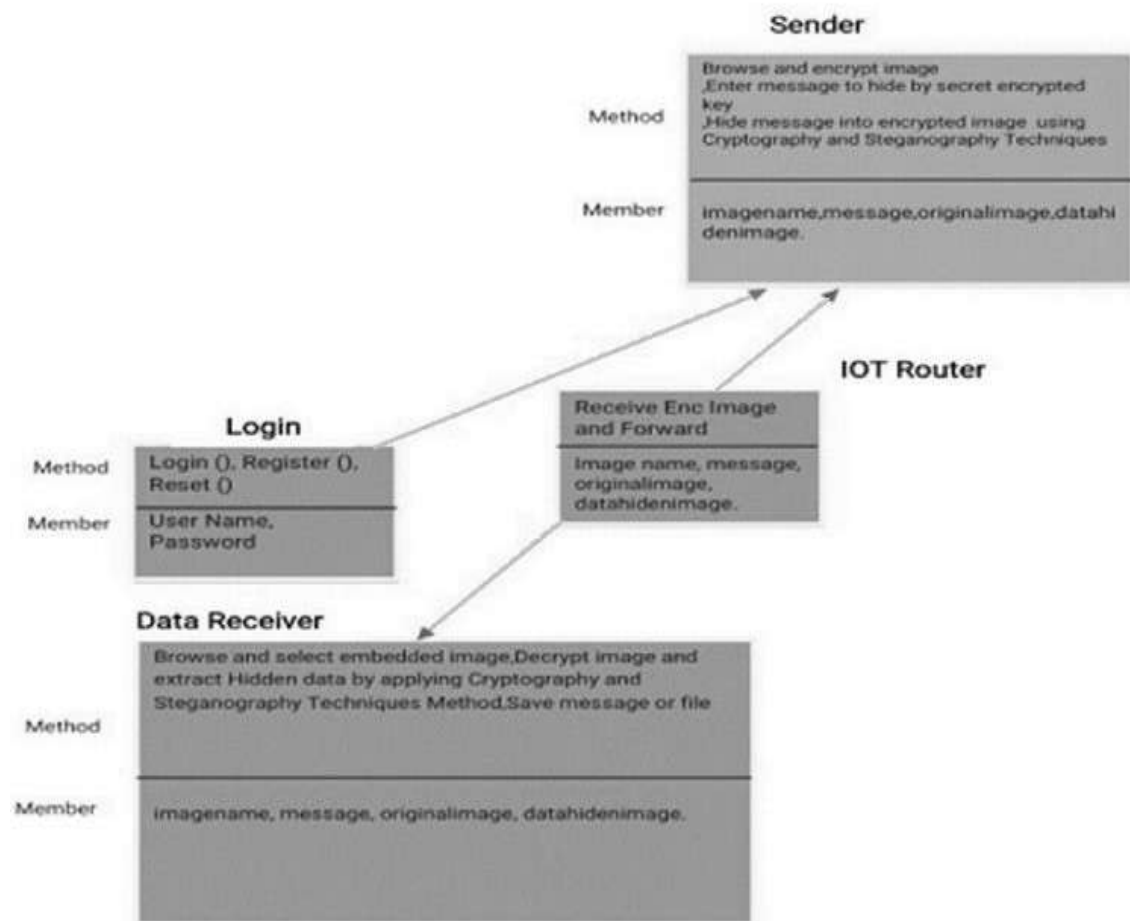


Fig 3.4. Class Diagram

3.5. SEQUENCE DIAGRAM

The sequence diagram for the project is as shown below.

Sequence Diagram:

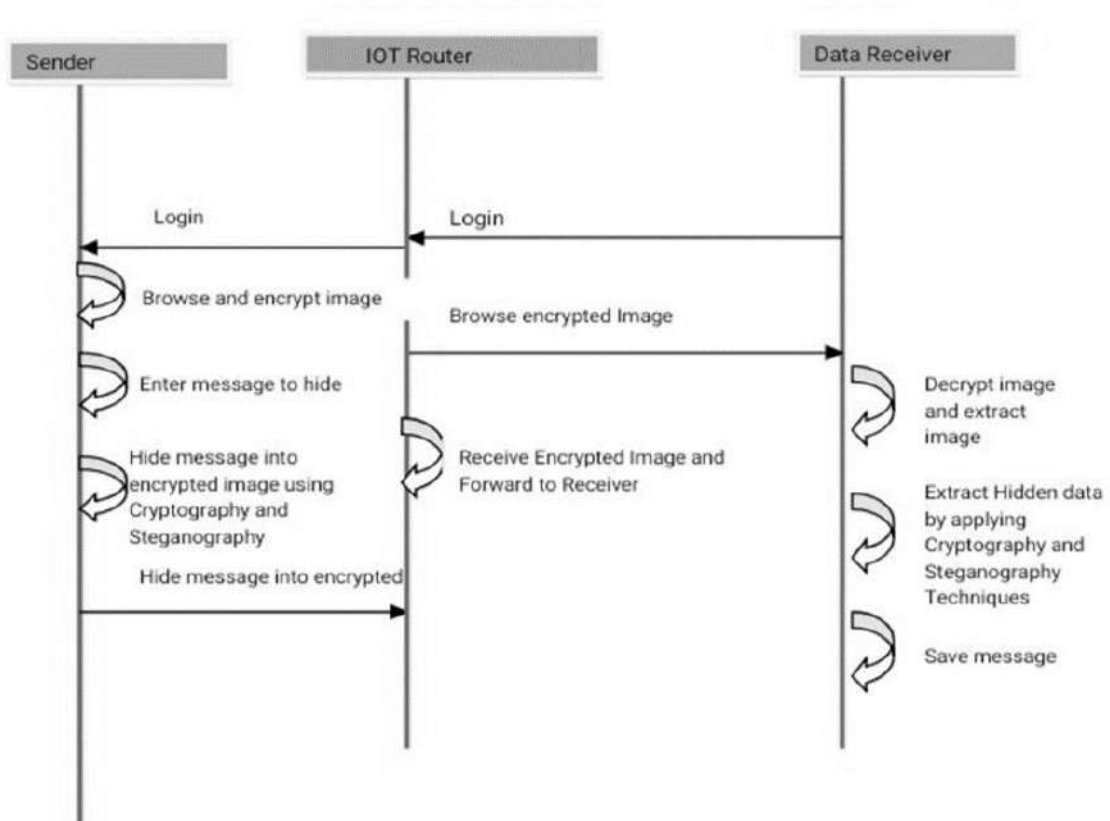


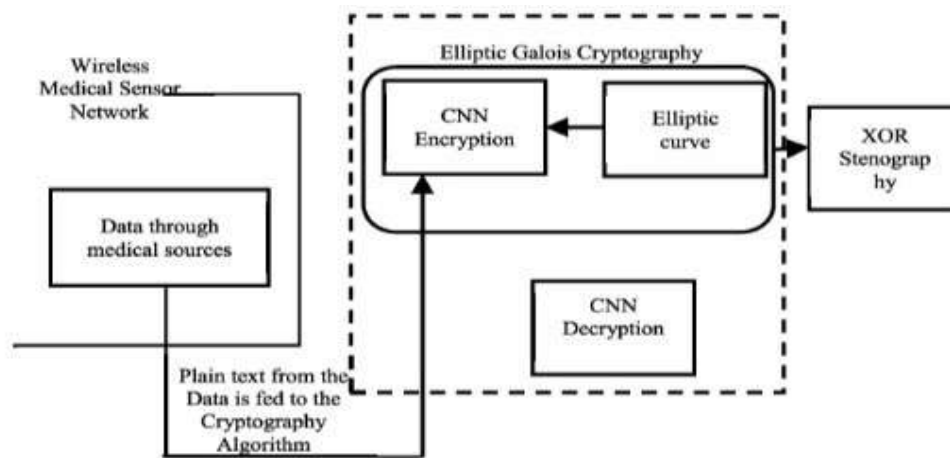
Fig 3.5. Sequence Diagram

4.IMPLEMENTATION

4. IMPLEMENTATION

4.1. ELLIPTIC GALLOIS CRYPTOGRAPHY AND STEGANOGRAPHY PROTOCOL:

The proposed system suggests the use of Elliptic Galois Cryptography (EGC) protocol for protection against data infiltration during transmission over the IoT network. In the proposed work, different devices in the IoT network transmit data through the proposed protocol as a part of the controller. The encrypted algorithm within the controller encrypts the data using the EGC protocol and then the encrypted and secured message is hidden in layers of the image, with help from the Steganography technique. The image can then be easily transferred throughout the Internet such that an intruder cannot extract the message hidden inside the image. Initially, the EGC technique encrypts confidential data. Subsequently, the encoded secret message is inserted within the image by the XOR Steganography technique.



4.1.1. Elliptic Galois Cryptography:

EGC, commonly known as the public key encryption technique, is based on elliptic curve theory. The keys are generated by using the properties of elliptic curve equations instead of traditional methods. The proposed work uses EGC. For improving the efficiency of calculations and to reduce the complexities of rounding errors, the elliptic curve over the Galois field (F_a) is used. The value of the Galois field must be greater than one.

4.1.2. Matrix XOR:

Matrix XOR is a technique for hiding encrypted data in which the encrypted data is hidden inside the H.264 video file. For this technique, the Firefly optimization technique is

used to optimize the blocks of the image. With the help of this optimization technique, block selection among the whole image is possible.

4.1.3. OM-XOR Steganography Technique:

The initial image is tiled and the secret data is hidden on the cover block with the help of Adaptive Firefly optimization. The tiled image is recombined and decoded. Finally, the encrypted message is decoded by using the secret key.

4.2. SAMPLE CODE

4.2.1. Encryption of text into image:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
import java.math.*;
import javax.imageio.ImageIO;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import javax.swing.event.*;
import javax.swing.filechooser.*;
import java.beans.*;
import java.awt.event.ActionListener;
import java.sql.*;
import java.lang.*;
import java.awt.event.ActionEvent;
import java.awt.font.*;
import java.lang.String;
import java.awt.Component;
import java.awt.geom.*;
import javax.swing.text.EditorKit;
import javax.swing.event.MouseInputAdapter;
```

```
import java.awt.image.BufferedImage;
import javax.swing.text.*;
import javax.crypto.Cipher;
import java.security.*;
import java.lang.Exception;
import java.io.DataOutputStream;
import java.io.DataInputStream;
import java.io.Serializable;
import java.io.UnsupportedEncodingException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.Random;
//import javax.crypto.BadPaddingException;
//import javax.crypto.IllegalBlockSizeException;
//import javax.crypto.NoSuchPaddingException;
import java.io.FilterInputStream;
import javax.crypto.*;
//import java.security.AlgorithmParameters;
import java.math.BigInteger;
//import java.security.interfaces.*;
//import java.security.KeyException;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStream;
import java.io.OutputStream;
import javax.crypto.spec.*;
import java.security.spec.*;
import java.applet.*;
import java.net.*;
import javax.swing.Timer;
import javax.swing.TransferHandler;
```

```

public class Encryption extends JFrame implements ActionListener,Serializable{
    JLabel label1;
    JLabel label2;
    JLabel label3;
    JButton button1;
    JButton button2;
    JTextField textfield1;
    JTextArea textarea2;
    JScrollPane sp_textarea2;
    JPasswordField passwordfield3;
    JButton button3;
    JButton button4;
    JButton button5;
    JButton button6;

    JFileChooser filechooser;
    File f,tempfilename,Ofilename,Sfilename;
    int Copened,Cencrypt,Csave;
    InetAddress ipaddress;
    String name,Ekey,address;
    String chosenFile;
    InputStream ins;
    OutputStream outs;
    Thread t;

    public Encryption()
    {
        EncryptionLayout customLayout = new EncryptionLayout();
        getContentPane().setFont(new Font("Helvetica", Font.PLAIN, 12));
        getContentPane().setLayout(customLayout);
        getContentPane().setBackground(Color.YELLOW);
    }
}

```

```
label1 = new JLabel("SOURCE");
getContentPane().add(label1);
label1.setFont(new Font("Garamond", Font.BOLD, 16));

label2 = new JLabel("MESSAGE");
getContentPane().add(label2);
label2.setFont(new Font("Garamond", Font.BOLD, 16));

label3 = new JLabel("KEY");
label3.setVisible(false);
getContentPane().add(label3);
label3.setFont(new Font("Garamond", Font.BOLD, 16));

button1 = new JButton("BACK");
getContentPane().add(button1);
button1.addActionListener(this);
button1.setFocusable(true);
button1.setRolloverEnabled(true);
button1.setContentAreaFilled(true);
        button1.setBorderPainted(true);
button1.setVerifyInputWhenFocusTarget(true);
button1.setFont(new Font("Garamond", Font.BOLD, 16));

button2 = new JButton("NEXT");
getContentPane().add(button2);
button2.setFocusable(true);
button2.setContentAreaFilled(true);
        button2.setBorderPainted(true);
button2.setRolloverEnabled(true);
button2.setVerifyInputWhenFocusTarget(true);
button2.addActionListener(this);
button2.setFont(new Font("Garamond", Font.BOLD, 16));
```

```
textfield1 = new JTextField("");
getContentPane().add(textfield1);
textfield1.setEditable(false);
textfield1.setBackground(Color.LIGHT_GRAY);
textfield1.setToolTipText("Choose the image file to be encrypted");

textarea2 = new JTextArea("");
getContentPane().add(textarea2);
textarea2.setFocusable(true);
GraphicsEnvironment.getLocalGraphicsEnvironment();
textarea2.setAutoscrolls(true);
textarea2.setWrapStyleWord(true);
textarea2.setBackground(Color.LIGHT_GRAY);
textarea2.setToolTipText("Enter the message to be embedded with image");

sp_textarea2 = new JScrollPane(textarea2);
getContentPane().add(sp_textarea2);

passwordfield3 = new JPasswordField("");
getContentPane().add(passwordfield3);
passwordfield3.setVisible(false);
passwordfield3.setToolTipText("Enter a 6 digit KEY");

button3 = new JButton("BROWSE");
getContentPane().add(button3);
button3.addActionListener(this);
button3.setFocusable(true);
button3.setContentAreaFilled(true);
        button3.setBorderPainted(true);
button3.setRolloverEnabled(true);
button3.setVerifyInputWhenFocusTarget(true);
button3.setFont(new Font("Garamond", Font.BOLD, 16));
```

```
button4 = new JButton("CLEAR");
getContentPane().add(button4);
button4.setFocusable(true);
button4.setContentAreaFilled(true);
    button4.setBorderPainted(true);
button4.setToolTipText("Clears the Textarea");
button4.setRolloverEnabled(true);
button4.setVerifyInputWhenFocusTarget(true);
button4.addActionListener(this);
button4.setFont(new Font("Garamond", Font.BOLD, 16));
```

```
button5 = new JButton("SAVE");
getContentPane().add(button5);
button5.setFocusable(true);
button5.setContentAreaFilled(true);
    button5.setBorderPainted(true);
button5.setToolTipText("Saves the Encrypted image file");
button5.setRolloverEnabled(true);
button5.setVerifyInputWhenFocusTarget(true);
button5.addActionListener(this);
button5.setFont(new Font("Garamond", Font.BOLD, 16));
```

```
button6 = new JButton("SEND");
getContentPane().add(button6);
button6.setFocusable(true);
button6.setContentAreaFilled(true);
    button6.setBorderPainted(true);
button6.setToolTipText("Sends the Encrypted image file to the remote machine");
button6.setRolloverEnabled(true);
button6.setVerifyInputWhenFocusTarget(true);
button6.addActionListener(this);
button6.setFont(new Font("Garamond", Font.BOLD, 16));
```

```

button6.setVisible(false);

Copened=0;
Cencrypt=0;
Csave=0;

filechooser=new JFileChooser();
filechooser.setSelectionMode(JFileChooser.FILES_ONLY);

setSize(getPreferredSize());

addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
        System.exit(0);
    }
});
}

public static void main(String args[]) {
    Encryption en = new Encryption();

    en.setTitle("Encryption");
    en.show();
    en.pack();
    JFrame.setDefaultLookAndFeelDecorated(true);
        JDialog.setDefaultLookAndFeelDecorated(true);
        try
        {

            UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
        }
    }
}

```

```

        //javax.swing.UIManager.setLookAndFeel("com.birosoft.liquid.LiquidLookAndFeel"
    );
        }
        catch (Exception ex)
        {
            System.out.println("Failed loading L&F: ");
            System.out.println(ex);
        }
    }

    public void Imageencrypt(String message,File file,int key) throws java.io.IOException
    {
        byte b[]=new byte[2];

        BigInteger Aki,Mki;
        int k,k1;
        DataInputStream ins=new DataInputStream(new FileInputStream(file));
        DataOutputStream outs=new DataOutputStream(new FileOutputStream(new
        File("1.jpg")));
        for(int c=0;c<key;c++)
        {
            int ch=ins.read();
            outs.write(ch);
        }
        int len=message.length();
        byte mess[]=new byte[2];
        char chmess[]=new char[len+1];
        k=k1=0;
        for(int i=0;i<=len;i++)
        {
            message.getChars(0,len, chmess,0);
            if(i==0)
            {

```



```

BigDecimal bd=new BigDecimal(len);
BigInteger Blen=bd.toBigInteger();
String Slen=Blen.toString(2);
char Clen[]=new char[Blen.bitLength()];
Slen.getChars(0,Blen.bitLength(),Clen,0);
for(int j=0;j<=7;j++)
{
if(j==0)
{
for(k=0;k<8-Blen.bitLength();k++)
{
int n=ins.read(b);
Abi=new BigInteger(b);
String Aby=Abi.toString(2);
int Alen=Abi.bitLength();
if(b[0]<0)
Alen++;
char Ach[]=new char[Alen+1];
Aby.getChars(0,Alen,Ach,0);

if(b[0]==0)
{
}
else
{
if(Ach[Alen-1]=='1')
{
if(Alen==Abi.bitLength())
{
BigInteger bi=new BigInteger("1111111111111110",2);
BigInteger big=Abi.and(bi);
b=big.toByteArray();

```

```

    }
    else
    {
        BigInteger bi=new BigInteger("-1",2);
        BigInteger big=Abi.subtract(bi);
        b=big.toByteArray();
    }
}
outs.write(b);
}
} //for loop k
j=j+k-1;
} // if of j
    else
    {
        int n=ins.read(b);
        Abi=new BigInteger(b);
        String Aby=Abi.toString(2);
        int Alen=Abi.bitLength();
        if(b[0]<0)
            Alen++;
        char Ach[]=new char[Alen+1];
        Aby.getChars(0,Alen,Ach,0);
        if(b[0]==0)
        {
            Alen=1;
        }
        if(Clen[j-k]=='0' && Ach[Alen-1]=='1')
        {
            if(Alen==Abi.bitLength())
            {
                BigInteger bi=new BigInteger("1111111111111110",2);

```

```

        BigInteger big=Abi.and(bi);
        b=big.toByteArray();
    }
    else
    {
        BigInteger bi=new BigInteger("-1",2);
        BigInteger big=Abi.subtract(bi);
        b=big.toByteArray();
    }
    }
    else if(Clen[j-k]=='1' && Ach[Alen-1]=='0')
    {
        if(Alen==Abi.bitLength())
        {
            BigInteger bi=new BigInteger("1",2);
            BigInteger big=Abi.add(bi);
            b=big.toByteArray();
        }
        else
        {
            BigInteger bi=new BigInteger("-1",2);
            BigInteger big=Abi.add(bi);
            b=big.toByteArray();
        }
    }
    outs.write(b);
} // end else

} // for loop j

} // end of if

```

```

else
{
String slen=String.valueOf(chmess[i-1]);
byte blen[]=slen.getBytes();
BigInteger Blen=new BigInteger(blen);
String Slen=Blen.toString(2);
char Clen[]=new char[Blen.bitLength()];
Slen.getChars(0,Blen.bitLength(),Clen,0);
for(int j=0;j<=7;j++)
{
if(j==0)
{
for(k1=0;k1<8-Blen.bitLength();k1++)
{
int n=ins.read(b);
Abi=new BigInteger(b);
String Aby=Abi.toString(2);
int Alen=Abi.bitLength();
if(b[0]<0)
Alen++;
char Ach[]=new char[Alen+1];
Aby.getChars(0,Alen,Ach,0);
if(b[0]==0)
{
}
else
{
if(Ach[Alen-1]=='1')
{
if(Alen==Abi.bitLength())
{
BigInteger bi=new BigInteger("1111111111111110",2);

```

```

        BigInteger big=Abi.and(bi);
        b=big.toByteArray();
    }
    else
    {
        BigInteger bi=new BigInteger("-1",2);
        BigInteger big=Abi.subtract(bi);
        b=big.toByteArray();
    }
}
}
outs.write(b);
} //for loop k
    j=j+k1-1;
} // if of j
else
{
    int n=ins.read(b);
    Abi=new BigInteger(b);
    String Aby=Abi.toString(2);
    int Alen=Abi.bitLength();
    if(b[0]<0)
        Alen++;
    char Ach[]=new char[Alen+1];
    Aby.getChars(0,Alen,Ach,0);
    if(b[0]==0)
    {
        Alen=1;
    }
    if(Clen[j-k1]=='0' && Ach[Alen-1]=='1')
    {
        if(Alen==Abi.bitLength())

```

```

        {
            BigInteger bi=new BigInteger("111111111111110",2);
            BigInteger big=Abi.and(bi);
            b=big.toByteArray();
        }
        else
        {
            BigInteger bi=new BigInteger("-1",2);
            BigInteger big=Abi.subtract(bi);
            b=big.toByteArray();
        }
    }
else if(Clen[j-k1]=='1' && Ach[Alen-1]=='0')
    {
        if(Alen==Abi.bitLength())
        {
            BigInteger bi=new BigInteger("1",2);
            BigInteger big=Abi.add(bi);
            b=big.toByteArray();
        }
        else
        {
            BigInteger bi=new BigInteger("-1",2);
            BigInteger big=Abi.add(bi);
            b=big.toByteArray();
        }
    }
    outs.write(b);
} // end else
} // for loop j
} // end of else
} // for loop i

```

```

while(true)
{
int i=ins.read();
if(i==-1) break;
outs.write(i);
}
ins.close();
outs.close();
}

public void actionPerformed(ActionEvent e)
{
try
{
String cmd;
cmd=e.getActionCommand();
if(cmd.equals("BACK"))
{
dispose();
Stegno s = new Stegno();
s.show();
s.pack();
s.setTitle("Main Menu - Implementation of Steganography");
}
if(cmd.equals("CLEAR"))
{
textarea2.setText("");
}
if(cmd.equals("SAVE"))
{
if(Copened==1 && Cencrypt==1)
{
int r=filechooser.showSaveDialog(this);

```

```

Sfilename=filechooser.getSelectedFile();
FileInputStream in=new FileInputStream("1.jpg");
FileOutputStream out=new FileOutputStream(Sfilename);

Ofilename=Sfilename;
textfield1.setEditable(true);
textfield1.setText(Sfilename.getPath());
textfield1.setEditable(false);

while(true)
{
    int i=in.read();
    if(i==-1) break;
    out.write(i);
}

in.close();
out.close();

JOptionPane.showMessageDialog(null,"\nYour image file has been encrypted and saved
successfully\n","message",JOptionPane.INFORMATION_MESSAGE);
}
else
{
    String m;
    if(Copened==0)
        m="File not Opened";
    else if(Cencrypt==0)
        m="Not Encrypted";
    else
        m="Not Decrypted";

JOptionPane.showMessageDialog(this,m,"Error",JOptionPane.ERROR_MESSAGE);

```



```

    }
}
if(cmd.equals("NEXT"))
{
    if(Copened==1)
    {
        Ekey=JOptionPane.showInputDialog("Enter 4 digit Key For Encryption");
//String type
        if(Ekey==null)
        {
            JOptionPane.showMessageDialog(this,"Enter only 4 Digit key", "Error",
JOptionPane.ERROR_MESSAGE);
        }
        if(Ekey.trim().length()<4)
        {
            JOptionPane.showMessageDialog(this,"Enter only 4 Digit key", "Error",
JOptionPane.ERROR_MESSAGE);
        }
        if(Ekey.trim().length()>4)
            JOptionPane.showMessageDialog(this,"Enter only 4 Digit key", "Error",
JOptionPane.ERROR_MESSAGE);
        else
        {
            // encrypt the message
            int key=Integer.parseInt(Ekey);
            Imageencrypt(textarea2.getText(),Ofilename,key);
            Cencrypt=1;
        }
    }
}
else
{
    JOptionPane.showMessageDialog(this,"File NotOpened", "Error",
JOptionPane.ERROR_MESSAGE);
}

```

```

    }
}

if(cmd.equals("BROWSE"))
{
    int r=filechooser.showOpenDialog(this);
    tempfilename=filechooser.getSelectedFile(); //File type

    if(r==JFileChooser.CANCEL_OPTION)

JOptionPane.showMessageDialog(this,"FileNotSelected","Error",JOptionPane.ERROR_ME
SSAGE);
    else
    {
        String name=tempfilename.getName();

        if(!name.endsWith(".jpg") && !name.endsWith(".gif") && !name.endsWith(".bmp")
&& !name.endsWith(".jpeg"))

            JOptionPane.showMessageDialog(this,"Select Only Image file", "Error",
JOptionPane.ERROR_MESSAGE);

    else
    {
        Copened=1;
        Ofilename=tempfilename;
        textfield1.setText(name);
        textfield1.setText(tempfilename.getPath());
        textfield1.setFont(new Font("Century",Font.PLAIN,15));
        textfield1.setBackground(Color.LIGHT_GRAY);
        textfield1.setEditable(false);
    }
}
}

```

```

    }
} // end try
catch(Exception xe)
{
    //xe.printStackTrace();
JOptionPane.showMessageDialog(this,xe,"Error",JOptionPane.ERROR_MESSAGE);
}
} // end of actionPerformed
} // End of class

class EncryptionLayout implements LayoutManager {
    public EncryptionLayout() {
    }
    public void addLayoutComponent(String name, Component comp) {
    }
    public void removeLayoutComponent(Component comp) {
    }
    public Dimension preferredLayoutSize(Container parent) {
        Dimension dim = new Dimension(0, 0);
        Insets insets = parent.getInsets();
        dim.width = 700 + insets.left + insets.right;
        dim.height = 400 + insets.top + insets.bottom;
        return dim;
    }
    public Dimension minimumLayoutSize(Container parent) {
        Dimension dim = new Dimension(0, 0);
        return dim;
    }
    public void layoutContainer(Container parent) {
        Insets insets = parent.getInsets();
        Component c;
        c = parent.getComponent(0);
        if (c.isVisible()) {c.setBounds(insets.left+75,insets.top+74,112,32);}
    }
}

```

```

c = parent.getComponent(1);
if (c.isVisible()) {c.setBounds(insets.left+75,insets.top+170,112,32);}
c = parent.getComponent(2);
if (c.isVisible()) {c.setBounds(insets.left+88,insets.top+160,112,32);}
c = parent.getComponent(3);
if (c.isVisible()) {c.setBounds(insets.left+145,insets.top+294,124,38);}
c = parent.getComponent(4);
if (c.isVisible()) {c.setBounds(insets.left+300,insets.top+294,124,38);}
c = parent.getComponent(5);
if (c.isVisible()) {c.setBounds(insets.left+158,insets.top+74,390,32);}
c = parent.getComponent(6);
if (c.isVisible()) {c.setBounds(insets.left+158,insets.top+120,390,150);}
c = parent.getComponent(7);
if (c.isVisible()) {c.setBounds(insets.left+200,insets.top+160,284,52);}
c = parent.getComponent(8);
if (c.isVisible()) {c.setBounds(insets.left+558,insets.top+74,114,35);}
c = parent.getComponent(9);
if (c.isVisible()) {c.setBounds(insets.left+558,insets.top+173,114,35);}
c = parent.getComponent(10);
if (c.isVisible()) {c.setBounds(insets.left+450,insets.top+294,124,38);}
c = parent.getComponent(11);
if (c.isVisible()) {c.setBounds(insets.left+559,insets.top+204,114,35);}
}
}

```

4.2.2. Decryption of text from image:

```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;
import java.awt.event.ActionListener;
import javax.imageio.ImageIO;

```

```
import java.sql.*;
import java.lang.*;
import java.awt.event.ActionEvent;
import java.awt.font.*;
import java.lang.String;
import java.awt.Component;
import java.awt.geom.*;
import javax.swing.text.EditorKit;
import javax.swing.event.MouseInputAdapter;
import java.awt.image.BufferedImage;
import javax.swing.text.Style;
import javax.swing.text.StyledDocument;
import javax.swing.text.Style;
import javax.swing.text.StyleConstants;
import javax.imageio.ImageIO;
import java.util.Vector;
import javax.swing.text.*;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.lang.String;
import java.lang.Byte;
import java.math.*;
import java.security.*;
import java.io.DataOutputStream;
import java.io.DataInputStream;
import java.io.Serializable;
import java.io.UnsupportedEncodingException;
import java.security.InvalidKeyException;
```

```
import java.security.NoSuchAlgorithmException;
import java.util.Random;
import javax.crypto.BadPaddingException;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import java.io.FilterInputStream;
import java.io.FilterOutputStream;
import javax.crypto.*;
import java.math.BigInteger;
import java.security.AlgorithmParameters;
import java.security.interfaces.*;
import java.security.KeyException;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStream;
import java.io.OutputStream;

class Decryption extends JFrame implements ActionListener,Serializable {
    JLabel label1;
    JLabel label2;
    JLabel label3;
    JButton button1;
    JButton button2;
    JTextField textfield1;
    JTextField textfield2;
    JTextArea textarea3;
    JScrollPane sp_textarea3;
    JButton button3;
    JButton button4;
    JFileChooser filechooser;
```

```

File f,tempfilename,Ofilename,Sfilename;
int Copened,Cdecrypt;
String name,Dkey;
String chosenFile;

public Decryption()
{
    decryptionLayout customLayout = new decryptionLayout();
    getContentPane().setFont(new Font("Helvetica", Font.PLAIN, 12));
    getContentPane().setLayout(customLayout);
    getContentPane().setBackground(Color.YELLOW);

    label1 = new JLabel("SOURCE");
    getContentPane().add(label1);
    label1.setFont(new Font("Garamond", Font.BOLD, 16));

    label2 = new JLabel("IMAGE");
    getContentPane().add(label2);
    label2.setVisible(false);
    label2.setFont(new Font("Garamond", Font.BOLD, 16));

    label3 = new JLabel("MESSAGE");
    getContentPane().add(label3);
    label3.setToolTipText("Encrypted message is:");
    label3.setFont(new Font("Garamond", Font.BOLD, 16));

    button1 = new JButton("BACK");
    getContentPane().add(button1);
    button1.setFocusable(true);
    button1.setRolloverEnabled(true);

```

```
button1.setVerifyInputWhenFocusTarget(true);
button1.addActionListener(this);
button1.setFont(new Font("Garamond", Font.BOLD, 18));

button2 = new JButton("NEXT");
getContentPane().add(button2);
button2.setFocusable(true);
button2.setRolloverEnabled(true);
button2.setVerifyInputWhenFocusTarget(true);
button2.addActionListener(this);
button2.setFont(new Font("Garamond", Font.BOLD, 18));

textfield1 = new JTextField("");
getContentPane().add(textfield1);
textfield1.setEditable(false);
textfield1.setBackground(Color.LIGHT_GRAY);
textfield1.setToolTipText("Choose the file to decrypt for message");

textfield2 = new JTextField("");
getContentPane().add(textfield2);
textfield2.setVisible(false);
textfield2.setToolTipText("Choose the image file to be encrypted");

textarea3 = new JTextArea("");
getContentPane().add(textarea3);
textarea3.setBackground(Color.LIGHT_GRAY);
textarea3.setToolTipText("The Encrypted Message is ::");
textarea3.setEditable(false);
sp_textarea3 = new JScrollPane(textarea3);
sp_textarea3.setWheelScrollingEnabled(true);
```



```

textarea3.setFocusable(true);

getContentPane().add(sp_textarea3);

button3 = new JButton("BROWSE");
getContentPane().add(button3);
button3.setFocusable(true);
button3.setRolloverEnabled(true);
button3.setVerifyInputWhenFocusTarget(true);
button3.addActionListener(this);
button3.setToolTipText("Select the Encrypted image file");
button3.setFont(new Font("Garamond", Font.BOLD, 16));

button4 = new JButton("BROWSE");
getContentPane().add(button4);
button4.addActionListener(this);
button4.setVisible(false);
button4.setFont(new Font("Garamond", Font.BOLD, 16));

Copened=0;
Cdecrypt=0;
FileInputStream ins;
FileOutputStream outs;
byte b[];
int len;
filechooser=new JFileChooser();
filechooser.setSelectionMode(JFileChooser.FILES_ONLY);
setSize(getPreferredSize());
addWindowListener(new WindowAdapter() {

```

```

    public void windowClosing(WindowEvent e) {
        System.exit(0);
    }
});
}
public void Imagedecrypt(File filename,int key)throws java.io.IOException
{
FileInputStream ins=new FileInputStream(filename);
byte b[]=new byte[2];
BigInteger bb1;
char mess[]=new char[8];
int c=0;
for(int i=0;i<key;i++)
{
    int n=ins.read();
}
for(int i=0;i<8;i++)
{
    ins.read(b);
    bb1=new BigInteger(b);
    String str=bb1.toString(2);
    int len=bb1.bitLength();
    if(b[0]<0)
        len++;
    char ch[]=new char[len+1];
    str.getChars(0,len,ch,0);
    if(b[0]==0)
        mess[i]='0';
    else
        mess[i]=ch[len-1];
}
}
}

```

```

    }
    String dd=new String(mess);
    BigInteger bb=new BigInteger(dd,2);
    String s=bb.toString(2);
    int l=bb.intValue();
    char me[]=new char[l];
    int count=0;

    for(int m=0;m<l;m++)
    {
    for(int i=0;i<8;i++)
    {
        ins.read(b);
        bb1=new BigInteger(b);
        String str=bb1.toString(2);
        int len=bb1.bitLength();
        if(b[0]<0)
            len++;
        char ch[]=new char[len+1];
        str.getChars(0,len,ch,0);
        if(b[0]==0)
            mess[i]='0';
        else
            mess[i]=ch[len-1];
    }
    String dd1=new String(mess);
    BigInteger bb2=new BigInteger(dd1,2);
    String s1=bb2.toString(2);
    int l1=bb2.intValue();
    me[count]=(char)l1;

```

```

count++;
}
String message=new String(me);
textarea3.setText(message);
ins.close();
}
public static void main(String args[] throws Exception
{
    Decryption dn = new Decryption();
    dn.setTitle("Decryption");
    dn.show();
    dn.pack();
    JFrame.setDefaultLookAndFeelDecorated(true);
        JDialog.setDefaultLookAndFeelDecorated(true);
        try
        {
            UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndF
eel");

            //javax.swing.UIManager.setLookAndFeel("com.birosoft.liquid.LiquidLookAndFeel"
);

                }
            catch (Exception ex)
            {
                System.out.println("Failed loading L&F: ");
                System.out.println(ex);
            }
        }
    public void actionPerformed(ActionEvent e)
    {
        try

```

```

{
String cmd;
cmd=e.getActionCommand();
if(cmd.equals("BACK"))
{
    dispose();
    Stegno s = new Stegno();
    s.show();
    s.pack();
    s.setTitle("Main Menu");
}
if(cmd.equals("NEXT"))
{
    if(Copened==1)
    {
        Dkey=JOptionPane.showInputDialog("Enter 4 digit Key For Decryption");
//String type
        if(Dkey.trim().equals(""))
            JOptionPane.showMessageDialog(this,"Invalid
Input","Error",JOptionPane.ERROR_MESSAGE);
        else
        {
            // decrypt the message
            int key=Integer.parseInt(Dkey);
            Imagedecrypt(Ofilename,key);
            Cdecrypt=1;
        }
    }
else
    {

```

```

        JOptionPane.showMessageDialog(this,"File
NotOpened","Error",JOptionPane.ERROR_MESSAGE);
    }
}
if(cmd.equals("BROWSE"))
{
    int r=filechooser.showOpenDialog(this);
    tempfilename=filechooser.getSelectedFile(); //File type
    if(r==JFileChooser.CANCEL_OPTION)

JOptionPane.showMessageDialog(this,"FileNotSelected","Error",JOptionPane.ERROR_ME
SSAGE);
    else
    {
        String name=tempfilename.getName();

        if(!name.endsWith(".jpg") && !name.endsWith(".gif") && !name.endsWith(".bmp")
&& !name.endsWith(".jpeg"))

        JOptionPane.showMessageDialog(this,"Select Only Encrypted Image file", "Error",
JOptionPane.ERROR_MESSAGE);
    else
    {
        Copened=1;
        Ofilename=tempfilename;
        textfield1.setEditable(true);
        textfield1.setText(tempfilename.getPath());
        textfield1.setFont(new Font("Century",Font.PLAIN,15));
        textfield1.setBackground(Color.LIGHT_GRAY);
        //textfield1.setText(name);
        textfield1.setEditable(false);
    }
}
}

```

```

    }
    } // end try
catch(Exception ae)
{
OptionPane.showMessageDialog(this,e,"Error",JOptionPane.ERROR_MESSAGE);
}
}
class decryptionLayout implements LayoutManager {
    public decryptionLayout() {
    }
    public void addLayoutComponent(String name, Component comp) {
    }
    public void removeLayoutComponent(Component comp) {
    }
    public Dimension preferredLayoutSize(Container parent) {
        Dimension dim = new Dimension(0, 0);
        Insets insets = parent.getInsets();
        dim.width = 700 + insets.left + insets.right;
        dim.height = 400 + insets.top + insets.bottom;
        return dim;
    }
    public Dimension minimumLayoutSize(Container parent) {
        Dimension dim = new Dimension(0, 0);
        return dim;
    }
    public void layoutContainer(Container parent) {
        Insets insets = parent.getInsets();
        Component c;
        c = parent.getComponent(0);
        if (c.isVisible()) {c.setBounds(insets.left+75,insets.top+64,112,32);}
    }
}

```

```

c = parent.getComponent(1);
if (c.isVisible()) {c.setBounds(insets.left+88,insets.top+112,112,32);}
c = parent.getComponent(2);
if (c.isVisible()) {c.setBounds(insets.left+75,insets.top+165,112,32);}
c = parent.getComponent(3);
if (c.isVisible()) {c.setBounds(insets.left+204,insets.top+294,134,42);}
c = parent.getComponent(4);
if (c.isVisible()) {c.setBounds(insets.left+380,insets.top+294,134,42);}
c = parent.getComponent(5);
if (c.isVisible()) {c.setBounds(insets.left+165,insets.top+64,390,32);}
c = parent.getComponent(6);
if (c.isVisible()) {c.setBounds(insets.left+200,insets.top+112,264,70);}
c = parent.getComponent(7);
if (c.isVisible()) {c.setBounds(insets.left+165,insets.top+110,390,150);}
c = parent.getComponent(8);
if (c.isVisible()) {c.setBounds(insets.left+570,insets.top+64,104,32);}
c = parent.getComponent(9);
if (c.isVisible()) {c.setBounds(insets.left+480,insets.top+112,104,32);}
}
}
}

```

4.2.3. IOT Router:

```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;
import java.awt.event.ActionListener;
import javax.imageio.ImageIO;
import java.sql.*;

```



```
import java.lang.*;
import java.awt.event.ActionEvent;
import java.awt.font.*;
import java.lang.String;
import java.awt.Component;
import java.awt.geom.*;
import javax.swing.text.EditorKit;
import javax.swing.event.MouseInputAdapter;
import java.awt.image.BufferedImage;
import javax.swing.text.Style;
import javax.swing.text.StyledDocument;
import javax.swing.text.Style;
import javax.swing.text.StyleConstants;
import javax.imageio.ImageIO;
import java.util.Vector;
import javax.swing.text.*;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.lang.String;
import java.lang.Byte;
import java.math.*;
import java.security.*;
import java.io.DataOutputStream;
import java.io.DataInputStream;
import java.io.Serializable;
import java.io.UnsupportedEncodingException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
```

```
import java.util.Random;
import javax.crypto.BadPaddingException;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import java.io.FilterInputStream;
import java.io.FilterOutputStream;
import javax.crypto.*;
import java.math.BigInteger;
import java.security.AlgorithmParameters;
import java.security.interfaces.*;
import java.security.KeyException;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStream;
import java.io.OutputStream;

class Decryption extends JFrame implements ActionListener,Serializable {
    JLabel label1;
    JLabel label2;
    JLabel label3;
    JButton button1;
    JButton button2;
    JTextField textfield1;
    JTextField textfield2;
    JTextArea textarea3;
    JScrollPane sp_textarea3;
    JButton button3;
    JButton button4;
    JFileChooser filechooser;
    File f,tempfilename,Ofilename,Sfilename;
```

```
int Copened,Cdecrypt;
String name,Dkey;
String chosenFile;

public Decryption()
{
    decryptionLayout customLayout = new decryptionLayout();
    getContentPane().setFont(new Font("Helvetica", Font.PLAIN, 12));
    getContentPane().setLayout(customLayout);
    getContentPane().setBackground(Color.YELLOW);

    label1 = new JLabel("SOURCE");
    getContentPane().add(label1);
    label1.setFont(new Font("Garamond", Font.BOLD, 16));

    label2 = new JLabel("IMAGE");
    getContentPane().add(label2);
    label2.setVisible(false);
    label2.setFont(new Font("Garamond", Font.BOLD, 16));

    label3 = new JLabel("MESSAGE");
    getContentPane().add(label3);
    label3.setToolTipText("Encrypted message is:");
    label3.setFont(new Font("Garamond", Font.BOLD, 16));

    button1 = new JButton("BACK");
    getContentPane().add(button1);
    button1.setFocusable(true);
    button1.setRolloverEnabled(true);
    button1.setVerifyInputWhenFocusTarget(true);
```

```
button1.addActionListener(this);
button1.setFont(new Font("Garamond", Font.BOLD, 18));

button2 = new JButton("NEXT");
getContentPane().add(button2);
button2.setFocusable(true);
button2.setRolloverEnabled(true);
button2.setVerifyInputWhenFocusTarget(true);
button2.addActionListener(this);
button2.setFont(new Font("Garamond", Font.BOLD, 18));

textfield1 = new JTextField("");
getContentPane().add(textfield1);
textfield1.setEditable(false);
textfield1.setBackground(Color.LIGHT_GRAY);
textfield1.setToolTipText("Choose the file to decrypt for message");

textfield2 = new JTextField("");
getContentPane().add(textfield2);
textfield2.setVisible(false);
textfield2.setToolTipText("Choose the image file to be encrypted");

textarea3 = new JTextArea("");
getContentPane().add(textarea3);
textarea3.setBackground(Color.LIGHT_GRAY);
textarea3.setToolTipText("The Encrypted Message is ::");
textarea3.setEditable(false);
sp_textarea3 = new JScrollPane(textarea3);
sp_textarea3.setWheelScrollingEnabled(true);
textarea3.setFocusable(true);
```

```

getContentPane().add(sp_textarea3);

button3 = new JButton("BROWSE");
getContentPane().add(button3);
button3.setFocusable(true);
button3.setRolloverEnabled(true);
button3.setVerifyInputWhenFocusTarget(true);
button3.addActionListener(this);
button3.setToolTipText("Select the Encrypted image file");
button3.setFont(new Font("Garamond", Font.BOLD, 16));

button4 = new JButton("BROWSE");
getContentPane().add(button4);
button4.addActionListener(this);
button4.setVisible(false);
button4.setFont(new Font("Garamond", Font.BOLD, 16));

Copened=0;
Cdecrypt=0;
FileInputStream ins;
FileOutputStream outs;
byte b[];
int len;

filechooser=new JFileChooser();
filechooser.setSelectionMode(JFileChooser.FILES_ONLY);
setSize(getPreferredSize());
addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
        System.exit(0);
    }
});

```

```

    }
    });
}
public void Imagedecrypt(File filename,int key)throws java.io.IOException
{
FileInputStream ins=new FileInputStream(filename);
byte b[]=new byte[2];
BigInteger bb1;
char mess[]=new char[8];
int c=0;
for(int i=0;i<key;i++)
{
int n=ins.read();
}
for(int i=0;i<8;i++)
{
ins.read(b);
bb1=new BigInteger(b);
String str=bb1.toString(2);
int len=bb1.bitLength();
if(b[0]<0)
len++;
char ch[]=new char[len+1];
str.getChars(0,len,ch,0);
if(b[0]==0)
mess[i]='0';
else
mess[i]=ch[len-1];
}
String dd=new String(mess);

```

```

BigInteger bb=new BigInteger(dd,2);
String s=bb.toString(2);
int l=bb.intValue();
char me[]=new char[l];
int count=0;

for(int m=0;m<l;m++)
{
for(int i=0;i<8;i++)
{
ins.read(b);
bb1=new BigInteger(b);
String str=bb1.toString(2);
int len=bb1.bitLength();
if(b[0]<0)
len++;
char ch[]=new char[len+1];
str.getChars(0,len,ch,0);
if(b[0]==0)
mess[i]='0';
else
mess[i]=ch[len-1];
}
String dd1=new String(mess);
BigInteger bb2=new BigInteger(dd1,2);
String s1=bb2.toString(2);
int l1=bb2.intValue();
me[count]=(char)l1;
count++;
}

```

```

String message=new String(me);
textarea3.setText(message);
ins.close();
}

public static void main(String args[]) throws Exception
{
    Decryption dn = new Decryption();
    dn.setTitle("Decryption");
    dn.show();
    dn.pack();
    JFrame.setDefaultLookAndFeelDecorated(true);
        JDialog.setDefaultLookAndFeelDecorated(true);
        try
        {
            UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndF
eel");

            //javax.swing.UIManager.setLookAndFeel("com.birosoft.liquid.LiquidLookAndFeel"
);
                }
            catch (Exception ex)
            {
                System.out.println("Failed loading L&F: ");
                System.out.println(ex);
            }
        }
    public void actionPerformed(ActionEvent e)
    {
        try
        {
            String cmd;

```



```

cmd=e.getActionCommand();
if(cmd.equals("BACK"))
{
    dispose();
    Stegno s = new Stegno();
    s.show();
    s.pack();
    s.setTitle("Main Menu");
}
if(cmd.equals("NEXT"))
{
    if(Copened==1)
    {
        Dkey=JOptionPane.showInputDialog("Enter 4 digit Key For Decryption");
//String type
        if(Dkey.trim().equals(""))
            JOptionPane.showMessageDialog(this,"Invalid
Input","Error",JOptionPane.ERROR_MESSAGE);
        else
        {
            // decrypt the message
            int key=Integer.parseInt(Dkey);
            Imagedecrypt(Ofilename,key);
            Cdecrypt=1;
        }
    }
    else
    {
        JOptionPane.showMessageDialog(this,"File
NotOpened","Error",JOptionPane.ERROR_MESSAGE);
    }
}

```

```

    }
    if(cmd.equals("BROWSE"))
    {
        int r=filechooser.showOpenDialog(this);
        tempfilename=filechooser.getSelectedFile(); //File type
        if(r==JFileChooser.CANCEL_OPTION)
        JOptionPane.showMessageDialog(this,"FileNotSelected","Error",JOptionPane.ERROR_MESSAGE);
        else
        {
            String name=tempfilename.getName();
            if(!name.endsWith(".jpg") && !name.endsWith(".gif") &&
            !name.endsWith(".bmp")&& !name.endsWith(".jpeg"))

                JOptionPane.showMessageDialog(this,"Select Only Encrypted Image
                file","Error",JOptionPane.ERROR_MESSAGE);
            else
            {
                Copened=1;
                Ofilename=tempfilename;
                textfield1.setEditable(true);
                textfield1.setText(tempfilename.getPath());
                textfield1.setFont(new Font("Century",Font.PLAIN,15));
                textfield1.setBackground(Color.LIGHT_GRAY);
                //textfield1.setText(name);
                textfield1.setEditable(false);
            }
        }
    }
} // end try
catch(Exception ae)

```

```

    {
OptionPane.showMessageDialog(this,e,"Error",OptionPane.ERROR_MESSAGE);
    }
}

class decryptionLayout implements LayoutManager {
    public decryptionLayout() {
    }
    public void addLayoutComponent(String name, Component comp) {
    }
    public void removeLayoutComponent(Component comp) {
    }
    public Dimension preferredLayoutSize(Container parent) {
        Dimension dim = new Dimension(0, 0);
        Insets insets = parent.getInsets();
        dim.width = 700 + insets.left + insets.right;
        dim.height = 400 + insets.top + insets.bottom;
        return dim;
    }
    public Dimension minimumLayoutSize(Container parent) {
        Dimension dim = new Dimension(0, 0);
        return dim;
    }
    public void layoutContainer(Container parent) {
        Insets insets = parent.getInsets();
        Component c;
        c = parent.getComponent(0);
        if (c.isVisible()) {c.setBounds(insets.left+75,insets.top+64,112,32);}
        c = parent.getComponent(1);
        if (c.isVisible()) {c.setBounds(insets.left+88,insets.top+112,112,32);}
        c = parent.getComponent(2);
    }
}

```

```

if (c.isVisible()) {c.setBounds(insets.left+75,insets.top+165,112,32);}
c = parent.getComponent(3);
if (c.isVisible()) {c.setBounds(insets.left+204,insets.top+294,134,42);}
c = parent.getComponent(4);
if (c.isVisible()) {c.setBounds(insets.left+380,insets.top+294,134,42);}
c = parent.getComponent(5);
if (c.isVisible()) {c.setBounds(insets.left+165,insets.top+64,390,32);}
c = parent.getComponent(6);
if (c.isVisible()) {c.setBounds(insets.left+200,insets.top+112,264,70);}
c = parent.getComponent(7);
if (c.isVisible()) {c.setBounds(insets.left+165,insets.top+110,390,150);}
c = parent.getComponent(8);
if (c.isVisible()) {c.setBounds(insets.left+570,insets.top+64,104,32);}
c = parent.getComponent(9);
if (c.isVisible()) {c.setBounds(insets.left+480,insets.top+112,104,32);}
}
}
}

```

4.2.4. Login:

```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;
import java.util.Vector;
import javax.swing.text.*;
import java.awt.*;
import java.lang.*;
import java.awt.image.*;
import javax.imageio.*;

```

```

import java.sql.*;
import java.lang.String;
import javax.swing.plaf.*;
import javax.swing.plaf.metal.*;
import java.io.Serializable;

public class Login extends JFrame implements ActionListener,Serializable {
    JLabel label1;
    JLabel label2;
    JTextField textfield1;
    JPasswordField passwordfield2;
    JButton button1;
    JLabel label3;

    public Login() {
        LoginLayout customLayout = new LoginLayout();

        getContentPane().setFont(new Font("Tahoma", Font.PLAIN, 12));
        getContentPane().setLayout(customLayout);
        getContentPane().setBackground(Color.YELLOW);

        label1 = new JLabel("Login Name");
        getContentPane().add(label1);
        label1.setFont(new Font("Bookman Old Style",Font.BOLD,20));

        label2 = new JLabel("Password");
        getContentPane().add(label2);
        label2.setFont(new Font("Bookman Old Style",Font.BOLD,20));
        textfield1 = new JTextField();
        getContentPane().add(textfield1);
    }
}

```

```

textfield1.setFont(new Font("RockWell",Font.PLAIN,18));

passwordfield2 = new JPasswordField();
getContentPane().add(passwordfield2);
passwordfield2.setEchoChar('^');
passwordfield2.setFont(new Font("SansSerif",Font.BOLD,20));

//ImageIcon icon1 = new ImageIcon("r1.jpg");
button1 = new JButton("LOGIN");
getContentPane().add(button1);
button1.addActionListener(this);
button1.setFont(new Font("Engravers MT",Font.BOLD,20));

//ImageIcon icon = new ImageIcon("r1.jpg");
//label3 = new JLabel("",icon,JLabel.LEFT);
//getContentPane().add(label3);
setSize(getPreferredSize());
addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
        System.exit(0);
    }
});
}

public static void main(String args[]) {
    Login window = new Login();
    window.setTitle("Congestion Avoidance for Smart Devices by Caching Information in
MANETS and IoT");
    window.pack();
    window.show();
    JFrame.setDefaultLookAndFeelDecorated(true);
        JDialog.setDefaultLookAndFeelDecorated(true);

```

```

        try
        {
            UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
        }
        catch (Exception ex)
        {
            System.out.println("Failed loading L&F: ");
            System.out.println(ex);
        }
    }
}

public void actionPerformed(ActionEvent e)
{
    String cmd;
    cmd=e.getActionCommand();
    if(cmd.equals("LOGIN"))
    {
        String user=textfield1.getText().trim().toLowerCase();
        String password=passwordfield2.getText().trim().toLowerCase();
        String DBusername ="Admin";
        String DBpassword = "Admin";

        if (DBusername.equalsIgnoreCase("Admin") &&
        DBpassword.equalsIgnoreCase("Admin"))
        {
            dispose();
            Stegno s = new Stegno();
            s.pack();
            s.show();
            s.setTitle("Steganography Implementation - Main Menu");
        }
    }
}

```

```

        }
        else
        {
            JOptionPane.showMessageDialog(null, "!!! ***Username and
password do not match*** !!!", "Error", JOptionPane.INFORMATION_MESSAGE);
            textfield1.setText("");
            passwordfield2.setText("");
        }
    }
}

class LoginLayout implements LayoutManager {
    public LoginLayout() {
    }
    public void addLayoutComponent(String name, Component comp) {
    }
    public void removeLayoutComponent(Component comp) {
    }
    public Dimension preferredLayoutSize(Container parent) {
        Dimension dim = new Dimension(0, 0);
        Insets insets = parent.getInsets();
        dim.width = 650 + insets.left + insets.right;
        dim.height = 400 + insets.top + insets.bottom;
        return dim;
    }
    public Dimension minimumLayoutSize(Container parent) {
        Dimension dim = new Dimension(0, 0);
        return dim;
    }
    public void layoutContainer(Container parent) {
        Insets insets = parent.getInsets();
        Component c;

```



```

c = parent.getComponent(0);
if (c.isVisible()) {c.setBounds(insets.left+150,insets.top+100,136,32);}
c = parent.getComponent(1);
if (c.isVisible()) {c.setBounds(insets.left+150,insets.top+150,136,32);}
c = parent.getComponent(2);
if (c.isVisible()) {c.setBounds(insets.left+280,insets.top+100,154,35);}
c = parent.getComponent(3);
if (c.isVisible()) {c.setBounds(insets.left+280,insets.top+150,154,35);}
c = parent.getComponent(4);
if (c.isVisible()) {c.setBounds(insets.left+276,insets.top+216,160,48);}
//c = parent.getComponent(5);
//if (c.isVisible()) {c.setBounds(insets.left+0,insets.top+0,700,600);}
}
}
}

```

4.2.5. Receiver_Decryption:

```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;
import java.awt.event.ActionListener;
import javax.imageio.ImageIO;
import java.sql.*;
import java.lang.*;
import java.awt.event.ActionEvent;
import java.awt.font.*;
import java.lang.String;
import java.awt.Component;
import java.awt.geom.*;

```

```
import javax.swing.text.EditorKit;
import javax.swing.event.MouseInputAdapter;
import java.awt.image.BufferedImage;
import javax.swing.text.Style;
import javax.swing.text.StyledDocument;
import javax.swing.text.Style;
import javax.swing.text.StyleConstants;
import javax.imageio.ImageIO;
import java.util.Vector;
import javax.swing.text.*;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.lang.String;
import java.lang.Byte;
import java.math.*;
import java.security.*;
import java.io.DataOutputStream;
import java.io.DataInputStream;
import java.io.Serializable;
import java.io.UnsupportedEncodingException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.Random;
import javax.crypto.BadPaddingException;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import java.io.FilterInputStream;
import java.io.FilterOutputStream;
```

```

import javax.crypto.*;
import java.math.BigInteger;
import java.security.AlgorithmParameters;
import java.security.interfaces.*;
import java.security.KeyException;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStream;
import java.io.OutputStream;

class Receiver_Decryption extends JFrame implements ActionListener,Serializable {
    JLabel label1;
    JLabel label2;
    JLabel label3;
    JButton button1;
    JButton button2;
    JTextField textfield1;
    JTextField textfield2;
    JTextArea textarea3;
    JScrollPane sp_textarea3;
    JButton button3;
    JButton button4;
    JFileChooser filechooser;
    File f,tempfilename,Ofilename,Sfilename;
    int Copened,Cdecrypt;
    String name,Dkey;
    String chosenFile;

    public Receiver_Decryption()
    {

```

```
decryptionLayout customLayout = new decryptionLayout();  
getContentPane().setFont(new Font("Helvetica", Font.PLAIN, 12));  
getContentPane().setLayout(customLayout);  
getContentPane().setBackground(Color.YELLOW);
```

```
label1 = new JLabel("SOURCE");  
getContentPane().add(label1);  
label1.setFont(new Font("Garamond", Font.BOLD, 16));
```

```
label2 = new JLabel("IMAGE");  
getContentPane().add(label2);  
label2.setVisible(false);  
label2.setFont(new Font("Garamond", Font.BOLD, 16));
```

```
label3 = new JLabel("MESSAGE");  
getContentPane().add(label3);  
label3.setToolTipText("Encrypted message is:");  
label3.setFont(new Font("Garamond", Font.BOLD, 16));
```

```
button1 = new JButton("EXIT");  
getContentPane().add(button1);  
button1.setFocusable(true);  
button1.setRolloverEnabled(true);  
button1.setVerifyInputWhenFocusTarget(true);  
button1.addActionListener(this);  
button1.setFont(new Font("Garamond", Font.BOLD, 18));
```

```
button2 = new JButton("NEXT");  
getContentPane().add(button2);
```

```
button2.setFocusable(true);
button2.setRolloverEnabled(true);
button2.setVerifyInputWhenFocusTarget(true);
button2.addActionListener(this);
button2.setFont(new Font("Garamond", Font.BOLD, 18));

textfield1 = new JTextField("");
getContentPane().add(textfield1);
textfield1.setEditable(false);
textfield1.setBackground(Color.LIGHT_GRAY);
textfield1.setToolTipText("Choose the file to decrypt for message");

textfield2 = new JTextField("");
getContentPane().add(textfield2);
textfield2.setVisible(false);
textfield2.setToolTipText("Choose the image file to be encrypted");

textarea3 = new JTextArea("");
getContentPane().add(textarea3);
textarea3.setBackground(Color.LIGHT_GRAY);
textarea3.setToolTipText("The Encrypted Message is ::");
textarea3.setEditable(false);
sp_textarea3 = new JScrollPane(textarea3);
sp_textarea3.setWheelScrollingEnabled(true);
textarea3.setFocusable(true);

getContentPane().add(sp_textarea3);

button3 = new JButton("BROWSE");
```

```

getContentPane().add(button3);
button3.setFocusable(true);
button3.setRolloverEnabled(true);
button3.setVerifyInputWhenFocusTarget(true);
button3.addActionListener(this);
button3.setToolTipText("Select the Encrypted image file");
button3.setFont(new Font("Garamond", Font.BOLD, 16));

button4 = new JButton("BROWSE");
getContentPane().add(button4);
button4.addActionListener(this);
button4.setVisible(false);
button4.setFont(new Font("Garamond", Font.BOLD, 16));

Copened=0;
Cdecrypt=0;
FileInputStream ins;
FileOutputStream outs;
byte b[];
int len;
filechooser=new JFileChooser();
filechooser.setSelectionMode(JFileChooser.FILES_ONLY);
setSize(getPreferredSize());
addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
        System.exit(0);
    }
});
}

```

```

public void Imagedecrypt(File filename,int key)throws java.io.IOException
{
FileInputStream ins=new FileInputStream(filename);
byte b[]=new byte[2];
BigInteger bb1;
char mess[]=new char[8];
int c=0;
for(int i=0;i<key;i++)
{
int n=ins.read();
}
for(int i=0;i<8;i++)
{
ins.read(b);
bb1=new BigInteger(b);
String str=bb1.toString(2);
int len=bb1.bitLength();
if(b[0]<0)
len++;
char ch[]=new char[len+1];
str.getChars(0,len,ch,0);
if(b[0]==0)
mess[i]='0';
else
mess[i]=ch[len-1];
}
String dd=new String(mess);
BigInteger bb=new BigInteger(dd,2);
String s=bb.toString(2);

```

```

int l=bb.intValue();
char me[]=new char[l];
int count=0;
for(int m=0;m<l;m++)
{
for(int i=0;i<8;i++)
{
ins.read(b);
bb1=new BigInteger(b);
String str=bb1.toString(2);
int len=bb1.bitLength();
if(b[0]<0)
len++;
char ch[]=new char[len+1];
str.getChars(0,len,ch,0);
if(b[0]==0)
mess[i]='0';
else
mess[i]=ch[len-1];
}
String dd1=new String(mess);
BigInteger bb2=new BigInteger(dd1,2);
String s1=bb2.toString(2);
int l1=bb2.intValue();
me[count]=(char)l1;
count++;
}
String message=new String(me);
textarea3.setText(message);
ins.close();

```



```

}
public static void main(String args[]) throws Exception
{
    Receiver_Decryption dn = new Receiver_Decryption();
    dn.setTitle("Decryption");
    dn.show();
    dn.pack();
    JFrame.setDefaultLookAndFeelDecorated(true);
        JDialog.setDefaultLookAndFeelDecorated(true);
        try
        {

            UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndF
eel");

            //javax.swing.UIManager.setLookAndFeel("com.birosoft.liquid.LiquidLookAndFeel"
);
                }
            catch (Exception ex)
            {
                System.out.println("Failed loading L&F: ");
                System.out.println(ex);
            }
        }
    public void actionPerformed(ActionEvent e)
    {
        try
        {
            String cmd;
            cmd=e.getActionCommand();
            if(cmd.equals("EXIT"))
            {

```

```

        dispose();
    }
    if(cmd.equals("NEXT"))
    {
        if(Copened==1)
        {
            Dkey=JOptionPane.showInputDialog("Enter 4 digit Key For Decryption");
//String type
            if(Dkey.trim().equals(""))
                JOptionPane.showMessageDialog(this,"Invalid
Input","Error",JOptionPane.ERROR_MESSAGE);
            else
            {
                // decrypt the message
                int key=Integer.parseInt(Dkey);
                Imagedecrypt(Ofilename,key);
                Cdecrypt=1;
            }
        }
        else
        {
            JOptionPane.showMessageDialog(this,"File
NotOpened","Error",JOptionPane.ERROR_MESSAGE);
        }
    }
    if(cmd.equals("BROWSE"))
    {
        int r=filechooser.showOpenDialog(this);
        tempfilename=filechooser.getSelectedFile(); //File type
        if(r==JFileChooser.CANCEL_OPTION)

```

```

JOptionPane.showMessageDialog(this,"FileNotSelected","Error",JOptionPane.ERROR_MESSAGE);

    else

        {

            String name=tempfilename.getName();

            if(!name.endsWith(".jpg")) && !name.endsWith(".gif") &&
!name.endsWith(".bmp")&& !name.endsWith(".jpeg"))

                JOptionPane.showMessageDialog(this,"Select Only Encrypted Image
file","Error",JOptionPane.ERROR_MESSAGE);

            else

                {

                    Copened=1;

                    Ofilename=tempfilename;

                    textfield1.setEditable(true);

                    textfield1.setText(tempfilename.getPath());

                    textfield1.setFont(new Font("Century",Font.PLAIN,15));

                    textfield1.setBackground(Color.LIGHT_GRAY);

                    //textfield1.setText(name);

                    textfield1.setEditable(false);

                }

            }

        } // end try

    catch(Exception ae)

        {

JOptionPane.showMessageDialog(this,e,"Error",JOptionPane.ERROR_MESSAGE);

        }

    }

class decryptionLayout implements LayoutManager {

    public decryptionLayout() {

```

```

}
public void addLayoutComponent(String name, Component comp) {
}
public void removeLayoutComponent(Component comp) {
}
public Dimension preferredLayoutSize(Container parent) {
    Dimension dim = new Dimension(0, 0);
    Insets insets = parent.getInsets();
    dim.width = 700 + insets.left + insets.right;
    dim.height = 400 + insets.top + insets.bottom;
    return dim;
}
public Dimension minimumLayoutSize(Container parent) {
    Dimension dim = new Dimension(0, 0);
    return dim;
}
public void layoutContainer(Container parent) {
    Insets insets = parent.getInsets();

    Component c;
    c = parent.getComponent(0);
    if (c.isVisible()) {c.setBounds(insets.left+75,insets.top+64,112,32);}
    c = parent.getComponent(1);
    if (c.isVisible()) {c.setBounds(insets.left+88,insets.top+112,112,32);}
    c = parent.getComponent(2);
    if (c.isVisible()) {c.setBounds(insets.left+75,insets.top+165,112,32);}
    c = parent.getComponent(3);
    if (c.isVisible()) {c.setBounds(insets.left+204,insets.top+294,134,42);}
    c = parent.getComponent(4);
    if (c.isVisible()) {c.setBounds(insets.left+380,insets.top+294,134,42);}
}

```

```

c = parent.getComponent(5);
if (c.isVisible()) {c.setBounds(insets.left+165,insets.top+64,390,32);}
c = parent.getComponent(6);
if (c.isVisible()) {c.setBounds(insets.left+200,insets.top+112,264,70);}
c = parent.getComponent(7);
if (c.isVisible()) {c.setBounds(insets.left+165,insets.top+110,390,150);}
c = parent.getComponent(8);
if (c.isVisible()) {c.setBounds(insets.left+570,insets.top+64,104,32);}
c = parent.getComponent(9);
if (c.isVisible()) {c.setBounds(insets.left+480,insets.top+112,104,32);}
}
}
}

```

4.3.PARAMETER EVALUATION:

For showing the efficiency of the proposed EGC system, carrier capacity, peak signal to noise ratio (PSNR), mean square error (MSE), and time complexity were evaluated.

The results of all these parameters were compared to some of the existing methods, such as LSB Steganography, FMO Steganography, and optimized modified matrix encoding (OMME) Steganography. Various graphs have been put-up to efficiently show the comparison between the proposed work and the existing methods.

The parameters are evaluated as follows:

4.3.1. Mean Square Error:

MSE is the amount of similarity and the range of distortion in an image and it also helps in the measurement of the amount of reliability N is the total pixel within the image, X is the initial image, and Y is the final Stegno image.

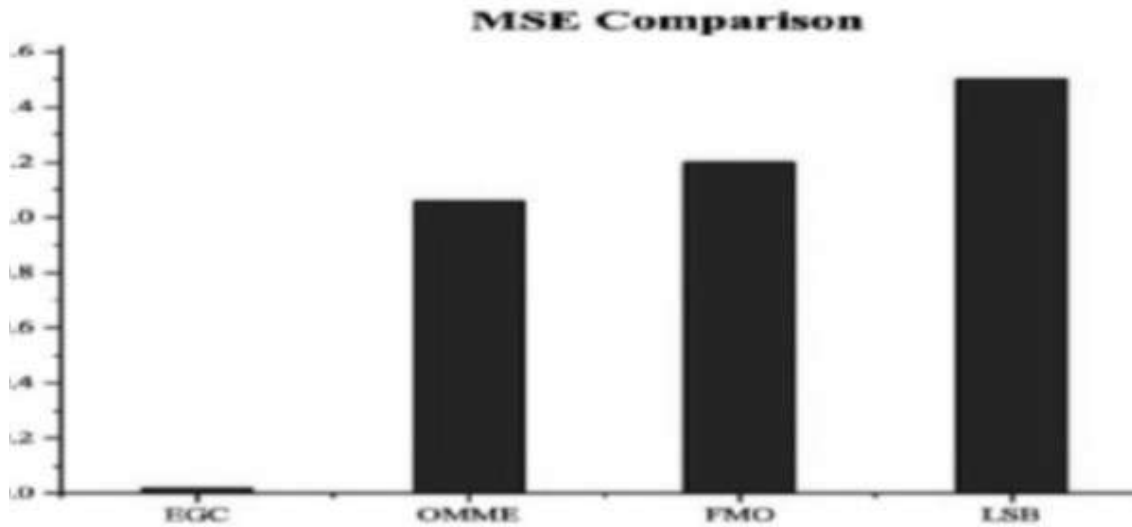


Fig: Mean Square Error Analysis

4.3.2. Peak Signal to Noise ratio:

PSNR calculates the invisibility of the image. PSNR can also be used for dynamic range images $PSNR = 10 \log_{10} \frac{225}{MSE}$.

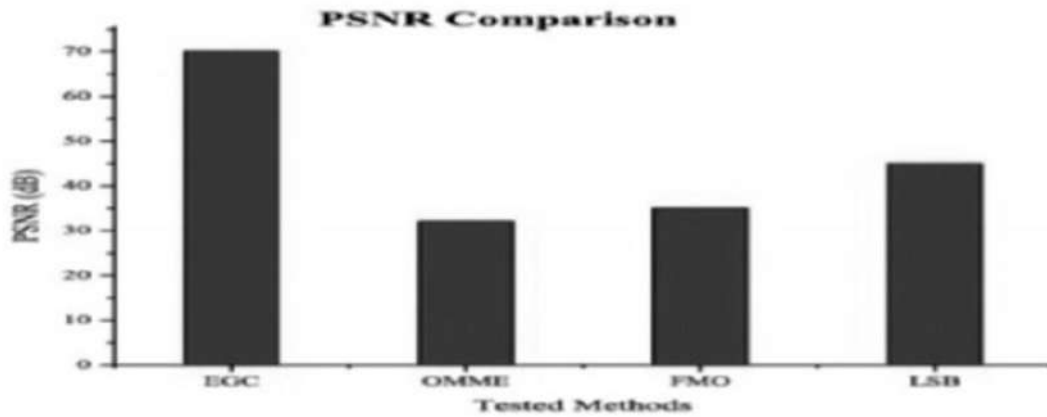


Fig. Peak signal to noise ratio analysis

4.3.3. Carrier Capacity (C):

Carrier capacity is the ability of the system to hide encrypted data inside the cover block. The value of carrier capacity is directly proportional to the performance. Carrier capacity is the other name for hiding capacity and is measured in terms of bits per pixel (BPP).

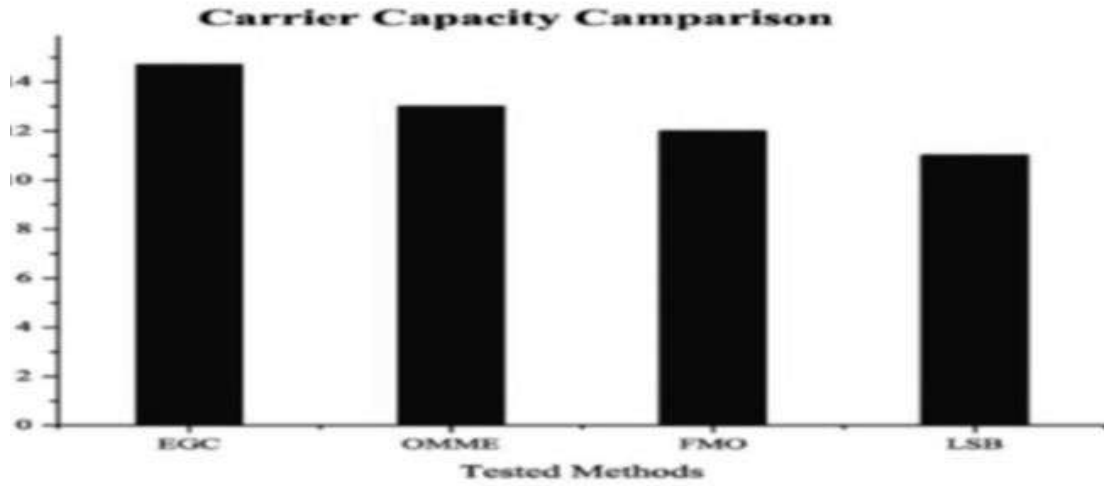


Fig. Carrier Capacity analysis

4.3.4. Time Complexity:

Time complexity is the amount of time taken between the encryption and decryption process. To increase the efficiency of the system, time complexity must be lowered.

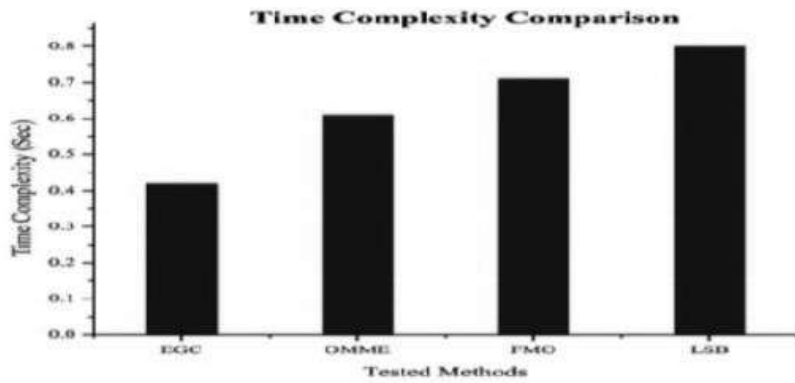


Fig. Time Complexity Comparison

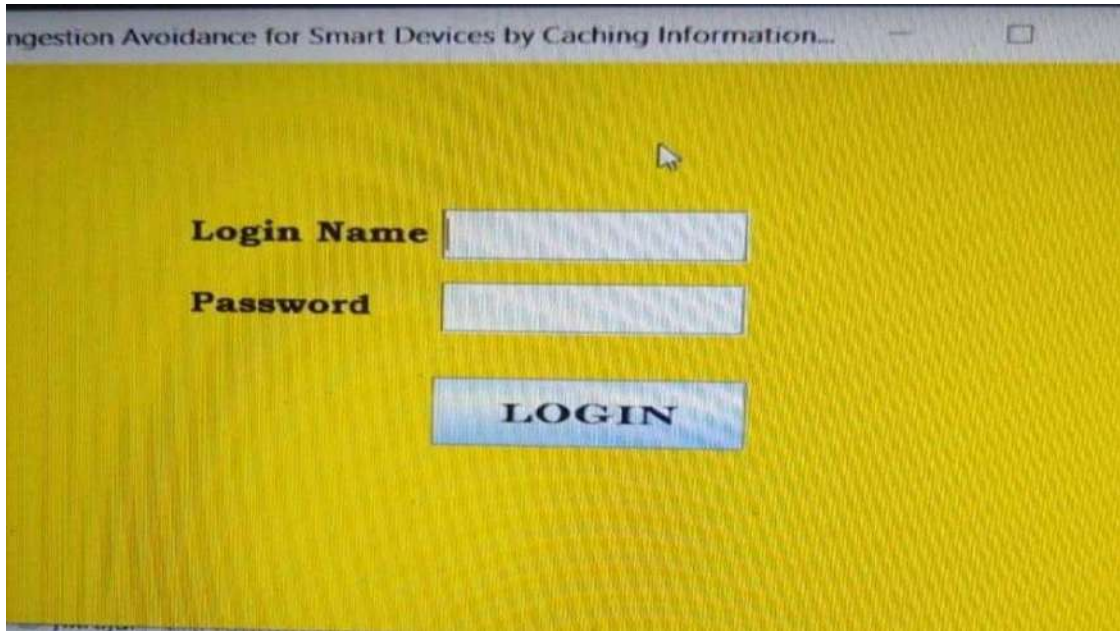
PARAMETER EVALUATION:

Sl. No	Parameters	Methods			
		EGC	OMME	FMO	LSB
1	Mean Square Error	0.02	1.0	1.2	1.5
2	Peak Signal to Noise Ratio	70	30	40	45
3	Carrier Capacity	15	13	11	10
4	Time Complexity	0.4	0.6	0.7	0.8

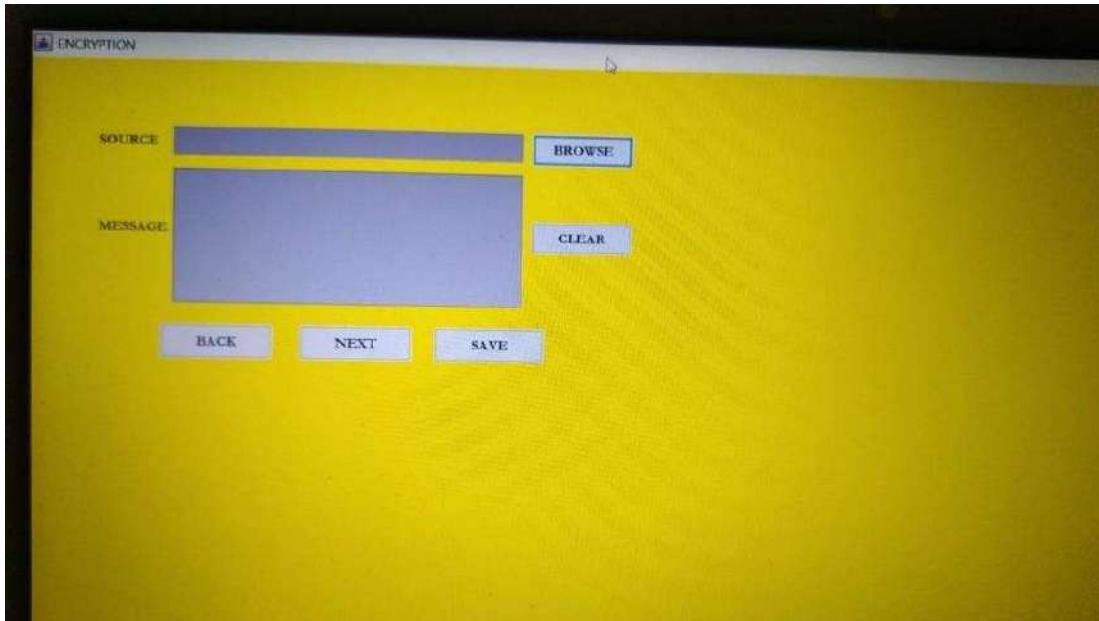
The EGC protocol gave better performance regarding carrier capacity, PSNR, MSE, and time complexity as compared to other techniques, such as LSB Steganography, FMO Steganography, and OMME Steganography. As depicted in Figs. 2 and 3, the MSE and time complexity of the proposed EGC protocol is very low, as compared to existing methods. The proposed protocol yielded better PSNR performance as compared to LSB, FMO, and OMME (32.42%, 45.62%, and 52.24% better performance as depicted in Fig. 2. Fig. 3 shows that the proposed protocol yielded better carrier capacity performance as compared to LSB, FMO, and OMME (0.33%, 16.35%, and 9.36% better performance, respectively). Therefore, overall the proposed method is well optimized and yielded better results when compared to the existing protocols.

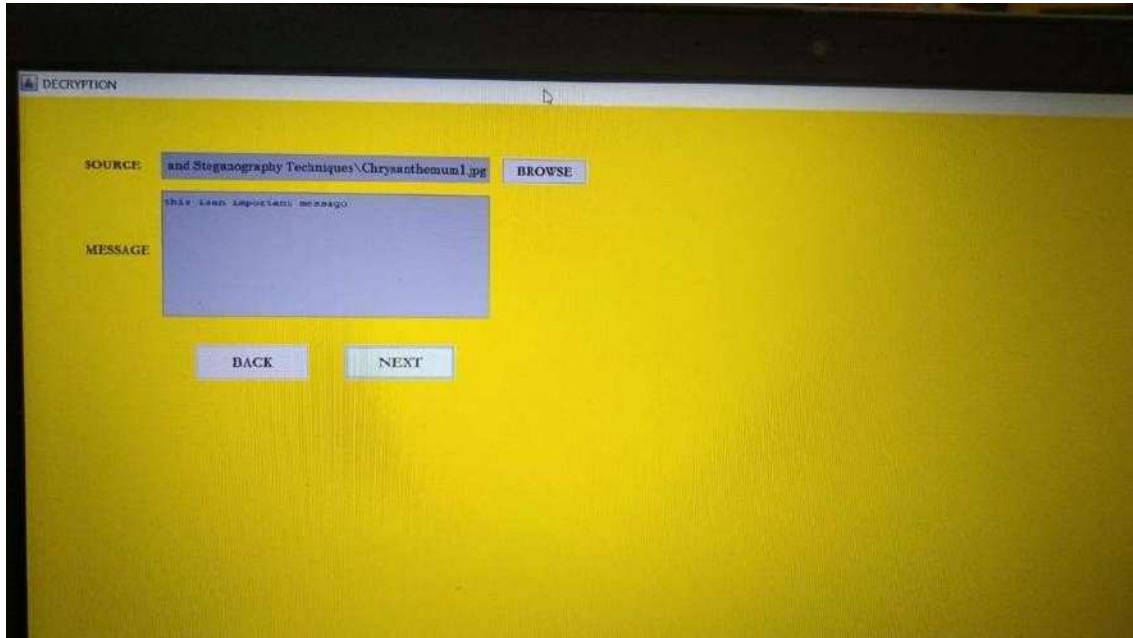
5. SCREENSHOTS

5.1. LOGIN PAGE:

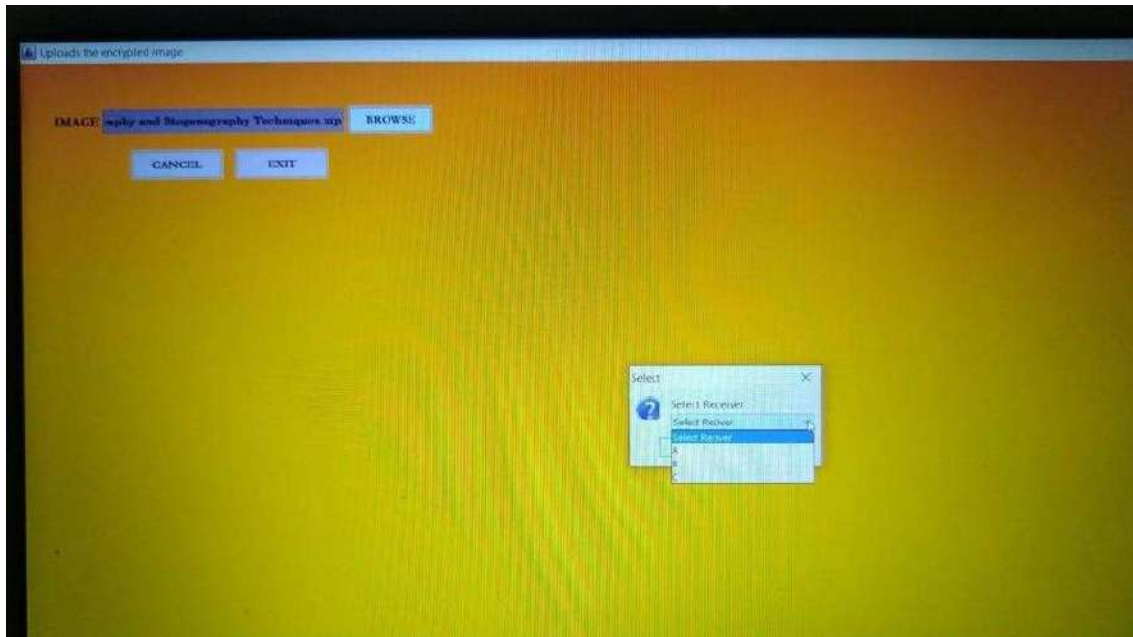


5.2. ENCRYPTION:

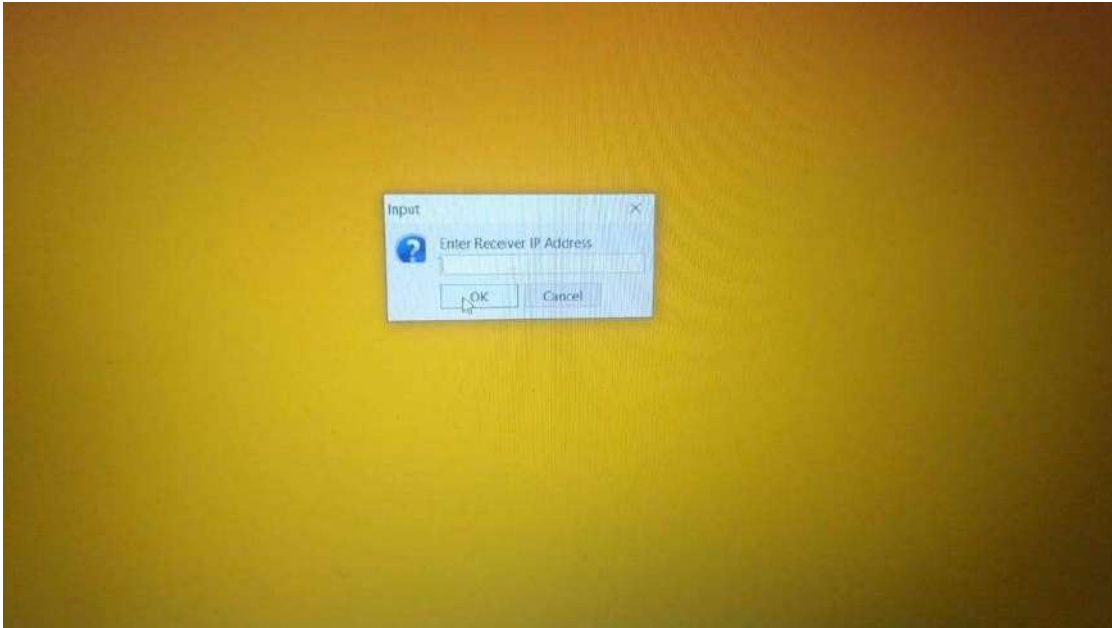




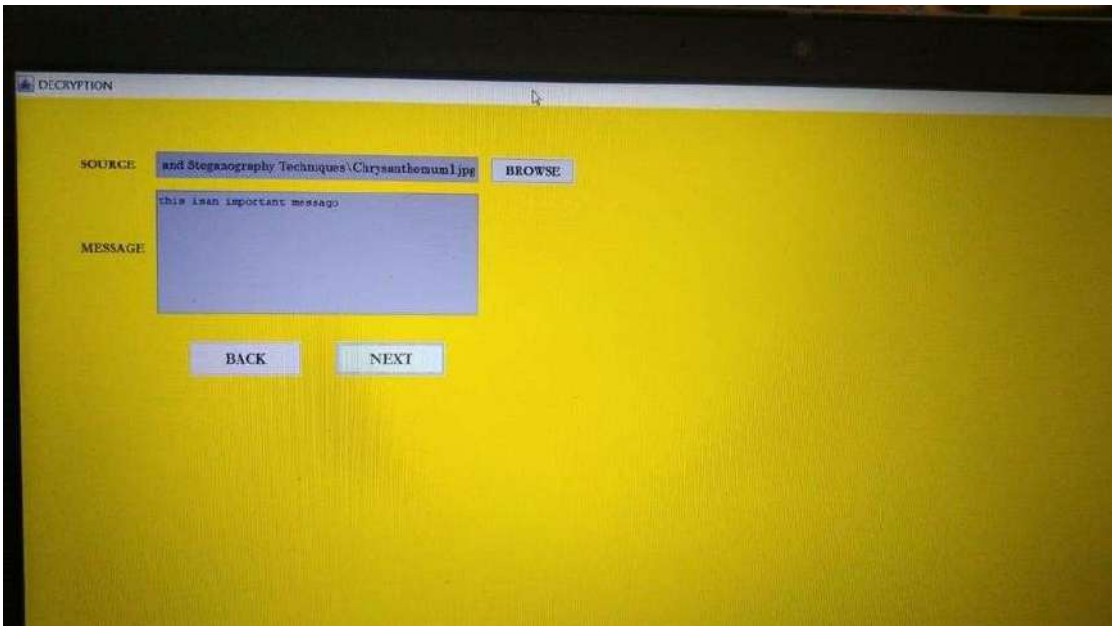
5.3. IOT ROUTER:



5.4. RECIEVER:



5.5. DECRYPTION:



6. TESTING

6. TESTING

6.1. INTRODUCTION TO TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

6.2. TYPES OF TESTING

6.2.1. UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2.2. INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfied, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.2.3. FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes.

6.2.4. SYSTEM TESTING

System testing ensures that the entire integrated software system meets requirements. It tests and sets the 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.

6.2.5. 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.

6.2.6. 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.

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.

6.3. TEST CASES

6.3.1. USER LOGIN

S No.	Test Cases	Condition being checked	Expected output	Actual output	Pass/Fail
1	Login screen must be opened	Cursor should be at user name	Cursor should be at user name	Cursor is at user name	Pass
2	Login screen must be opened	Cursor should be at password	Cursor should be at password	Cursor is at password	Pass
3	Valid/Invalid username	Valid username	If username is valid, it will login	Username valid	Pass
4	Valid/Invalid password	Valid password	If password is valid, it will login	Password valid	Pass

6.3.2. ADMIN LOGIN

S No.	Test Cases	Condition being checked	Expected output	Actual output	Pass/Fail
1	Admin screen must be opened	Cursor should be at username	Cursor should be at username	Cursor is at username	Pass
2	Admin screen must be opened	Cursor should be at password	Cursor should be at password	Cursor is at password	Pass
3	Valid/Invalid username	Valid username	If the username is valid, it will login	Username valid	Pass
4	Valid/Invalid username	Valid password	If the password is correct, it will login	Password valid	Pass

6.3.3. VIEW PROFILE

S No.	Test Cases	Condition being checked	Expected Output	Actual Output	Pass/Fail
1	User page must be opened	Cursor should be at view profile	Cursor should be at view profile	Cursor is at view profile	Pass
2	User page must be opened	Cursor should be at request kdc	Cursor should be at add image	Cursor is at add image	Pass
3	User page must be opened	Cursor should be at user search	Cursor should be at user search	Cursor is at user search	Pass

4	User page must be opened	Cursor should be at send friend request	Cursor should be at send friend request	Cursor is at send friend request	Pass
5	User page must be opened	Cursor should be at search other friends	Cursor should be at search other friends	Cursor is at search other friends	Pass
6	User page must be opened	Cursor should be at view my search history	Cursor should be at view my search history	Cursor is at view my search history	Pass
7	User page must be opened	Cursor should be at set policies on image	Cursor should be at set policies on image	Cursor is at set policies on image	Pass
8	User page must be opened	Cursor should be at recommend image to other friends	Cursor should be at recommend image to other friends	Cursor is at recommend image to other friends	Pass
9	User page must be opened	Cursor should be at view recommended images	Cursor should be at view recommended images	Cursor is at view recommended images	Pass
10	User page must be opened	Cursor should be at view all friends request	Cursor should be at view all friends request	Cursor is at view all friends request	Pass
11	User page must be opened	Cursor should be at logout	Cursor should be at logout	Cursor is at logout	Pass

7. CONCLUSION

7. CONCLUSION

In order to protect data during IOT transmission, the ECG protocol generated a high level of data security. The proposed ECG protocol improved security with the latest ECG on the Galois region. Because of the increased integration performance, advanced capabilities for hiding data are feasible. Due to the suggested specification and configuration of the adaptive firefly, some volume of the PSNR, carrier capacity, time complexity and the SSE are evaluated using parameters. Finally, the proposed work is implemented in a MATLAB simulator, and approximately 86% steganography embedding efficiency was achieved. Results from this proposed protocol were compared to existing methods, such as OMME, FMO, and LSB.

8. ACKNOWLEDGEMENT

8. ACKNOWLEDMENT

We thank **CMR Technical Campus** for supporting this paper titled with “SECURING DATA IN IOT USING CRYPTOGRAPHY AND STEGANOGRAPHY TECHNIQUES”, which provided good facilities and support to accomplish our work. We sincerely thank our Chairman, Director, Deans, Guide and faculty members for giving valuable suggestions and guidance in every aspect of our work.

9. BIBLIOGRAPHY

8. BIBLIOGRAPHY

1. R. H. Weber, "Internet of Things—New security and privacy challenges," *Comput. Law Security Rev.*, vol. 26, no. 1, pp. 23–30, 2010.
2. A. Ukil, J. Sen, and S. Koilakonda, "Embedded security for Internet of Things," in *Proc. 2nd Nat. Conf. Emerg. Trends Appl. Comput. Sci. (NCETACS)*, Mar. 2011, pp. 1–6.
3. W. Daniels *et al.*, "S μ V—the security microvisor: A virtualisation-based security middleware for the Internet of Things," in *Proc. ACM 18th ACM/IFIP/USENIX Middleware Conf. Ind. Track*, Dec. 2017, pp. 36–42.
4. U. Banerjee, C. Juvekar, S. H. Fuller, and A. P. Chandrakasan, "eeDTLS: Energy-efficient datagram transport layer security for the Internet of Things," in *Proc. GLOBECOM IEEE Glob. Commun. Conf.*, Dec. 2017, pp. 1–6.
5. G. Manogaran, C. Thota, D. Lopez, and R. Sundarasekar, "Big data security intelligence for healthcare industry 4.0," in *Cybersecurity for Industry 4.0*. Cham, Switzerland: Springer, 2017, pp. 103–126.
6. H. Sun, X. Wang, R. Buyya, and J. Su, "CloudEyes: Cloud-based malware detection with reversible sketch for resource-constrained Internet of Things (IoT) devices," *Softw. Pract. Exp.*, vol. 47, no. 3, pp. 421–441, 2017.
7. N. Chervyakov *et al.*, "AR-RRNS: Configurable reliable distributed data storage systems for Internet of Things to ensure security," *Future Gener. Comput. Syst.*, vol. 92, pp. 1080–1092, Mar. 2019.
8. S. Raza, H. Shafagh, K. Hewage, R. Hummen, and T. Voigt, "Lithe: Lightweight secure CoAP for the Internet of Things," *IEEE Sensors J.*, vol. 1, no. 10, pp. 3711–3720, Oct. 2013.
9. M. Vućinić *et al.*, "OSCAR: Object security architecture for the Internet of Things," *Ad Hoc Netw.*, vol. 32, pp. 3–16, Sep. 2015.
10. Y. Yang, X. Liu, and R. H. Deng, "Lightweight break-glass access control system for healthcare Internet-of-Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3610–3617, Aug. 2017.
11. A. K. Bairagi, R. Khondoker, and R. Islam, "An efficient steganographic approach for protecting communication in the Internet of Things (IoT) critical infrastructures," *Inf. Security J. Glob. Perspective*, vol. 25, nos. 4–6, pp. 197–212, 2016.

12. C.-T. Huang, M.-Y. Tsai, L.-C. Lin, W.-J. Wang, and S.-J. Wang, "VQ-based data hiding in IoT networks using two-level encoding with adaptive pixel replacements," *J. Supercomput.*, vol. 74, no. 9, pp. 4295–4314, 2018.
13. T. Shanableh, "Data hiding in MPEG video files using multivariate regression and flexible macroblock ordering," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 2, pp. 455–464, Apr. 2012.
14. X. Liao, J. Yin, S. Guo, X. Li, and A. K. Sangaiah, "Medical JPEG image steganography based on preserving inter-block dependencies," *Comput. Elect. Eng.*, vol. 67, pp. 320–329, Apr. 2018.
15. C. J. Benvenuto, *Galois Field in Cryptography*, Univ. Washington, Seattle, WA, USA, 2012. T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.
16. A. H. Gandomi, X. S. Yang, and A. H. Alavi, "Mixed variable structural optimization using firefly algorithm," *Comput. Struct.*, vol. 89, nos. 23–24, pp. 2325–2336, 2011.
17. R. Hegde and S. Jagadeesh a, "An optimal modified matrix encoding technique for secret writing in MPEG video using ECC," *Comput. Interfaces*, vol. 48, pp. 173–182, Nov. 2