A PROJECT REPORT ON

ENHANCED SECURITY IN STEGANOGRAPHY USING ENCRYPTION AND QUICK RESPONSE CODE

Project report submitted in partial fulfillment for the

requirement of award of B.Tech. Degree in Information Technology

by

115015002 - K. ABHILASH CHOUDARY

115015134 - S. SUDEEP GUPTA

Under the Guidance of

Prof. P. RAJENDIRAN

Assistant Professor III

SCHOOL OF COMPUTING

SHANMUGHA

ARTS, SCIENCE, TECHNOLOGY & RESEARCH ACADEMY

(SASTRA UNIVERSITY)

(A University Established under section 3 of the UGC Act, 1956)

TIRUMALAISAMUDRAM

THANJAVUR – 613 401

April 2015

SHANMUGHA
ARTS, SCIENCE, TECHNOLOGY & RESEARCH ACADEMY
(SASTRA UNIVERSITY)
(A University Established under section 3 of the UGC Act, 1956)
TIRUMALAISAMUDRAM
THANJAVUR – 613401

SCHOOL OF COMPUTING

BONAFIDE CERTIFICATE

This is to certify that the Project entitled
ENHANCED SECURITY IN STEGANOGRAPHY USING ENCRYPTION AND QUICK RESPONSE CODE
is a work done by

K. Abhilash Choudary 115015002
S. Sudeep Gupta 115015134

BACHELOR OF TECHNOLOGY
IN
INFORMATION TECHNOLOGY
OF SASTRA UNIVERSITY, Thanjavur during the year 2014-15

INTERNAL GUIDE **ASSOCIATE DEAN**

Submitted for University Examination held on___________________

EXAMINER - I **EXAMINER - II**

SHANMUGHA

ARTS, SCIENCE, TECHNOLOGY & RESEARCH ACADEMY

(SASTRA UNIVERSITY)

(A University Established under section 3 of the UGC Act, 1956)

TIRUMALAISAMUDRAM

THANJAVUR – 613401

DECLARATION

We submit this project work entitled "**Enhanced Security in Steganography using Encryption and Quick Response code** " to the Shanmugha Arts, Science, Technology & Research Academy (SASTRA) University, Tirumalaisamudram–613 401, in partial fulfillment of the requirement for the award of the degree of **BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY** and declare that it is our original and independent work carried out under the guidance of **Prof. P. Rajendiran**, Department of Information Technology, SASTRA.

Date:	**Name: K. Abhilash Choudary**	**Signature:**
	115015002	
Place:	**Name: S. Sudeep Gupta**	**Signature:**
	115015134	

ACKNOWLEDGEMENT

First and Foremost, I take pride in thanking the **Almighty** who gave me the strength for the successful completion of this project.

I take immense pride in thanking my **Parents** who were a constant source of encouragement and inspiration and provided their cooperation to enable me to complete this project to the fullest satisfaction.

I also would forever remain grateful to our **Prof. R. Sethuraman**, Vice Chancellor, **Dr. G. Balachandran**, Registrar, SASTRA University and **Dr. S. Vaidhyasubramaniam,** Dean of Planning and Development for giving us an opportunity to do this project.

I am extremely thankful to our dignified **Dr. P. Swaminathan, Dean, School Of Computing** and **Dr. N. Sairam**, **Associate Dean, Department of Information Technology**, **School of Computing**, for motivating me to bring out a perfect and successful project work.

I am highly grateful to our project internal guide **Prof. P. Rajendiran**, **Assistant Professor III, Department of Information Technology, School of Computing** for his timely help, valuable suggestion, guidance and moral support throughout the completion of the project.

I also thank all the **Teaching staff**, **Non-teaching staff** and those who have directly or indirectly helped me by extending their support and encouragement for completion of this project.

TABLE OF CONTENTS

CHAPTER 1

SYNOPSIS

Due to tremendous growth in communication technology, now it is a real challenge to send confidential data through communication network. Several information security systems, combining cryptography and steganography together have been introduced. The two-dimensional barcode, known as QR code is currently being developed to increase the encoding space. It is also necessary to build a perfect secure system where both cryptographic and steganographic methods may be clubbed together to make the system unbreakable. The secret message is encrypted first and hidden in a QR Code and then again that QR Code is embedded in a cover image, using the standard method of steganography. In this way the data, which is secured, is almost impossible to be retrieved without knowing the cryptography key, QR code encoding and the exact steganographic technique.

The proposed method suggests a combination of strong encrypting algorithm and steganographic technique to make the communication of confidential information safe, secure and extremely hard to decode. It includes an efficient encoding technique in encrypting a secret message before being fed in to the QR codes, which is encoding of the encrypted message and a steganographic technique for embedding the QR code into a cover image. The cover image is then transferred securely for exchanging secret information, from which the information is retrieved at the receiver's side.

The proposed data-hiding algorithm may be used in banking sectors, defense sector, educational sector, in e-Business etc. One can send secret key or password over internet. Users can get the information about secret key or data instantly by scanning the QR Code images without the need of memorization, from which the original key or data can be retrieved by decoding the obtained information from QR codes. This work can be further improvised upon in the future in a number of ways.

CHAPTER 2

SYSTEM CONFIGURATION

2.1 HARDWARE SPECIFICATIONS:

The hardware requirements for developing the proposed method

- The code was developed on a system with Intel i3 (2nd Gen) CPU operating @ 2.3 GHz.
- The Installed memory (RAM) with minimum of 2 GB.
- Windows 7/8 operating system.
- A Smartphone with a QR code scanning application installed in it.

2.2 SOFTWARE SPECIFICATONS:

The software requirements for developing the proposed method

- MATLAB 7 must be installed and should be in working condition, to develop the scrambling and embedding techniques.
- Eclipse LUNA must be installed and should be in working condition, to develop the encryption and QR code generation techniques.
- Java 8 SE must be installed in the system, to support the packages used in encryption and QR code generation.

CHAPTER 3

PROJECT DESCRIPTIION

3.1 Introduction:

Transferring the confidential data is a real challenge and is the need of the hour. Steganography is the art of concealing secret message in the images whereas cryptography is the process of altering the secret message into a distorted form, so that it is prevented from unauthorized access. Combining both the steganographic and cryptographic techniques will yield a secure and sophisticated system for exchanging the secret information between the sender and receiver. QR codes, also known as two-dimensional barcodes are used for increased encoding space. In this paper, a secure information model is built by combining cryptography and steganography. QR codes are employed for encoding the encrypted message. A nested image steganography is performed with QR codes on a suitable cover image. The proposed approach has a potential to be employed in communicating confidential information.

Due to advancements in communicating technologies and proliferation of internet facilities, it is a necessity to build a system for transferring confidential data securely over a network. Steganography involves concealing the message into images which are referred as cover images. These cover images are then transferred across the network securely. The hidden messages are overlooked and cannot be retrieved without knowing the exact steganographic technique. Cryptography includes modifying the message format by applying various techniques and finally rendering a distorted message which is not intelligible. Both the techniques have their own pros and cons. By combining both the techniques, a secure and more sophisticated system can be built.

3.2 Related Work:

3.2.1 Advanced Encryption Standard:

The Advanced Encryption Standard (AES), also known as Rijndael, is an efficient cryptographic algorithm that can be employed to protect electronic data. The design principle involved in AES is known as Substitution-Permutation network. AES operates on a 4×4 column-major order matrix of bytes, referred as the *state*. AES encryption is an iterative symmetric-key block cipher with varying key sizes (128, 192, and 256 bits). Symmetric-key ciphers use the same key to encrypt and decrypt data, unlike public-key ciphers, which utilizes a pair of keys. The key size of an AES cipher defines the repetitions of transformation rounds that change the input, referred as plaintext, into the final output, referred as the cipher text.

- 10 cycles of repetition for 128 –bit keys.
- 12 cycles of repetition for 192 –bit keys.
- 14 cycles of repetition for 256 –bit keys.

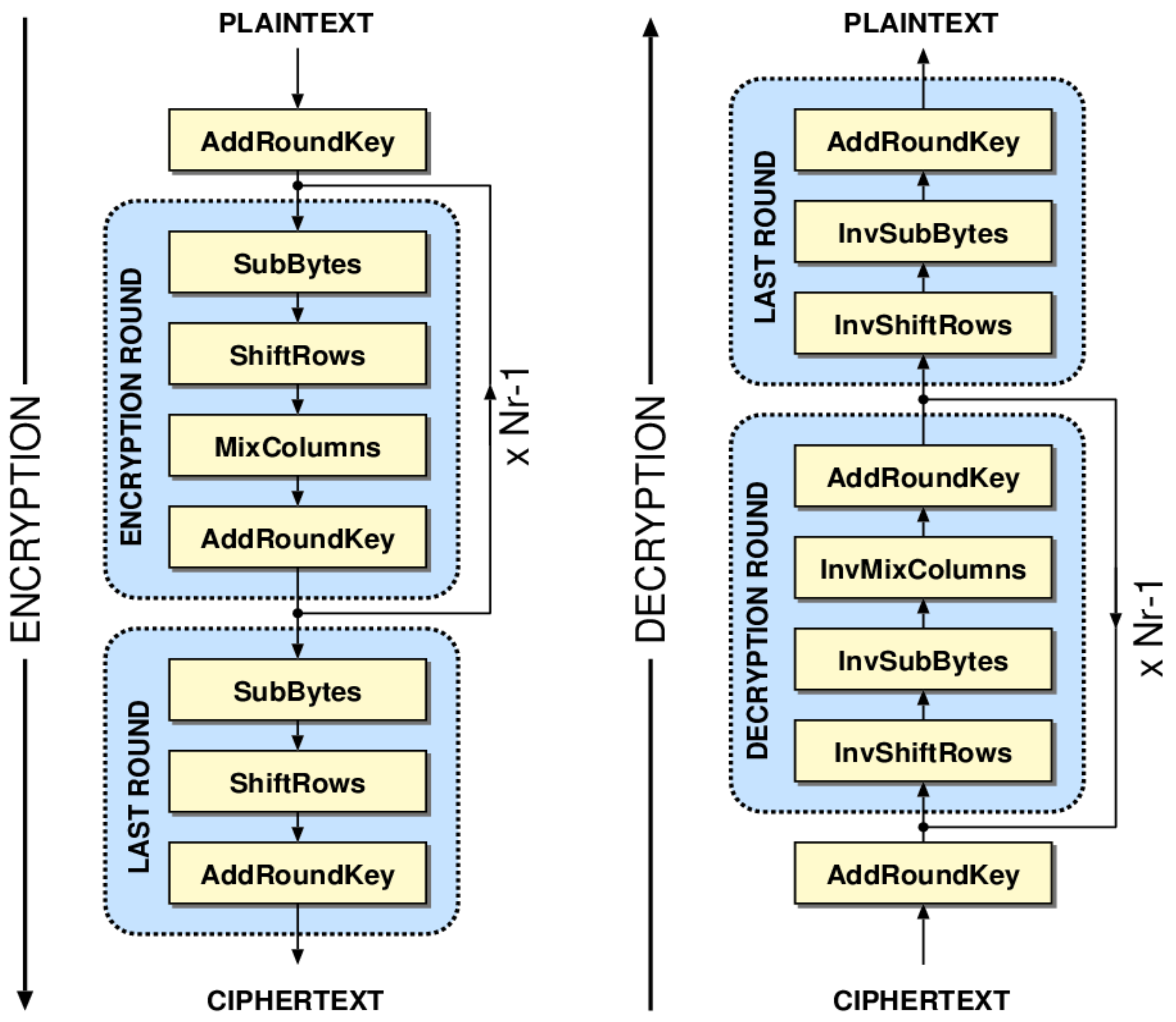


Fig: 1. AES Encryption and Decryption

AES encrypts and decrypts data in blocks of 128 bits (16 bytes). The returned encrypted data by block ciphers contains the exact number of bits as the input data. AES algorithm consists of several rounds each containing many processing steps. They include

Key Expansion

Round keys are derived from the cipher key using Rijndael's key schedule.

First Round

1. ***Add Round Key***: Each byte of the state is combined with a block of the round key using bitwise xor.

Rounds

1. Sub Bytes: a non-linear substitution step in which each byte is replaced with another according to a look up table

2*. *Shift Rows: a transposition step where the last three rows of the state are shifted cyclically a certain number of steps.

3. Mix Columns: a mixing operation which operates on the columns of the state, combining the four bytes in each column.

4. *Add Round Key:* Each byte of the state is combined with a block of the round key using bitwise xor.

Last Round

1. Sub Bytes: Each byte of the state is combined with a block of the round key using bitwise xor.

2. Shift Rows: a transposition step where the last three rows of the state are shifted cyclically a certain number of steps.

3. Add Round Key: Each byte of the state is combined with a block of the round key using bitwise xor.

AES has 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys. AES is regarded as the de facto standard for carrying out the encryption for all types of electronic data in telecommunications, banking and financial transactions, private and Federal information exchange.

3.2.2 Quick Response codes:

Quick Response codes alias QR codes are known as two dimensional codes are increasingly used now-a-days. Invented by Denso-Wave in 1994, QR codes are handy, portable and can be generated efficiently. The QR codes provide large space for encoding. QR Codes are generally employed for communicating small information like URL, a phone number or even small text. QR codes are the generalization of barcodes. QRs have been widely accepted as they are being used by large number applications on iOS and Android mobile platforms due to the advancements in smart-phone technology.

The QR codes can be generated easily from many online open sources and also by utilizing software. The decoding can be easily carried out with help of a smart phone equipped with a camera and a suitable decoding application. The QR codes come with a predefined structure, with specific allocations for information in encoding the data. QR codes also provide many desirable features like high capacity encoding of data, small size, resistant to dirt and damages. It is also readable from any direction.

The QR code has a cell architecture arranged in the square and is a matrix type symbol. The functionality patterns in the QR code enable the reading of the data area. The QR code architecture as shown in Fig.2, has position patterns, alignment patterns, timing patterns, quite zone, and data area. A symbol arranged at the three corners determines the position pattern, from which the position, the angle and the size of a QR code can be found. It is because of this the QR code can be detected in all directions. For adjusting nonlinear distortions Alignment patterns are extremely efficient. The distortion of the symbol can be corrected by identifying the central coordinate of the alignment pattern. The QR Code contains white and black patterns arranged in an alternate fashion, which is used by timing pattern for identifying the central coordinate of each cell. Its main purpose is, adjusting the central coordination of the data cell when the symbol is distorted. The quiet zone enables easy spotting of the code from the image by the CCD sensor.

Error Correction in QR Codes is based on Reed-Solomon Codes, a specific form of BCH error correction codes. There are four level of error correction that can be chosen by the user at creation time

.

1. L - 7%
2. M - 15%
3. Q - 25%
4. H - 30%

Higher error correction levels increase the percentage of code words used for error correction and therefore decrease the amount of data that can be stored inside the code.

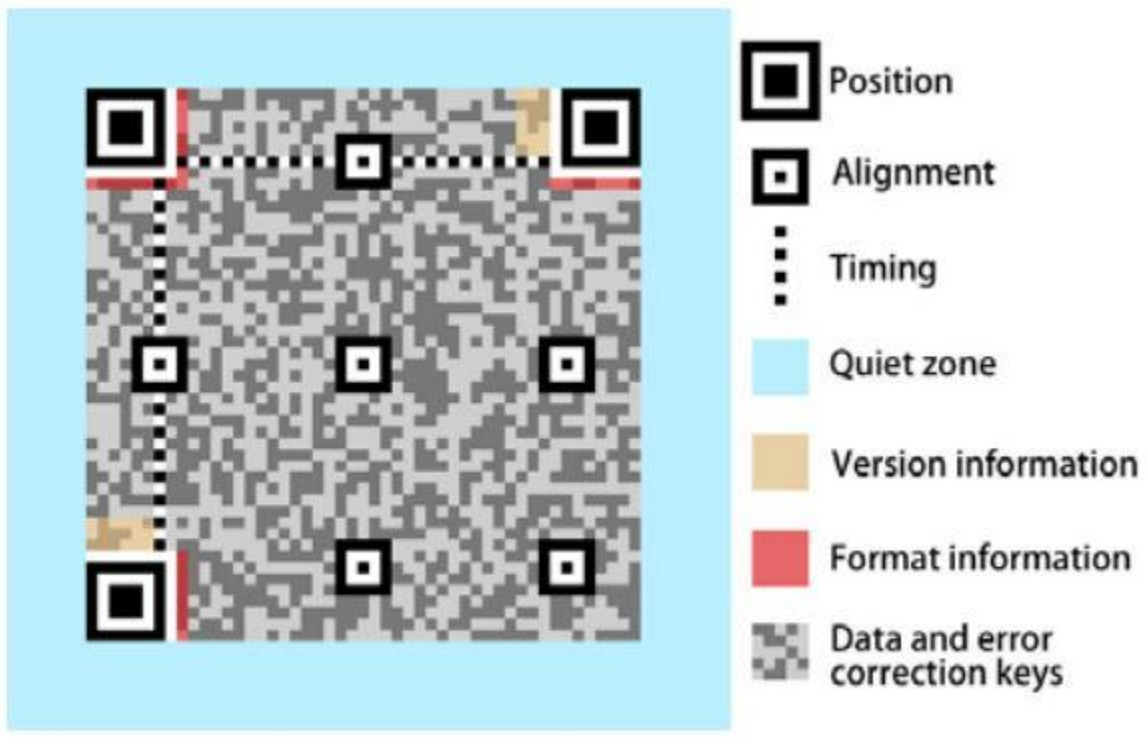


Fig.2 QR Code Architecture

3.2.3 Scrambling:

Image scrambling is a useful approach to secure the image data by scrambling the image into an unintelligible format. Image scrambling aims to mitigate security issues related to images. In image scrambling and descrambling, it is necessary to have simple algorithm to 'shuffle' the pixel values and reorder it to reveal the original. A pseudorandom sequence can be used to generate a scramble order. The amount of scrambling achieved determines the amount of distortion in the image. There are many ways to achieve scrambling in the images. Mostly scrambling approaches are grounded on Arnold Transform or on a combination of Arnold Transform and other techniques. Scrambling can also be achieved by employing random sequences grounded on chaos or pseudo random number generation grounded on parameters. The key based scrambling techniques are also reliable, where a key is generated which is shared by both parties to communicate. The main objective of image scrambling is to generate a non-intelligible image in order to prevent the understanding of the true content by human visual system or computer vision system.

3.2.4 Least Significant Bit (LSB) Technique:

LSB technique is employed to embed the bits of the message directly into the least significant bit plane of the cover image in a deterministic sequence. Modulating the least significant bit does not result in a human perceptible difference because the amplitude of the change is small. The implementation of LSB method is quite easy and it is a popular method. To hide a secret message inside an image, a proper cover image is needed. Because this method uses bits of each pixel in the image, it is necessary to use a lossless compression format, otherwise the hidden information will get lost in the transformations of a lossy compression algorithm. Digital image steganography is accomplished by using a common principle called least significant bit insertion. Each pixel contains a number of bytes that describe the color and appearance of the pixel. Depending on the resolution of that image, there are a set number of bytes for each pixel. When the LSB are removed from an image, it can be viewed as a gradient of redundant bits that resembles a black and white star burst. These bits are not really necessary for the integrity of the photography so these are the bits that are manipulated**.**

3.3 Proposed Methodology:

The use of QR codes in steganography is not novel but the novelty of the proposed methodology lies in the utilization of QR codes for achieving security levels in transmitting the secret message. The proposed methodology suggests a combination of strong encrypting algorithm and steganographic technique to make the communication of confidential information safe, secure and extremely hard to decode. It includes efficient encoding techniques in encrypting a secret message before encoding it in to the QR codes. The QR code is then scrambled to achieve another security level. Standard embedding technique is employed for embedding QR code in a suitable cover image. The cover image is then transferred securely for exchanging secret information, from which the information is retrieved using proposed retrieval procedure at the receiver's side.

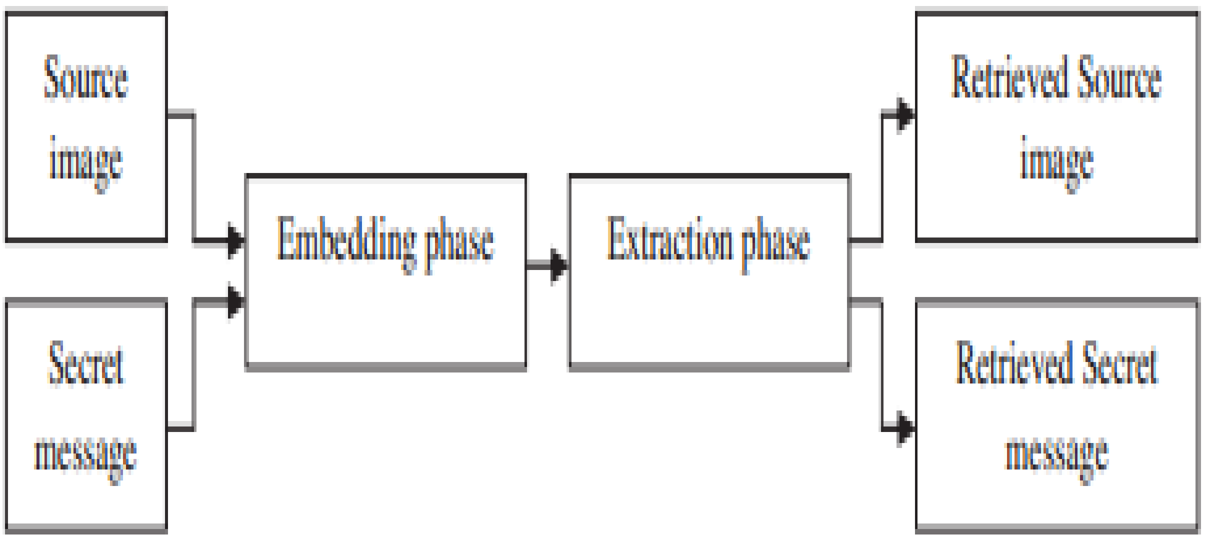


Fig.3. Framework for proposed methodology

The proposed methodology is categorized into four modules at both sender's side as well as receiver's side.

3.3.1 SENDER'S SIDE:

The procedure at the sender's side includes four modules. They are

1. Encryption
2. Encoding
3. Scrambling
4. Embedding

Every module imparts a layer of security to the secret message to be transmitted. The final stego image is the result of enhanced security imparted to the steganography. A detailed description of each module is given below.

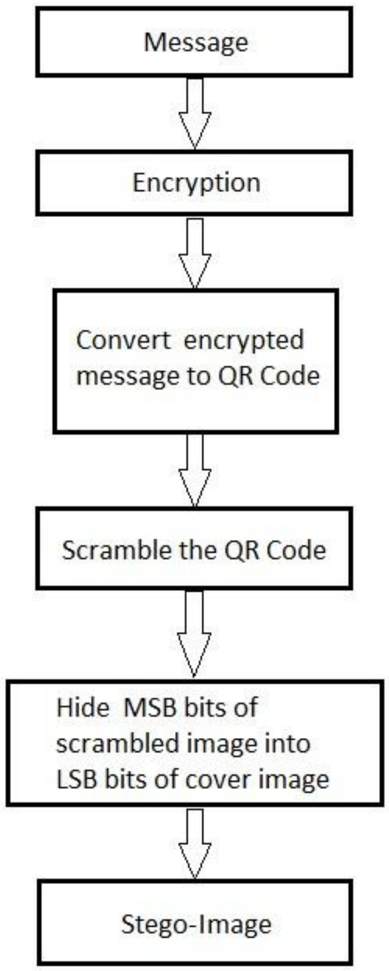

Message
Encryption
Convert encrypted message to QR Code
Scramble the QR Code
Hide MSB bits of scrambled image into LSB bits of cover image
Stego-Image

Fig.4. Block diagram for sender's side procedure

1. Encryption:

The first module at the sender’s side deals with the encryption of the secret message. The message which is to be conveyed is encrypted using an efficient encryption technique. In the proposed methodology, AES-128 key encryption technique is chosen for this purpose. The technique takes a 16 character password (8 bits per character) for encrypting the message. The encrypted message, in UTF-8 format is converted in to base64 format to make it compatible for further processing, which is then written in to file and stored for further processing.

Coding specifications:

The code for encrypting the secret message is written in java language and compiled in Eclipse IDE. The code takes two inputs,

i) A 16 character password,

ii) The message to be encrypted.

The inputs are given as command line arguments. The password is given as arg[0] and the message to be encrypted is given as arg[1], with the string separated by underscore(_) instead of space or else including space itself. The encrypted message is written in to the file.

2. Encoding:

The second module includes the encoding of the encrypted text (base64 format) in to QR code. For the encrypted text written in to the file, a unique QR code is generated. The QR encoding is unique and serves well for hiding the message. QR codes come with large encoding space. It can be transferred easily and has error correction capabilities.

Coding specifications:

The code for generating QR images is written in java programming language and executed on eclipse platform. The generation of QR codes is made possible by the utilization of Zebra Crossing (ZXing), an open source library and QRGEN, a library which creates a layer on top of ZXing to make it easier in java language. For this the ZXing and QRGEN jar files are included in the class path. It takes the encrypted text string as a static input and when compiled it generates a unique QR code for the concerned string at the specified location. The QR code can be generated in the required format.

3. Scrambling:

Scrambling includes realignment of the pixels of QR code to make it deceivable and also it make QR code elusive for human eyes. The main purpose of this manipulation is to prevent QR codes from getting scanned by QR scanners. A scramble order is generated at random and the RGB values are extracted for the QR code and both are stored. The random scramble order is applied on the RGB values extracted from the image and the same is communicated to the receiver's side. The resulting values of the scramble operation are reshaped according to the image size (Rows *Columns). Finally a scrambled image is generated concatenating the three RGB values and then generating an image from it.

Coding specifications:

i) randperm () - generates a row vector containing a random permutation of list of integers in the range of [1, n] instead of a single random number.

E.g. b= randperm (10)

ii) reshape() - reshapes array to the specified size.

Eg: r = reshape (IMG(; ;1), rows, columns)

iii) cat() - used to construct matrices along a specified dimension.

Eg: C=cat(1,A,B) concatenates along first dimension; A,B are matrices.

iv) Function to obtain the Red, Green, Blue matrix values of an image.

E.g. If IMG is an image

r=IMG (; ; 1) gives red color matrix

g=IMG (; ; 2) gives green color matrix

b=IMG (; ; 3) gives blue color matrix

To access the information of a particular pixel at row I and column J,

E.g. r(I,J) g(I,J) b(I,J)

Algorithm 1: Scrambling

Step 1: ***Read the image***

Step 2: ***Generate a random scramble order***

Step 3: ***Save the scramble order and the size of the image***

Step 4: ***Extract the RGB values of the image***

Step 5: ***Apply scramble order on RGB values***

Step 6: ***Reshape RGB values according to the image size***

Step 7: ***Concatenate the RGB values***

4. Embedding:

Embedding refers to embedding the secret image which in this case is scrambled QR code into a cover image. The embedding operation is performed to obtain a stego image, which is then transmitted securely to the receiver’s side. In this technique the bits of the image to be hidden are directly embedded into the least significant bit plane of the cover image in a deterministic sequence. Modulating the least significant bit does not result in a human perceptible difference as the amplitude of the change is little. To accomplish the digital image steganography a common principle called least significant bit insertion is employed. A pixel contains a number of bytes which describe the color and appearance of the pixel. There exist a set number of bytes for each pixel depending on the resolution of that image. When the LSB are removed from an image, it can be viewed as a gradient of redundant bits that resembles a black and white star burst. These bits are not really necessary for the integrity of the photography so these are the bits that are manipulated.

The insertion techniques available in LSB method are 1-bit insertion, 2-bit insertion, 3-bit insertion and 4-bit insertion. For embedding the QR image into the cover image a 4 bit insertion is utilized.

Coding specifications:

First, the image to be hidden is read and then the cover image is read by the code. Specify the names with their extensions. It is advisable to choose the images of similar dimensions. If they are not of similar dimensions also it is acceptable as initially in the code the size of both the input images is verified and compared. If required then the resizing of the images are done to fit for the purpose.

The main idea employed here is, the LSB (Least Significant Bits) of the cover image are replaced by the MSB (Most Significant Bits) of the image to be hidden. The functions used in this code are

i) bitand() - this performs bit by bit 'AND' operation of bits between the two images specified in arguments.

ii) bitshift() - this is used in order to shift the bits in an image. First argument takes image and Second argument specifies the number of positions to be shifted. If the number in second arguments is positive then Left shift is done else Right shift is done.

iii) bitor() - this performs bit by bit 'OR' operation of bits between the two images given as arguments.

Algorithm 2: LSB embedding

Step 1: ***Generate stream of binary bits from the cover image.***

Step 2: ***Bits of each pixel ANDed with 240(11110000).***

Step 3: ***Right shift by 4 bits on the bits of pixel in the scrambled image.***

Step 4: ***Value of bits of two is ORed.***

Step 5: ***Save the new image.***

3.3.2 RECEIVER'S SIDE:

The procedure at the receiver's side also includes four modules, which decodes the security levels. They are

1. Retrieval

2. Descrambling

3. Scanning

4. Decrypting

At the end of the process the secret message is rendered to the receiver accurately. Each process reverses the effect imparted at the sender's side.

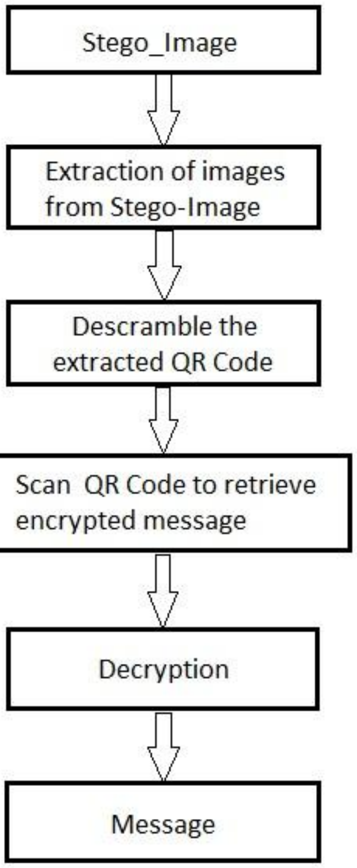

Stego_Image
Extraction of images from Stego-Image
Descramble the extracted QR Code
Scan QR Code to retrieve encrypted message
Decryption
Message

Fig.5. Block diagram of receiver's side procedure

1. Retrieval:

The stego image is processed to retrieve the hidden image from the cover image. The retrieved image is the scrambled QR code here. The quality of retrieved image is not much affected and is sufficient enough to retrieve the hidden content from it.

Coding specifications:

The code to retrieve the embedded image is written in MATLB. In the retrieval process bitand() is performed on encoded image to obtain one image and then the reverse shift is performed to obtain the second image. Thus, both the initial images are obtained with some details lost.

Algorithm 3: Retrieving the embedded image

Step 1: ***Generate the stream of binary bits for the received image***.

Step 2: ***Perform reverse operation to extract cover image***.

Step 3: ***Left shift by 4 bits on the received image***

Step 4: ***Retain the scrambled image with acceptable data loss.***

2. Descrambling:

The retrieved image is then descrambled using the descrambling technique by utilizing the received random sequence. The scrambled image is loaded and then a reverse order is generated from the file containing the scrambling order information. The RGB matrix values are then arranged according to the newly generated reverse order. The required QR code image can then be generated by concatenating these values to form an image.

Algorithm 4: Descrambling

Step 1: ***Load the image.***

Step2: Generate the reverse order.

Step3: Arrange the RGB matrix values.

Step4: Concatenate RGB values.

Step5: Obtain the descrambled image.

3. Scanning:

The descrambled QR code is then scanned to get the encoded information, which is here the secret message in encrypted form. The scanning can be easily carried out, with the help of mobile applications which are handy. They can be easily downloaded and installed. These applications use mobile's built in camera and a decoding program to scan and display the content of the encoded QR code.

Scanner specifications:

The QR code scanners are readily available in all the app stores and works well on android, windows and Mac operating systems.

4. Decryption:

The final phase, Decryption involves decrypting the encrypted message, to retrieve the secret message. It takes the same 16 character key, which is used to encrypt the message at the sender's side. The retrieved message is then delivered to the person concerned.

Coding Specifications:

The code to decrypt the encrypted message is written in java. The program takes two inputs.

i) The secret password,

ii) The encrypted text string(base64 format)

The inputs are given in the User Interface which are fed as command line arguments. The secret password is fed through first argument arg[0] , the encrypted string as second argument arg[1].The output of the program code, the secret message sent, is printed on the screen.

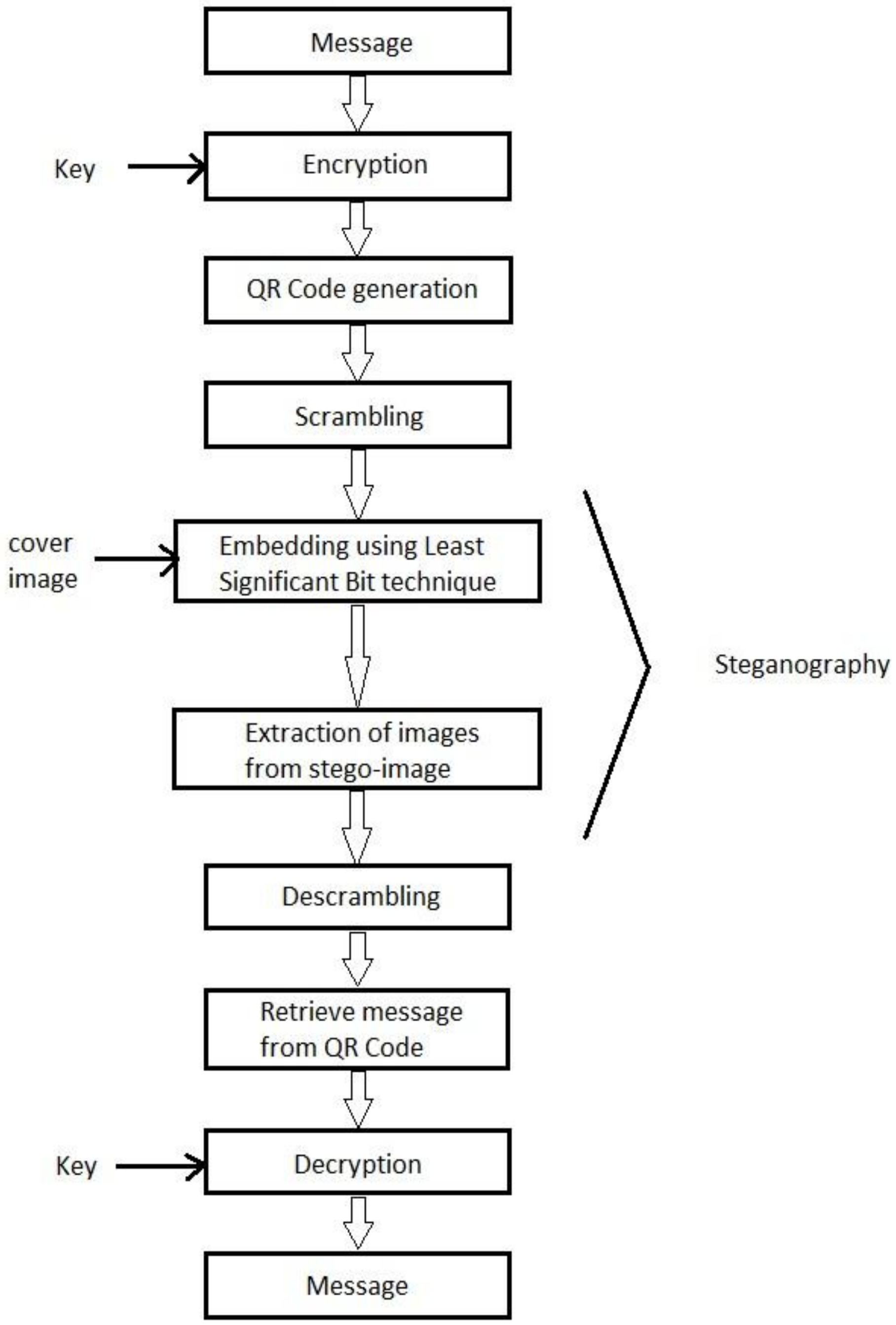

Message
Key
Encryption
QR Code generation
Scrambling
cover image
Embedding using Least Significant Bit technique
Steganography
Extraction of images from stego-image
Descrambling
Retrieve message from QR Code
Key
Decryption
Message

Fig.6. Block diagram of the proposed methodology

CHAPTER 4

DIAGRAMS

4.1 USE-CASE DIAGRAM:

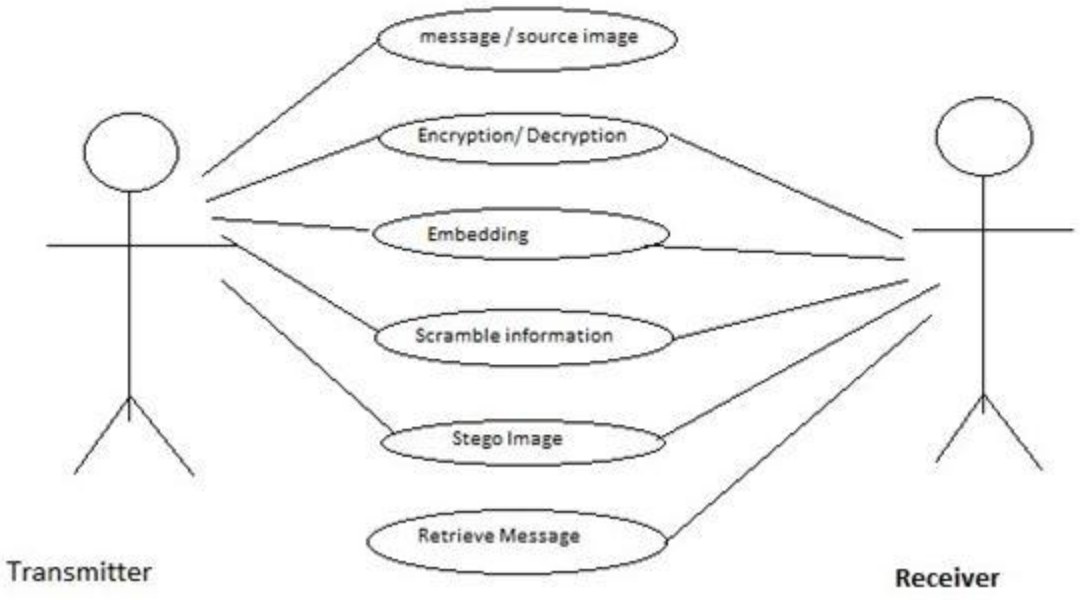


Fig.7. Use-case diagram for the proposed methodology

4.2 SEQUENCE DIAGRAM:

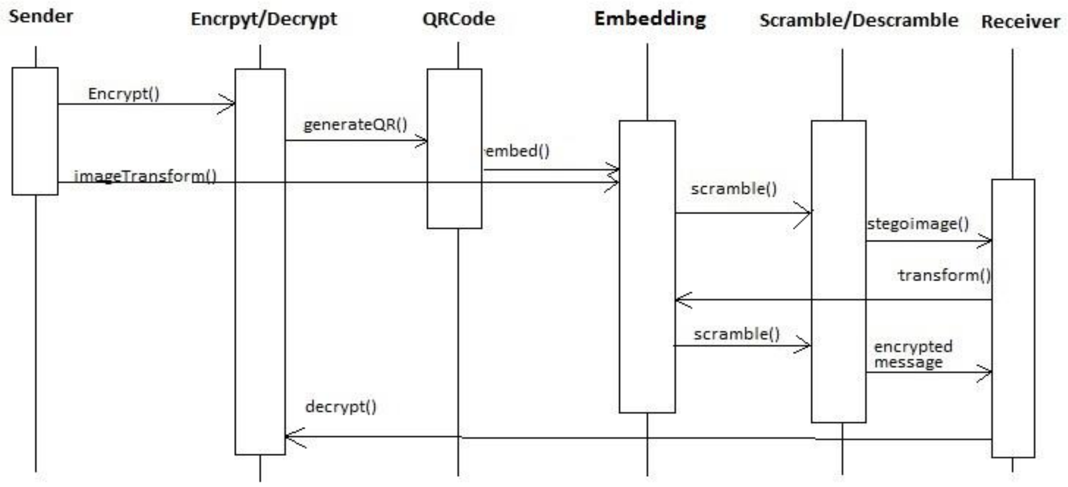


Fig.8. Sequence diagram for the proposed methodology

4.3 CLASS DIAGRAM:

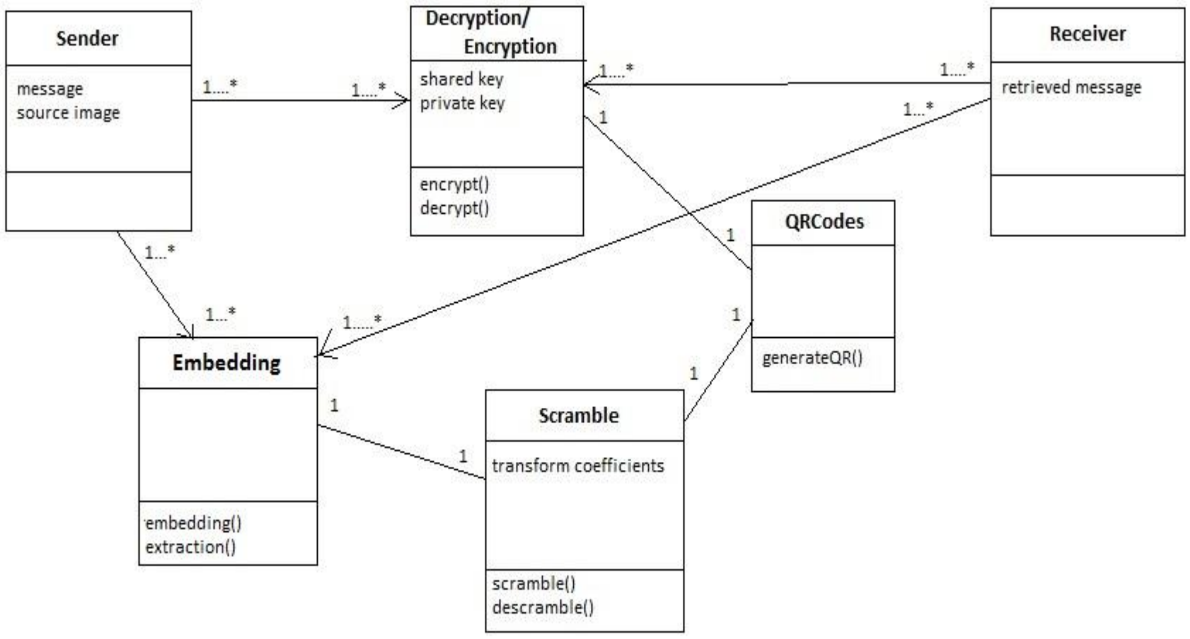


Fig.9. Class diagram for the proposed methodology

CHAPTER 5

SAMPLE SOURCE CODE

5.1 Encryption and QR code Generation

```
import net.glxn.qrgen.QRCode;

import net.glxn.qrgen.image.ImageType;

import javax.crypto.*;

import java.util.Base64;

import javax.crypto.spec.*;

import java.io.*;

import java.awt.*;

import java.awt.event.*;

import javax.swing.*;


 public class EncryptTest

  {

    String code;

      public static void main(String[] args) throws IOException

    {

    AwtControlDemo b=new AwtControlDemo();

    b.showTextFieldDemo();

     }

    public void start (String a[])

    {

        code = a[0];
    a[0] = null;
    for (String lines : a)
  {
       if (lines != null)
     {
           if (lines == a[1])
             {
               createFile(lines);
             }
           else
             {
               addToFile(lines);
             }
          }
     }
  }

public void createFile (String toEncode)
{
 try
  {
    SecretKeySpec skeySpec = new SecretKeySpec(code.getBytes(), "AES");
    Cipher cipher = Cipher.getInstance("AES");
    cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
    byte[] encrypted = cipher.doFinal(toEncode.getBytes());
```

```
      Base64.Encoder encoder = Base64.getEncoder();
      OutputStream out = new FileOutputStream("psswd.txt");
      String encryptedText = encoder.encodeToString(encrypted);
      out.write(encryptedText.getBytes());
      out.close();
    }
    catch (Exception e)
      {
         System.err.println("Error: " + e.getMessage());
      }
  }
   public void addToFile(String toEncode)
   {
   try
    {
     SecretKeySpec skeySpec = new SecretKeySpec(code.getBytes(), "AES");
      Cipher cipher = Cipher.getInstance("AES");
      Base64.Encoder encoder = Base64.getEncoder();
      cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
      byte[] encrypted = cipher.doFinal(toEncode.getBytes());
      String encrypt = encoder.encodeToString(encrypted);
      byte[] spacer = cipher.doFinal(System.getProperty("line.separator").getBytes());
      String encryptText = encoder.encodeToString(spacer);
      OutputStream out = new FileOutputStream("psswd", true);
      out.write(encryptText.getBytes());
```

```
        out.write(encrypt.getBytes());
            out.close();
      }
    catch (Exception e)
     {
            System.err.println("Error: " + e.getMessage());
      }
   }
}


 class QRCod{
            static String ttemp;
            public void demo() throws IOException
            {
BufferedReader br=new BufferedReader(new FileReader("PATH TO SAVE THE QR CODE"));
ttemp=br.readLine ();
ByteArrayOutputStream out =
QRCode.from(ttemp).to(ImageType.PNG).withSize(250,250).stream();
     try {
            FileOutputStream fout = new FileOutputStream(new File("PATH TO SAVE FILE"));
            fout.write (out.toByteArray ());
            fout.flush ();
            fout.close ();
        } catch (FileNotFoundException e) {
            // Do Logging
            e.printStackTrace();
        } catch (IOException e) {
            // Do Logging
            e.printStackTrace();
        }
    }
}
 class AwtControlDemo
            {
              private Frame mainFrame;
              private Label headerLabel;
              private Label statusLabel;
              private Panel controlPanel;
              public String a[];
              final TextField userText1 = new TextField(16);
              final TextField userText2 = new TextField(20);
              EncryptTest tester = new EncryptTest();

              public AwtControlDemo()
              {
                prepareGUI();
              }
                 private void prepareGUI()
               {
                mainFrame = new Frame("QR CODE PROJECT");
                mainFrame.setSize(500,500);
```

```
  mainFrame.setLayout(new GridLayout(3, 1));
  mainFrame.addWindowListener(new WindowAdapter()
{
    public void windowClosing(WindowEvent windowEvent)
{
      System.exit(0);
 }
 });
  headerLabel = new Label();
  headerLabel.setAlignment(Label.CENTER);
  statusLabel = new Label();
  statusLabel.setAlignment(Label.CENTER);
  statusLabel.setSize(350,100);
  controlPanel = new Panel();
  controlPanel.setLayout(new FlowLayout());
  mainFrame.add(headerLabel);
  mainFrame.add(controlPanel);
  mainFrame.add(statusLabel);
  mainFrame.setVisible(true);
}

public void showTextFieldDemo()
 {
  headerLabel.setText("PROJECT DEMO");
   Label  namelabel1= new Label("Key ", Label.RIGHT);
 }


 Label  nameLabel2 = new Label("Message ", Label.CENTER);

 //final TextField userText1 = new TextField(16);

 //final TextField userText2 = new TextField(20);

// userText2.setEchoChar('*');


 Button loginButton = new Button("Generate");


 loginButton.addActionListener(new ActionListener()

  {

    public void actionPerformed(ActionEvent e)

      {

       String data = "Key is:  " + userText1.getText();

       data += ", Message is:  " + userText2.getText();

       statusLabel.setText(data);

       String ab[]={userText1.getText(),userText2.getText()};

       tester.start(ab);

       QRCod q=new QRCod();

       try {

                       q.demo();

                  } catch (IOException e1)

                                 {

                                              // TODO Auto-generated catch block

                                              e1.printStackTrace();

                                 }

    }
```

```
        });
        controlPanel.add(namelabel1);
        controlPanel.add(userText1);
        controlPanel.add(nameLabel2);
        controlPanel.add(userText2);
        controlPanel.add(loginButton);
        mainFrame.setVisible(true);
      }
    }
```

5.2 Scrambling and Embedding:

```
fontSize = 16;
% Read in a color demo image.
folder = fileparts(which('cameraman.tif')); % Determine where demo folder is (works with all
versions).
baseFileName = 'newuntitled.jpg';% name of the image u want to scramble
% Get the full filename, with path prepended.
fullFileName = fullfile(folder, baseFileName);
if ~exist(fullFileName, 'file')
    % Didn't find it there.  Check the search path for it.
    fullFileName = baseFileName; % No path this time.
    if ~exist(fullFileName, 'file')
        % Still didn't find it.  Alert user.
        errorMessage = sprintf('Error: %s does not exist.', fullFileName);
        uiwait(warndlg(errorMessage));
```

```
                return;
        end
end
rgbImage = imread(fullFileName);
% Get the dimensions of the image.  numberOfColorBands should be = 3.
[rows, columns, numberOfColorBands] = size(rgbImage);
save('info2','rows','columns');
% Display the original color image.
%subplot(2, 2, 1);
%imshow(rgbImage);
%title('Original Color Image', 'FontSize', fontSize);
% Enlarge figure to full screen.
%set(gcf, 'units','normalized','outerposition',[0 0 1 1]);

% GETTING THE ORDER TO SCRAMBLE
scrambleOrder = randperm(rows*columns);
save('info','scrambleOrder')
% EXTRACTING THE RGB COLOR CHANNELS.
redChannel = rgbImage(:, :, 1);
greenChannel = rgbImage(:, :, 2);
blueChannel = rgbImage(:, :, 3);
% SCRAMBLING.
redChannel = redChannel(scrambleOrder);
greenChannel = greenChannel(scrambleOrder);
blueChannel = blueChannel(scrambleOrder);
```

```
% RESHAPING INTO A 2D IMAGE
redChannel = reshape(redChannel, [rows, columns]);
greenChannel = reshape(greenChannel, [rows, columns]);
blueChannel = reshape(blueChannel, [rows, columns]);

% RECOMBINE SEPARATE COLOR CHANNELS INTO A SINGLE, TRUE COLOR
RGB IMAGE.
scrambledImage = cat(3, redChannel, greenChannel, blueChannel);

% DISPLAY THE SCRAMBLED COLOR IMAGE
h=figure;
imshow(scrambledImage);
saveas(h,'scramble.jpg');
close all;

% EMBEDDING THE QR CODE IN  THE COVER IMAGE
disp(' ');
disp(' ***** EMBEDDING  IMAGE *****');
disp('___PROGRAM FOR EMBEDDING  ONE IMAGE INSIDE THE OTHER IMAGE___');
disp(' ');
disp('__________________________________________________________');
disp('___WELCOME ___');
```

```
X = imread(input(' ENTER THE COVER IMAGE FILE NAME: ','S'));

Y = imread(input(' ENTER THE FILE NAME OF IMAGE TO BE HIDDEN: ','S'));


% CHECK COMPATIBILITY

SX = size(X);

SY = size(Y);

if (SX(1) ~= SY(1))(SX(2)~=SY(2))

X=imresize(X,[SY(1),SY(2)]);

end


% CLEARING IST FILES LSB BITS & MOVING IIND FILES MSBITS BITS TO
LSBITS

X1 = bitand(X,uint8(240));

Y1 = bitshift(Y,-4);


% INSERTING IIND TO IST FILE

Z=bitor(X1,Y1);


% DISPLAY THE FIRST IMAGE

figure(1)

image(X);

xlabel(' IST IMAGE ');


% DISPLAY IIND IMAGE

figure(2);
```

```
image(Y);

xlabel(' IIND IMAGE ');

% DISPLAY ENCODED IMAGE

figure(3);

image(Z);

xlabel(' ENCODED IMAGE ');


% SAVING FILE

SAV=input('DO YOU WANT TO SAVE THE FILE Y/N [Y] ','S');

if SAV == 'Y'

NAME=input('ENTER A NAME FOR THE ENCODED IMAGE: ','S');

NAME=[NAME,'.BMP']; % CONCATINATION

imwrite(Z,NAME);

end

close all;
```

5.3 RETRIEVAL and DESCRAMBLING:

```
disp(' ');

disp('________________________________________________________');

disp('___WELCOME ___');

Z=imread(input('ENTER THE IMAGE FILE NAME TO BE DECODED:','S'));


% XO IS FIST FILE- OBTAINED BY CLEARING LSB BITS, YO IS IIND FILE RIGHT

% SHIFTING Z BY 4 BITS

XO=bitand(Z,uint8(240));
```

```
YO=bitshift(Z,4);



% DISPLAYING IST & IIND IMAGE FROM ENCODED IMAGE
figure;
image(YO);
xlabel('IIND DECODED IMAGE');
figure;
image(XO);
xlabel('IST DECODED IMAGE ');

% SAVING FILE
SAV=input('DO YOU WANT TO SAVE THE DECODED IMAGES Y/N [Y] ','S');
if SAV == 'Y'
NAME1=input('ENTER A NAME FOR THE DECODED IMAGE 1: ','S');
NAME1=[NAME,'.JPG']; % CONCATINATION
imwrite(XO,NAME1);

NAME2=input('ENTER A NAME FOR THE DECODED SCRAMBLED IMAGE 2: ','S');
NAME2=[NAME,'.JPG']; % CONCATINATION
imwrite(YO,NAME2);
end
close all;
```

```
% RECOVERING THE IMAGE KNOWING THE SORT ORDER
a=imread(NAME2);%name of scrambled image obtained
load('info2');
recoverOrder = zeros([rows*columns], 2);
recoverOrder(:, 1) = 1 : (rows*columns);
load('info');
recoverOrder(:, 2) = scrambleOrder;
% Sort this to find out where each scrambled location needs to be sent to.
newOrder = sortrows(recoverOrder, 2);
% Extract just column 1, which is the order we need.
newOrder = newOrder(:,1);

% DESCRAMBLING
redChannel = redChannel(newOrder);
greenChannel = greenChannel(newOrder);
blueChannel = blueChannel(newOrder);

% RESHAPING INTO 2D IMAGE
redChannel = reshape(redChannel, [rows, columns]);
greenChannel = reshape(greenChannel, [rows, columns]);
blueChannel = reshape(blueChannel, [rows, columns]);

% RECOMBINING COLORS INTO SINGLE, TRUE COLOR RGB IMAGE.
scrambledImage = cat(3, redChannel, greenChannel, blueChannel);
```

```
% DISPLAYING THE IMAGE.
a=figure
imshow(scrambledImage);
saveas(a,'RetrievedQRCode.jpg');
disp('The QR Code is saved as "RetrievedQRCode" ');
disp('It can be scanned to know the encoded information');
```

5.3 Decryption:

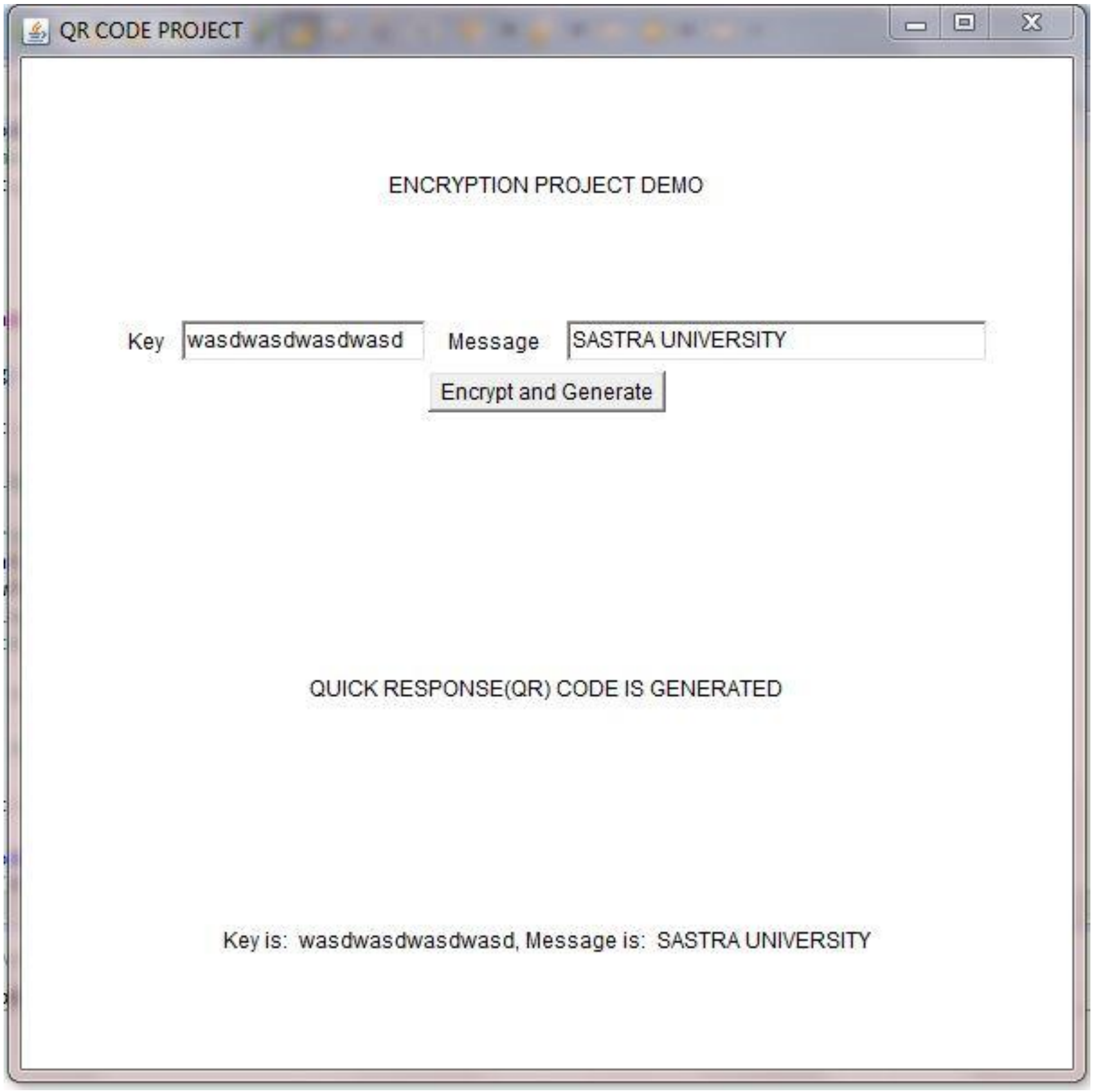
```
import java.security.*;
import javax.crypto.*;
import java.util.Base64;
import javax.crypto.spec.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;

public class DecryptTest {
   public static void main(String[] arg) throws Exception {


         AwtControlDemo a=new AwtControlDemo();
         a.showTextFieldDemo();
   }


   public String decrypt(String args[]) throws Exception{
      String code = args[0];
```

```
        Cipher cipher = Cipher.getInstance("AES");

        SecretKeySpec skeySpec = new SecretKeySpec(code.getBytes(), "AES");

        Base64.Decoder decoder = Base64.getDecoder();//byte[] encryptedTextByte =
decoder.decode(encryptedText);

        cipher.init(Cipher.DECRYPT_MODE, skeySpec);

        OutputStream out = new FileOutputStream("pssd.txt");

        String encryptedText = args[1];

        out.write(encryptedText.getBytes());

        File f=new File("psswd.txt");

        InputStream in = new FileInputStream(f);

        byte[] encrypted = new byte[(int)f.length()];

        in.read(encrypted);

        in.close();

        String encryp =new String(encrypted);

           byte[] encryptedTextByte = decoder.decode(encryp);

        byte[] original = cipher.doFinal(encryptedTextByte);

        String originalString = new String(original,"UTF-8");


        System.out.println("The result is\n\t"+originalString);

        return originalString;

    }

}


    class AwtControlDemo {

      private Frame mainFrame;
      private Label headerLabel;

      private Label statusLabel;

      private Panel controlPanel;

      public Label l;

      DecryptTest t=new DecryptTest();

      //String s1,s2;


      public AwtControlDemo(){

        prepareGUI();

      }


      private void prepareGUI(){

        mainFrame = new Frame("QR CODE PROJECT");

        mainFrame.setSize(500,500);

        mainFrame.setLayout(new GridLayout(4, 1));

        mainFrame.addWindowListener(new WindowAdapter() {

          public void windowClosing(WindowEvent windowEvent){

            System.exit(0);

          }

        });

        headerLabel = new Label();

        headerLabel.setAlignment(Label.CENTER);

        statusLabel = new Label();

        statusLabel.setAlignment(Label.CENTER);

        statusLabel.setSize(350,100);
        controlPanel = new Panel();
        controlPanel.setLayout(new FlowLayout());

        l=new Label("Result will be displayed here",Label.CENTER);

        mainFrame.add(headerLabel);
        mainFrame.add(controlPanel);
        mainFrame.add(l);
        mainFrame.add(statusLabel);
        mainFrame.setVisible(true);
      }

      public void showTextFieldDemo(){
        headerLabel.setText("PROJECT DEMO");

        Label  namelabel1= new Label("Key ", Label.RIGHT);
        Label  nameLabel2 = new Label("Message ", Label.CENTER);
        final TextField userText1 = new TextField(16);
        final TextField userText2 = new TextField(20);
       // userText2.setEchoChar('*');

        Button loginButton = new Button("Generate");

        loginButton.addActionListener(new ActionListener() {
          public void actionPerformed(ActionEvent e) {
            String data = "Key is:  " + userText1.getText();
            data += ", Message is:  " + userText2.getText();
            statusLabel.setText(data);
            String a[]={userText1.getText(),userText2.getText()};
            try {
                                    l.setText(t.decrypt(a));
                                } catch (Exception e1) {
                                    // TODO Auto-generated catch block
                                    e1.printStackTrace();
                                }
          }
        });
        //s1=userText1.getText();
        //s2=userText2.getText();

        controlPanel.add(namelabel1);
        controlPanel.add(userText1);
        controlPanel.add(nameLabel2);
        controlPanel.add(userText2);
        controlPanel.add(loginButton);
        mainFrame.setVisible(true);
    }
}
```

CHAPTER 6

SAMPLE OUTPUT

6.1 ENCRYPTION:


QR CODE PROJECT

ENCRYPTION PROJECT DEMO

Key: wasdwasdwasdwasd   Message: SASTRA UNIVERSITY

Encrypt and Generate

QUICK RESPONSE(QR) CODE IS GENERATED

Key is: wasdwasdwasdwasd, Message is: SASTRA UNIVERSITY

Fig.10. AES-128 BIT ENCRYPTION

6.2 GENERATED QR CODE FOR THE ENCRYPTED MESSAGE:

Fig.11. QR code

6.3 SCRAMBLED IMAGE:

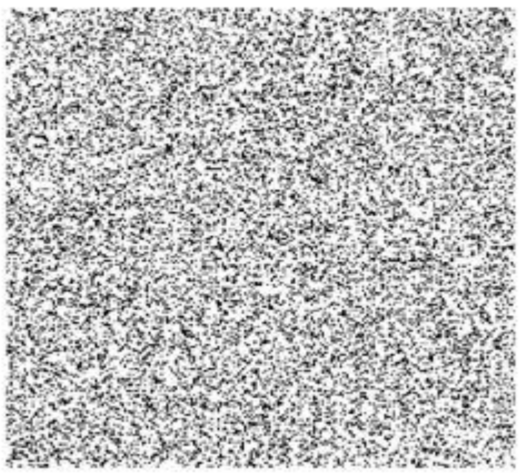


Fig.12. Scrambled image

6.4 EMBEDDING:

COVER IMAGE:

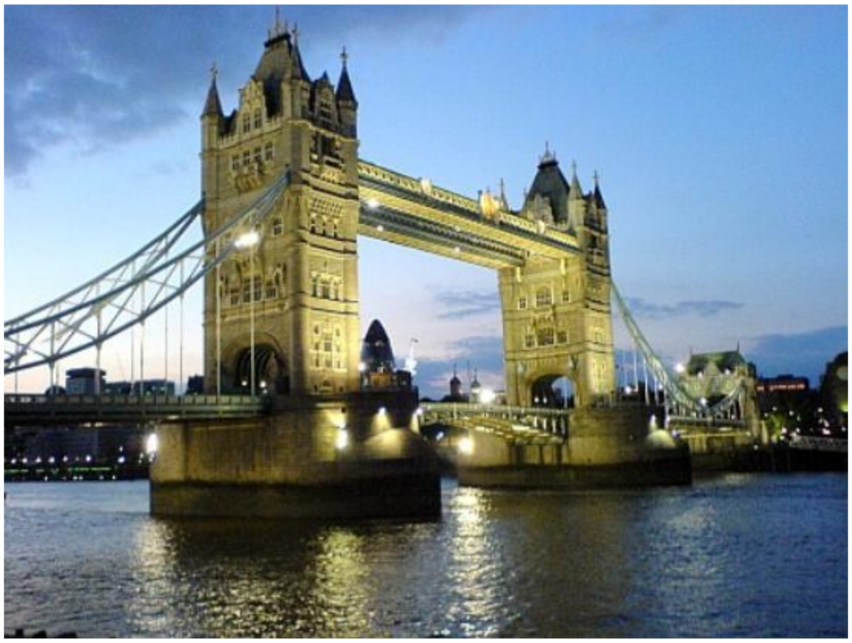


Fig.13. Cover image

STEGO IMAGE:

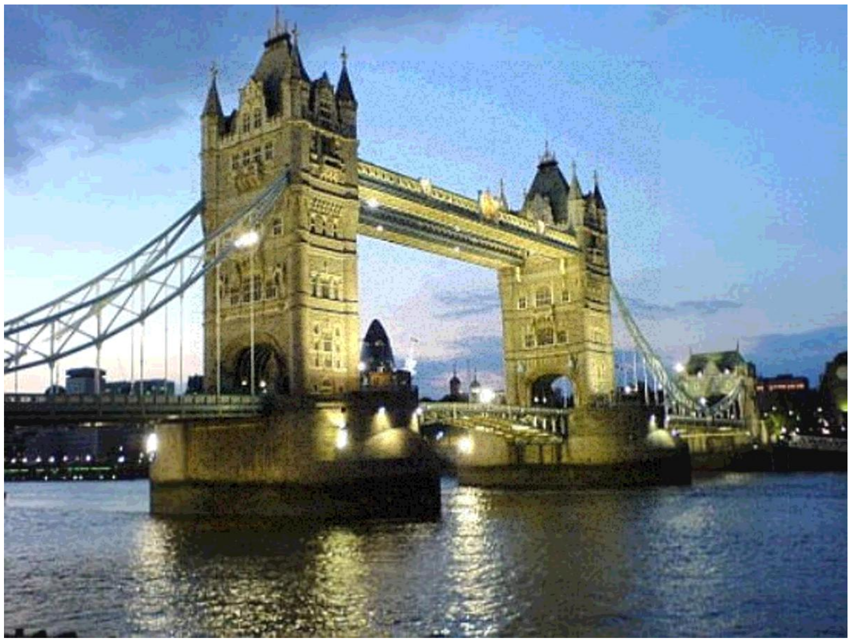


Fig.14. Stego image

6.5 RETRIEVED IMAGE:

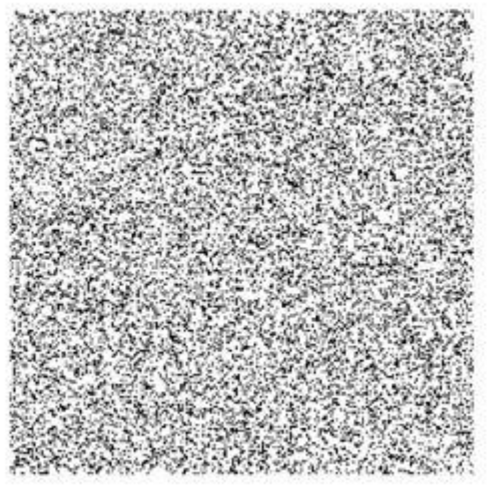


Fig.15. Retrieved scrambled image

6.6 RETRIEVED QR CODE:

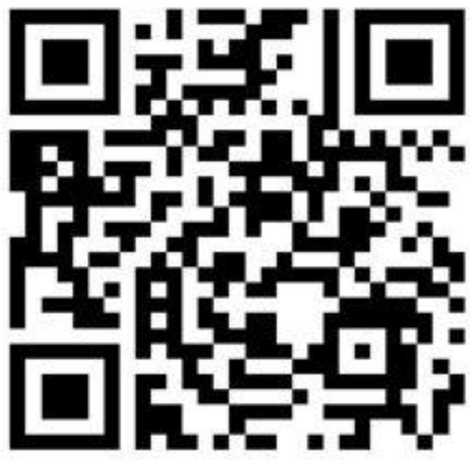
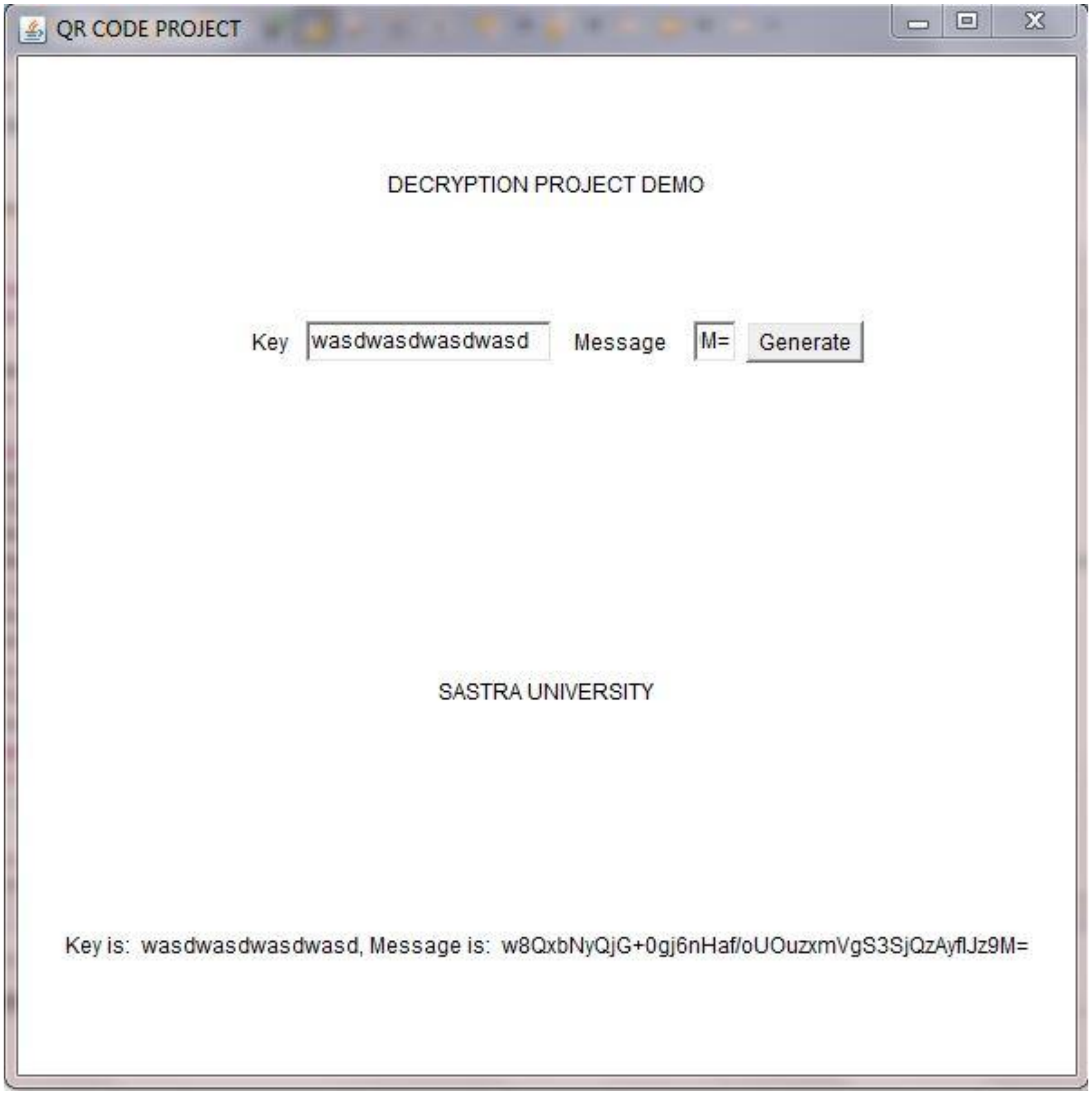


Fig.16. Retrieved QR code

6.7 DECRYPTION:



DECRYPTION PROJECT DEMO

Key: wasdwasdwasdwasd Message: M=

SASTRA UNIVERSITY

```
Key is:  wasdwasdwasdwasd, Message is:  w8QxbNyQjG+0gj6nHaf/oUOuzxmVgS3SjQzAyflJz9M=
```

Fig.17. AES-128 BIT DECRYPTION

CHAPTER 7

CONCLUSION

The proposed methodology presents a combination of an efficient encrypting algorithm and a steganographic technique to communicate the confidential information in a safe and secure manner making it extremely hard to decode. It includes an efficient encoding technique in encrypting a secret message before encoding it in to a QR code. This encoded image is scrambled to achieve another security level. The scrambled QR code is finally embedded in a suitable cover image. The cover image is then transferred securely for exchanging secret information, from which the information is retrieved through the decoding process at the receiver’s side. This methodology provides a four level security for the secret message to be transferred. It includes the combination of cryptography and steganography to achieve efficient results. The use of scrambling technique further distorts the QR code making it reliable for encoding important information. Scrambling protects it from any unauthorized scans. Finally, Embedding yields a standard stego image. This technique can find application in communicating confidential information in banking, defense, educational, e-Business sectors. The technique can be further improved to achieve required level of security by adopting various necessary techniques to suit the requirements.

CHAPTER 8

BIBLOGRAPHY

[1] B. Padmavathi, S. Ranjitha Kumari," A Survey on Performance Analysis of DES, AES and RSA Algorithm along with LSB Substitution", International Journal of Science and Research (IJSR), India Online ISSN: 2319-7064.

[2] Ji-Hong Chen, Wen-Yuan Chen and Chin-Hsing Chen,"Identification Recovery Scheme using Quick Response(QR) Code and Watermarking Technique", Applied Mathematics and Information Sciences, An International Journal, 8, No. 2, 585-596 (2014).

[3] Kaustubh Choudhary," Properties of Images in LSB Plane", IOSR Journal of Computer Engineering (IOSRJCE), 2278-0661 Volume 3, Issue 5 (July-Aug. 2012), PP 08-16

[4] Peter Kieseberg, Manuel Leithner, Martin Mulazzani, Lindsay Munroe, Sebastian Schrittwieser, Mayank Sinha, Edgar Weippl,"QR Code Security",2014.

[5] Shashi Mehrotra Seth, Rajan Mishra," Comparative Analysis Of Encryption Algorithms For Data Communication",IJCST Vol 2,Issue 2,June 2011.

[6] Somdip Dey, Kalyan Mandal, Joyshree Nath,Asoke Nath,"Advanced Steganography Algorithm Using Randomized Intermediate QR Host Embedded With Any Encrypted Secret Message: ASA_QR Algorithm",I.J.Modern Education and Computer Science, 2012, 6, 59-67 .

[7] S. Uma Maheswari, D. Jude Hemanth," Frequency domain QR code based image steganography using Fresnelet transform", Int. J. Electron. Commun. (AEÜ) 69 (2015) 539–544.

[8] Wen-Yuan Chen, Jing-Wein Wang,"Nested Image steganography scheme using QR-barcode technique",Society of Photo-Optical Instrumentation Engineers 485, 057004 May 2009.1-10.

[9] Wai Wai Zin," Message Embedding In PNG File Using LSB Steganographic Technique", International Journal of Science and Research (IJSR), India Online ISSN: 2319-7064.

[10] Yin-Jen Chiang, Pei-Yu Lin, Ran-Zan Wang, Yi-Hui Chen,” Blind QR Code Steganographic Approach Based upon Error Correction Capability”, KSII Transactions on Internet and Information Systems vol. 7, no. 10, Oct. 2013.