

B.M.S College of Engineering
P.O. Box No.: 1908 Bull Temple Road,
Bangalore-560 019

Department of INFORMATION Science & Engineering



SEND SECURE

Report on Computer Network Project

Submitted To - Soumya K S

Submitted By –

Group 2

Nyamagouda Sagar

1BM14IS059

Sameer R Katti

1BM14IS079

Nagendra J

1BM15IS410

Problem Statement:

In the fast developing world, the problem of data protection is ever increasing. The data must be accessed by only those who are authorized to do so. A platform where people can share messages between devices connected to same network. The objective is to develop a P2P protocol that can work in both mobile and fixed networks. It must provide end-to-end encryption, so that while sharing the information the 3rd party or the hacker won't be able to decipher the information .

Abstract:

This application is designed for the use of the people that know each other through IP addresses. The system is basically decentralized - there are no dedicated servers. The application is designed with privacy considerations in mind – nobody other than connected people are supposed to get any information about the information shared. The application provides end-to-end encryption ,where the message is embedded into an image and image is sent to receiver across networks and then embedded message is obtained and decrypted to view the actual message sent.

Techniques used for end-to-end encryption are as follows:

1. RSA or AES for Encryption/Decryption
2. Image Steganography

Objective:

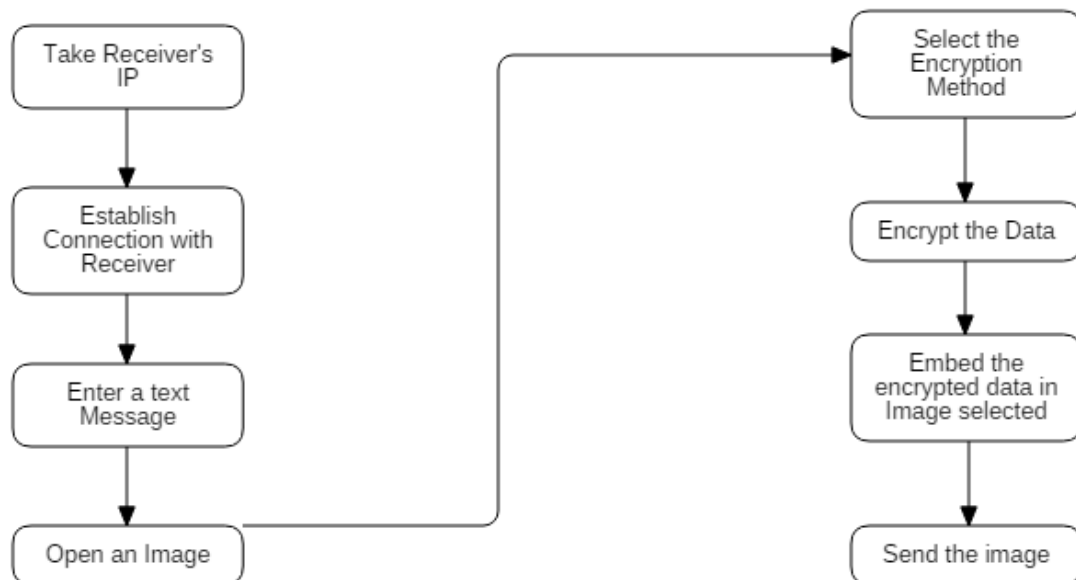
To create an application where in messages are encrypted and embedded in image which is sent to receiver on the same network and then decrypted to show actual message.

System Requirements:

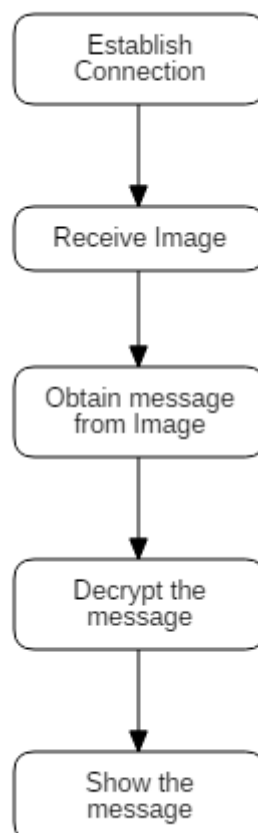
1. Operating System: Windows 7 or above
2. Front End: Java Swing using Netbeans IDE (Version 9.2)
3. RAM: 2GB
4. JAVA SDK and JRE should be pre-installed.
5. Wifi Connectivity.

Flowchart:

1. For the Sender

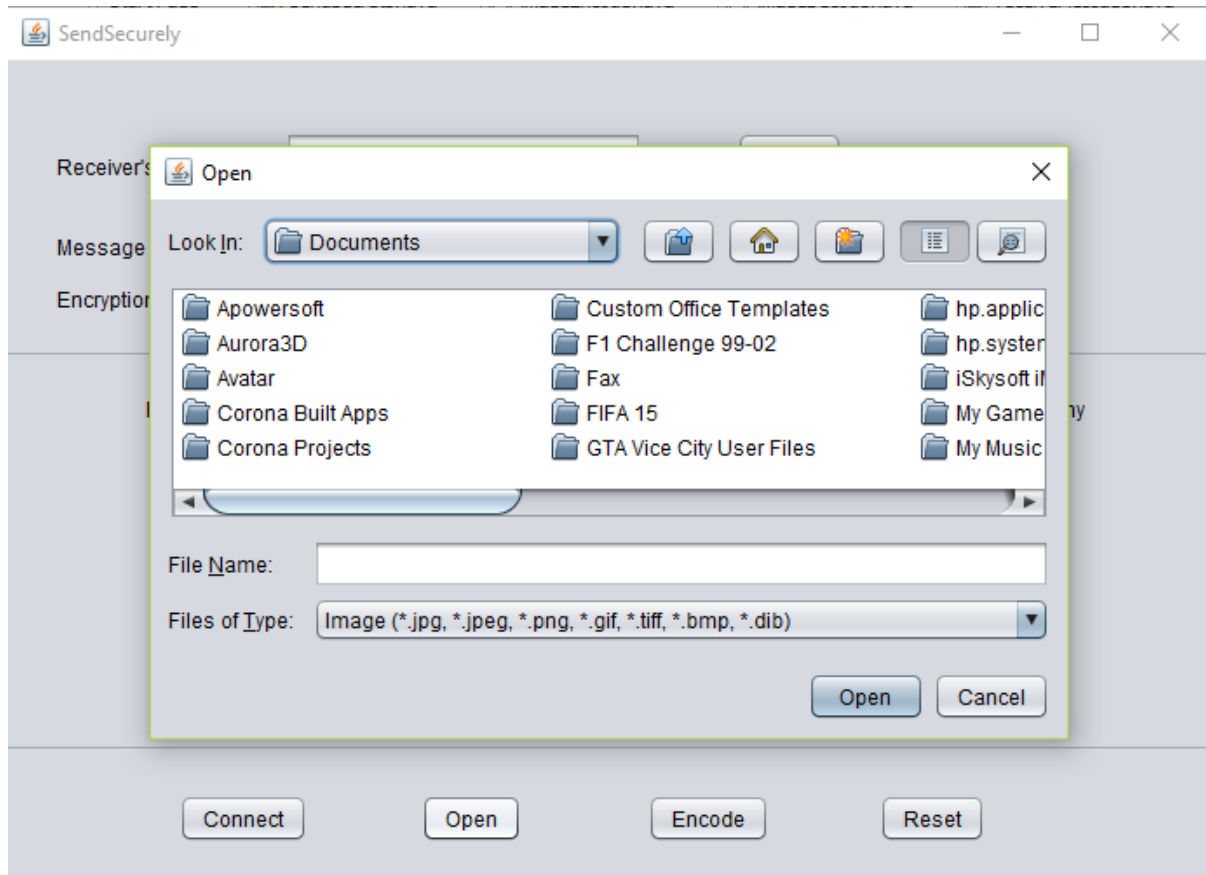


2. For the Receiver



Source Code and Output:

1. Open an Image



Code:

```
JFileChooser fc = new JFileChooser("Open an image");

javax.swing.filechooser.FileFilter ff = new javax.swing.filechooser.FileFilter() {

    public boolean accept(java.io.File f) {

        String name = f.getName().toLowerCase();

        if(open)

            return f.isDirectory() || name.endsWith(".jpg") || name.endsWith(".jpeg") ||

                name.endsWith(".png") || name.endsWith(".gif") || name.endsWith(".tiff") ||

                name.endsWith(".bmp") || name.endsWith(".dib");

        return f.isDirectory() || name.endsWith(".png") || name.endsWith(".bmp");

    }

    public String getDescription() {
```

```

        if(open)
            return "Image (*.jpg, *.jpeg, *.png, *.gif, *.tiff, *.bmp, *.dib)";
        return "Image (*.png, *.bmp)";
    }
};

fc.setAcceptAllFileFilterUsed(false);
fc.addChoosableFileFilter(ff);
java.io.File f = null;
if(open && fc.showOpenDialog(this) == fc.APPROVE_OPTION)
    f = fc.getSelectedFile();
else if(!open && fc.showSaveDialog(this) == fc.APPROVE_OPTION)
    f = fc.getSelectedFile();
return f;

```

2. Show the IPs connected



Code:

```

String subnet="192.168";
int timeout = 100;
for(int j=2;j<255;j++){
    String host1=subnet+"."+j;
    for (int i = 2; i < 255&&this.isDisplayable(); i++)
    {

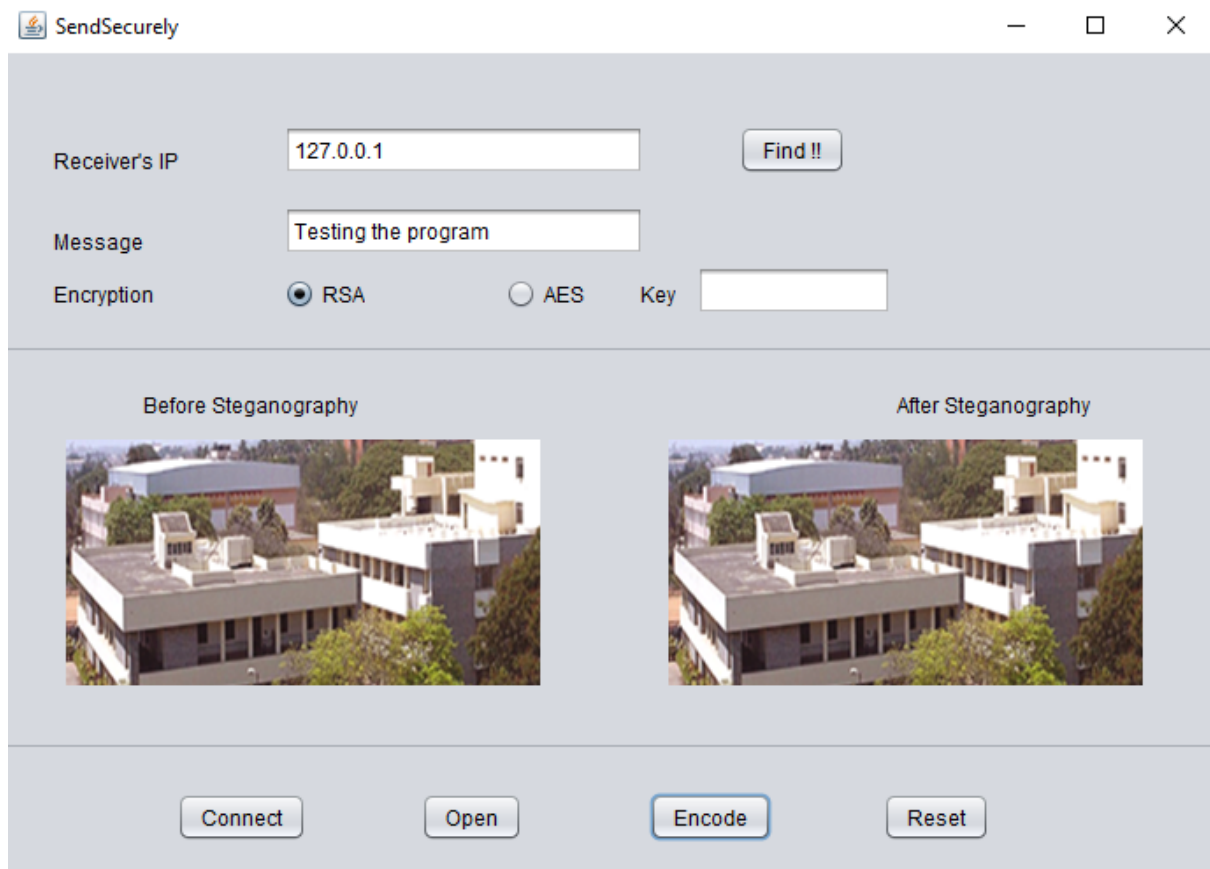
```

```

String host = host1 + "." + i;
if (InetAddress.getByName(host).isReachable(timeout))
{
    System.out.println(host + " is reachable");
    InetAddress addr = InetAddress.getByName(host);
    String hostname = addr.getCanonicalHostName();
    jTextArea1.append("\n" + host + "<---->" + hostname);
    this.validate();
}
}
}

```

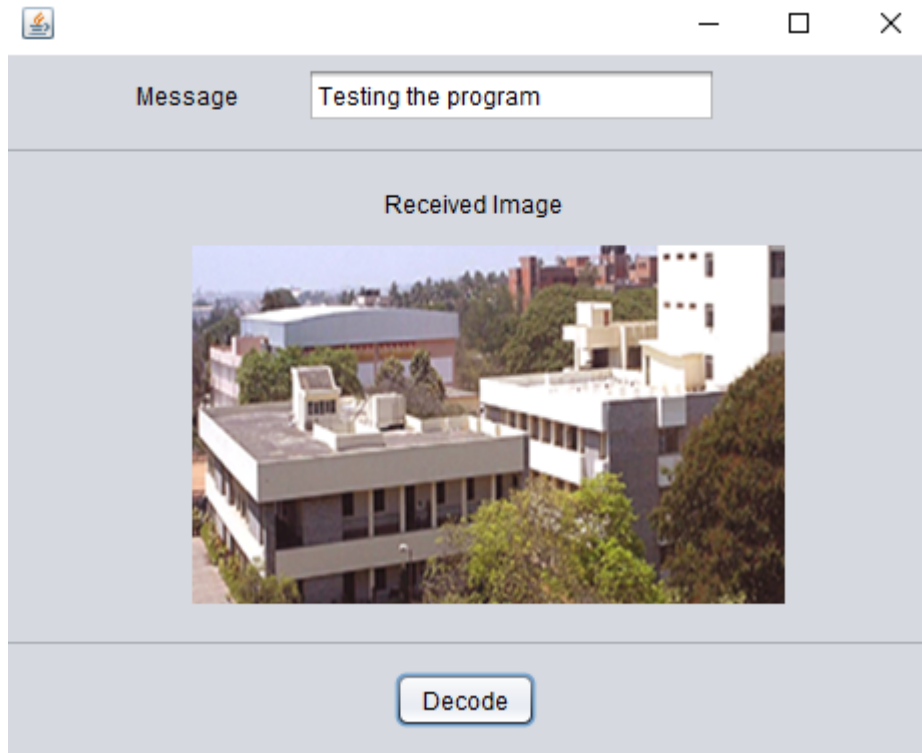
3. Main Window:



The screenshot shows the 'SendSecurely' application window. The interface includes the following elements:

- Receiver's IP:** A text field containing '127.0.0.1' and a 'Find !!' button.
- Message:** A text field containing 'Testing the program'.
- Encryption:** Two radio buttons, 'RSA' (selected) and 'AES', followed by a 'Key' text field.
- Before Steganography:** A placeholder image showing a building.
- After Steganography:** A placeholder image showing the same building, representing the result of the steganography process.
- Buttons:** 'Connect', 'Open', 'Encode' (highlighted with a blue border), and 'Reset'.

4. Receiver Window:



5. Image Steganography Code:

```
int i=0,j=0,green;
String message=encryptData();
if(message.equals(""))
    return;
byte b[] = message.getBytes();
String[] val=new String[b.length];
String bitMsg="";
for(i=0; i<b.length; i++){
    val[i]=Integer.toBinaryString(b[i]);
    val[i]=convertTo8(val[i]);
    bitMsg+=val[i];
}
if(bitMsg.length()>(image.getHeight()*image.getWidth()+1)
{
```

```

        JOptionPane.showMessageDialog(this, "Too long message!",
        "", JOptionPane.ERROR_MESSAGE);
    return;
    }
    img=image;
    int curPos=0;
    int red,blue,alpha=255;
    String col="";
    int argb,n ;
    Color c;
    for(i=0;i<image.getHeight()&&curPos<bitMsg.length();i++)
    {
        for(j=0;j<image.getWidth()&&curPos<bitMsg.length();j++)
        {

            c=new Color(image.getRGB(j, i));
            green=c.getGreen();
            col=convertTo8(Integer.toBinaryString(green));

            col=assign(col,bitMsg.charAt(curPos++));

            red=c.getRed();
            if(red%2==1)
            {
                red-=1;
            }
            blue=c.getBlue();
            if(blue%2==1)
            {
                blue-=1;
            }
            Color colo=new Color(red,Integer.parseInt(col, 2),blue);
            image.setRGB( j, i, colo.getRGB());
        }
    }

```



```

    }
    c=new Color(image.getRGB(j,i));
    green=c.getGreen();

    red=c.getRed();
    if(red%2==0)
    {
        red+=1;
    }
    blue=c.getBlue();
    if(blue%2==0)
    {
        blue+=1;
    }
    Color colo=new Color(red,Integer.parseInt(col, 2),blue);
    image.setRGB( j, i, colo.getRGB());

```

6. Image Steganography (To obtain message from image):

```

int i=0,j=0,green;
    String bitMsg="";
    int red,blue,alpha=255;
    String col="";
    int bitPos=7 ;
    Color c = new Color(image.getRGB(j, i));
    for(i=0;i<image.getHeight();i++)
    {
        for(j=0;j<image.getWidth();j++)
        {
            c=new Color(image.getRGB(j, i));
            green=c.getGreen();
            red=c.getRed();
            blue=c.getBlue();
            if(red%2==0&&blue%2==0){
                col=convertTo8(Integer.toBinaryString(green));

```

```

        bitMsg+=col.charAt(7);
    }
    else if(red%2!=0||blue%2!=0)
    {
        i=image.getHeight();
        j=image.getWidth();
    }
}
}
this.getMessage(bitMsg);

```

7. RSA Encryption:

```

    public String RSAEncrypt(String message,BigInteger E,BigInteger n) throws IOException
    {
        byte[] encrypted = this.encrypt(message.getBytes(),E,n);
        // decrypt
        return bytesToString(encrypted);
    }
    public String RSADecrypt(String message,BigInteger D,BigInteger n) throws IOException
    {
        byte[] decrypted = this.decrypt(StringToBytes(message),D,n);
        return finalString(decrypted);
    }
    public String finalString(byte[] decrypted)
    {
        String s=new String(decrypted);
        return s;
    }
    private static byte[] StringToBytes(String encrypted)
    {
        String[] parts = encrypted.split("\n");
        byte[] h=new byte[parts.length];
        for(int i=0;i<parts.length;i++){
            h[i]=Byte.valueOf(parts[i]);
        }
    }

```

```

    }
    return h;
}

private static String bytesToString(byte[] encrypted)
{
    String test = "";
    for (byte b : encrypted)
    {
        test += Byte.toString(b);
        test+="\n";
    }
    return test;
}

// Encrypt message
public byte[] encrypt(byte[] message,BigInteger E, BigInteger n)
{
    return (new BigInteger(message)).modPow(E, n).toByteArray();
}

// Decrypt message
public byte[] decrypt(byte[] message,BigInteger D, BigInteger n)
{
    return (new BigInteger(message)).modPow(D, n).toByteArray();
}

```

8. AES Encryption:

```

public class AES {
    static String IV = "AAAAAAAAAAAAAAAA";

    public static byte[] encrypt(String plainText, String encryptionKey) throws Exception {
        if(plainText.length()%16!=0)
        {

```

```

        for(int i=0;i<plainText.length()%16;i++)
            plainText+="\0";
    }
    Cipher cipher = Cipher.getInstance("AES/CBC/NoPadding", "SunJCE");
    SecretKeySpec key = new SecretKeySpec(encryptionKey.getBytes("UTF-8"), "AES");
    cipher.init(Cipher.ENCRYPT_MODE, key,new IvParameterSpec(IV.getBytes("UTF-8")));
    return cipher.doFinal(plainText.getBytes("UTF-8"));
}

    public static byte[] StringToBytes(String encrypted)
    {
        String[] parts = encrypted.split("\n");
        byte[] h=new byte[parts.length];
        for(int i=0;i<parts.length;i++){
            h[i]=Byte.valueOf(parts[i]);
        }
        return h;
    }

    public static String bytesToString(byte[] encrypted)
    {
        String test = "";
        for (byte b : encrypted)
        {
            test += Byte.toString(b);
            test+="\n";  }
        return test;
    }

    public static String decrypt(byte[] cipherText, String encryptionKey) throws Exception{
        Cipher cipher = Cipher.getInstance("AES/CBC/NoPadding", "SunJCE");
        SecretKeySpec key = new SecretKeySpec(encryptionKey.getBytes("UTF-8"), "AES");
        cipher.init(Cipher.DECRYPT_MODE, key,new IvParameterSpec(IV.getBytes("UTF-8")));
        return new String(cipher.doFinal(cipherText),"UTF-8");
    } }

```

Conclusion:

The system provides a easy way of sending the messages securely while maintaining the user friendly characteristics. Since the image steganography is used, the very existence of data can be hided. And to add to that the successful encryption algorithms such as RSA and AES are used to encrypt and decrypt messages before sending. So, it becomes highly difficult to obtain the message sent.

References:

1. <https://en.wikipedia.org/wiki/Steganography> - Wikipedia
2. [https://en.wikipedia.org/wiki/RSA_\(cryptosystem\)](https://en.wikipedia.org/wiki/RSA_(cryptosystem)) - RSA Algorithm
3. <https://www.youtube.com/watch?v=5opGM7jXvHM&index=2&list=WL&t=6s> – Image Steganography