

# IS Assignment

## Image Encryption Using AES Algorithm

Submitted By:

SAURABH KISHOR

Reg No: 19CS1100

Dept: CSE-2

# Image Encryption Using AES Algorithm

**AES Image encryption:** It is a technique that convert original image to another form that is difficult to understand. No one can access the content without knowing a decryption key. Image encryption has applications in corporate world, health care, military operations(screate missile dealing), and multimedia systems.

## **AES (Advanced Encryption Standard):**

A more secure encryption algorithm. which is a symmetric encryption algorithm.

The Advanced Encryption Standard (AES) algorithm is a symmetric block cipher that processes image which is of blocks size 128 bits using three different cipher key size of lengths 128,192 or 256 bits.

It uses AES Key Expansion in which the encryption process is a bit wise exclusive or operation of a set of image pixels along with the a 128 bit key which changes for every set of pixels.

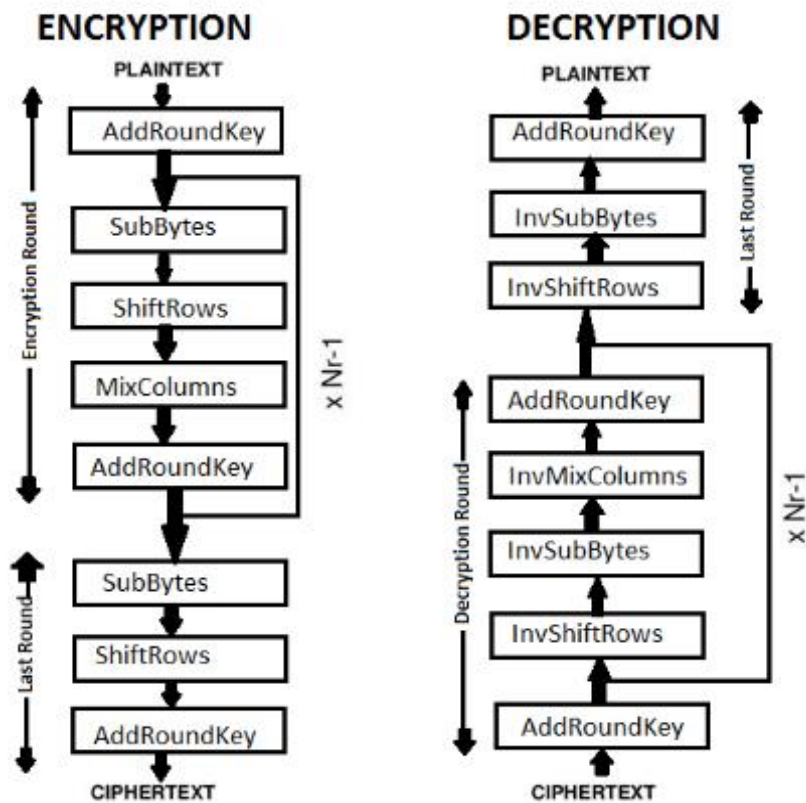


## 1. Rounds

1. Sub Bytes—a non-linear substitution step where each byte is replaced with another according to a lookup 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

## 2. Final Round (no Mix Columns)

1. Sub Bytes
2. Shift Rows
3. Add Round Key.



### AES Image Encryption:

Conversion of original image i.e plain image into encrypted image i.e cipher image is known as image encryption.

The round consists of the following stages for image encryption shown in above picture:

- SubstituteBytes
- ShiftRow
- MixColumns
- AddRoundKey

- **SubstituteBytes:**

The SubBytes transformation includes non-linear byte substitution, operating on each of the state bytes independently. This is done by using a once-precalculated substitution table called S-box. S-box table contains 256 numbers (from 0 to 255) and their corresponding resulting values.

- **ShiftRow:**

ShiftRows transformation includes, the rows of the state are cyclically left shifted. Row 0 remain unchanged; row 1 does shift of one byte to the left; row 2 does shift of two bytes to the left and row 3 does shift of three bytes to the left.

- **InverseShiftRow:**

InvShiftRows exactly functions the same as ShiftRows, only in the opposite direction. The first row is not shifted, while the second, third and fourth rows are shifted right by one, two and three bytes respectively.

Algorithm	Key length	Block size	Number of round
ASE-128	4	4	10
ASE-196	6	4	12
ASE-256	8	4	14

**InverseSubstituteByte:** The InvSubBytes transformation is done using a onceprecalculated substitution table called InvS-box. That InvSbox table contains 256 numbers (from 0 to 255) and their corresponding values. InverseMixColumns: In the InvMixColumns transformation, the polynomials of degree less than 4 over GF(28), which coefficients are the elements in the columns of the state, are multiplied modulo  $(x^4 + 1)$  by a fixed polynomial  $d(x) = \{0B\}x^3 + \{0D\}x^2 + \{09\}x + \{0E\}$ , where  $\{0B\}$ ,  $\{0D\}$ ;  $\{09\}$ ,  $\{0E\}$  denote hexadecimal values.

## **CONCLUSION:**

In this, image encryption and decryption using Advanced Encryption Standard (AES) algorithm is proposed for image encryption and decryption that can process with the data block of 128 bit and cipher key length of 256 bit. The usage of 256 bit cipher key to achieve the high security, because 256 bit cipher key is difficult to break. As a result of this secure transmission of image can be possible. Future scope is, it can be used in various applications like Military communication, Forensics, Intelligent systems etc.

### **Advantages:**

- The image can only be viewed by the receiver as the image is encrypted using AES and the key is only known to the sender and receiver.
- Since the image is encrypted using AES, it is more secure than the DES and triple DES.
- AES allows you to choose a 128-bit, 192-bit or 256-bit key, making it exponentially stronger than the 56-bit key of DES.
- Since the key size is 192 bits, it makes the encryption and decryption more secure.

### **Disadvantages**

- The file size to be transmitted becomes large since it contains encrypted data.
- Since the file size is huge it can be suspected to contain some critical information.

## Code:

```
package imageencryption.java;

/**
 * @author SAURABH KISHOR
 */
import java.awt.Font;
import javax.swing.*;
import java.awt.*;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
public class ImageEncryptionJava {

    /**
     * @param args the command line arguments
     */

    public static void encryption(int key){
        JFileChooser fileChooser = new JFileChooser();
        fileChooser.showOpenDialog(null);
        File file = fileChooser.getSelectedFile();

        try{
            FileInputStream fis = new FileInputStream(file);
            byte [] data = new byte[fis.available()];
```



```
    fis.read(data);
    int i = 0;
    for(byte b:data){
        System.out.println(b);
        data[i] = (byte) (b^key);
        i++;
    }
```

```
        FileOutputStream fos = new
FileOutputStream(file);
        fos.write(data);
        fos.close();
        fis.close();
        JOptionPane.showMessageDialog(null,"Image
Encrypted done.");
    }
    catch(Exception e){
        e.printStackTrace();
    }
}
```

//decryption

```
public static void decryption(int key){
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.showOpenDialog(null);
    File file = fileChooser.getSelectedFile();

    try{
        FileInputStream fis = new FileInputStream(file);
```

```

byte [] data = new byte[fis.available()];

fis.read(data);
int i = 0;
for(byte b:data){
    System.out.println(b);
    data[i] = (byte) (b^key);
    i++;
}

FileOutputStream fos = new
FileOutputStream(file);
    fos.write(data);
    fos.close();
    fis.close();
    // JOptionPane.showMessageDialog(null,"Image
Encrypted done.");
}
catch(Exception e){
    e.printStackTrace();
}
}

```

```

public static void main(String[] args) {
    // TODO code application logic here
    //System.out.println("Test");
    JFrame f = new JFrame();
    f.setTitle("Image Encryption");
    f.setSize(500,500);
    f.setLocationRelativeTo(null);
}

```

```
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
Font font = new Font("Roboto",Font.BOLD,26);
```

```
label
```

```
JLabel l1 = new JLabel();
```

```
l1=new JLabel("Image Encryption & Decryption.");
```

```
l1.setFont(font);
```

```
JButton button = new JButton();
```

```
button.setBounds(50,100, 200,30);
```

```
button.setText("Open Image");
```

```
button.setText("Encryption");
```

```
button.setFont(font);
```

```
JTextField textField = new JTextField(10);
```

```
textField.setFont(font);
```

```
f.setLayout(new FlowLayout());
```

```
button.addActionListener(e->{
```

```
System.out.println("button clicked");
```

```
String text = textField.getText();
```

```
int temp = Integer.parseInt(text);
```

```
encryption(temp);});
```

```
//decryption
```

```
System.out.println("");
```

```
JButton button2 = new JButton();
```

```
button.setText("Open Image");
```

```

        button2.setText("Decryption");
        button2.setBounds(50,100, 200,30);
        button2.setFont(font);

        JOptionPane.showMessageDialog(null,"Image
Encrypted.");
        JTextField textField2 = new JTextField(10);
        textField2.setFont(font);
        f.setLayout(new FlowLayout());

        button2.addActionListener(e->{
            System.out.println("Decryption button clicked");
            String text2 = textField2.getText();
            int temp2 = Integer.parseInt(text2);
            decryption(temp2);
            JOptionPane.showMessageDialog(null,"Now Image
Decrypted.");

        });
        f.add(button);
        f.add(textField);
        System.out.println("");
        System.out.println("");
        f.add(button2);

        f.add(textField2);

        f.setVisible(true);

    }
}

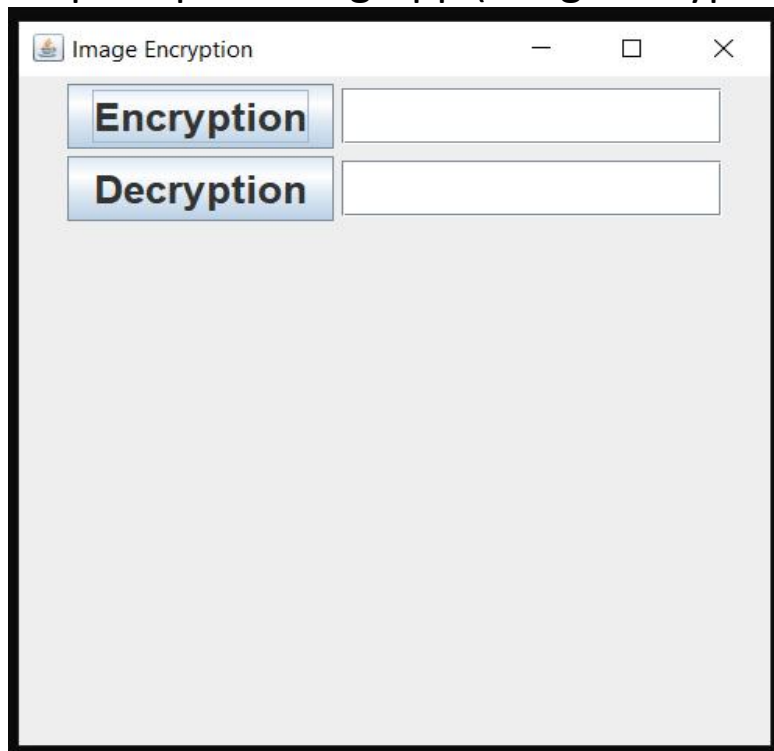
```

## Demo:

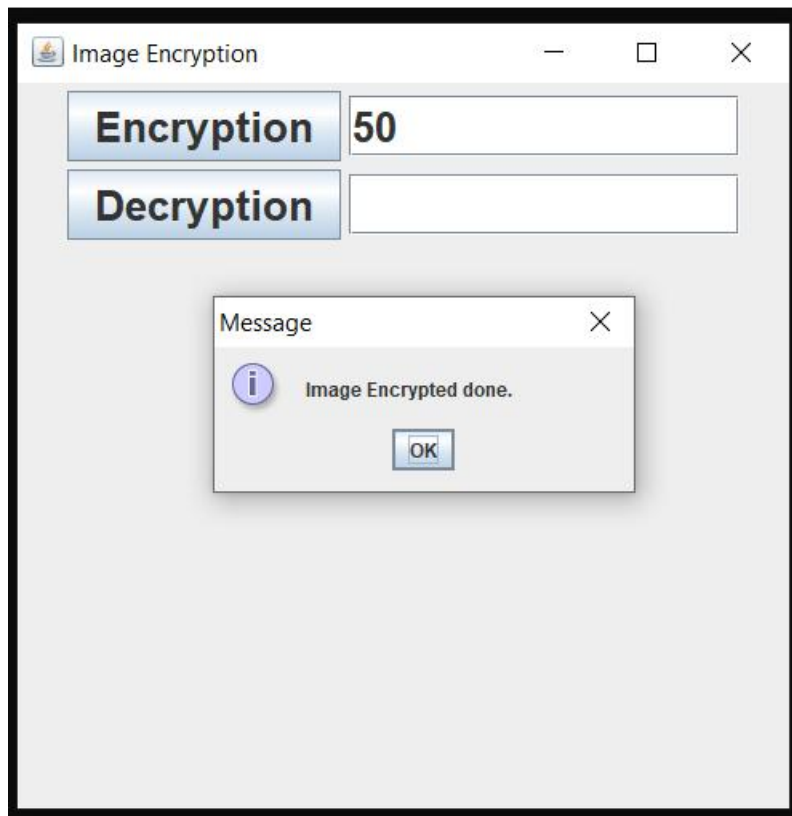
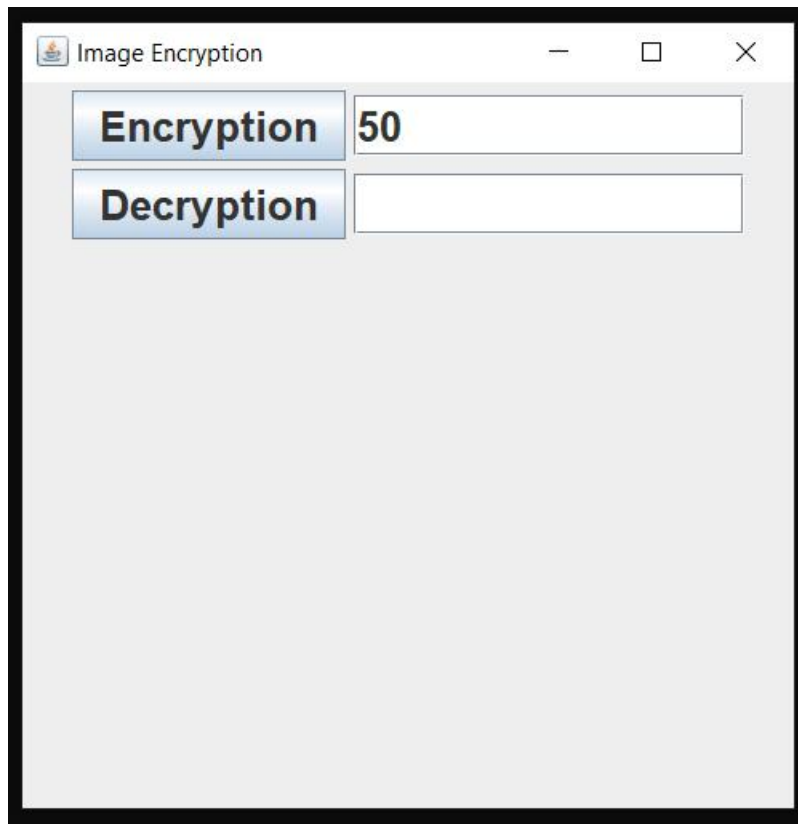
Steps 1: Select image, which I have to do Encryption.



Step2: Open swing App (Image Encryption APP)



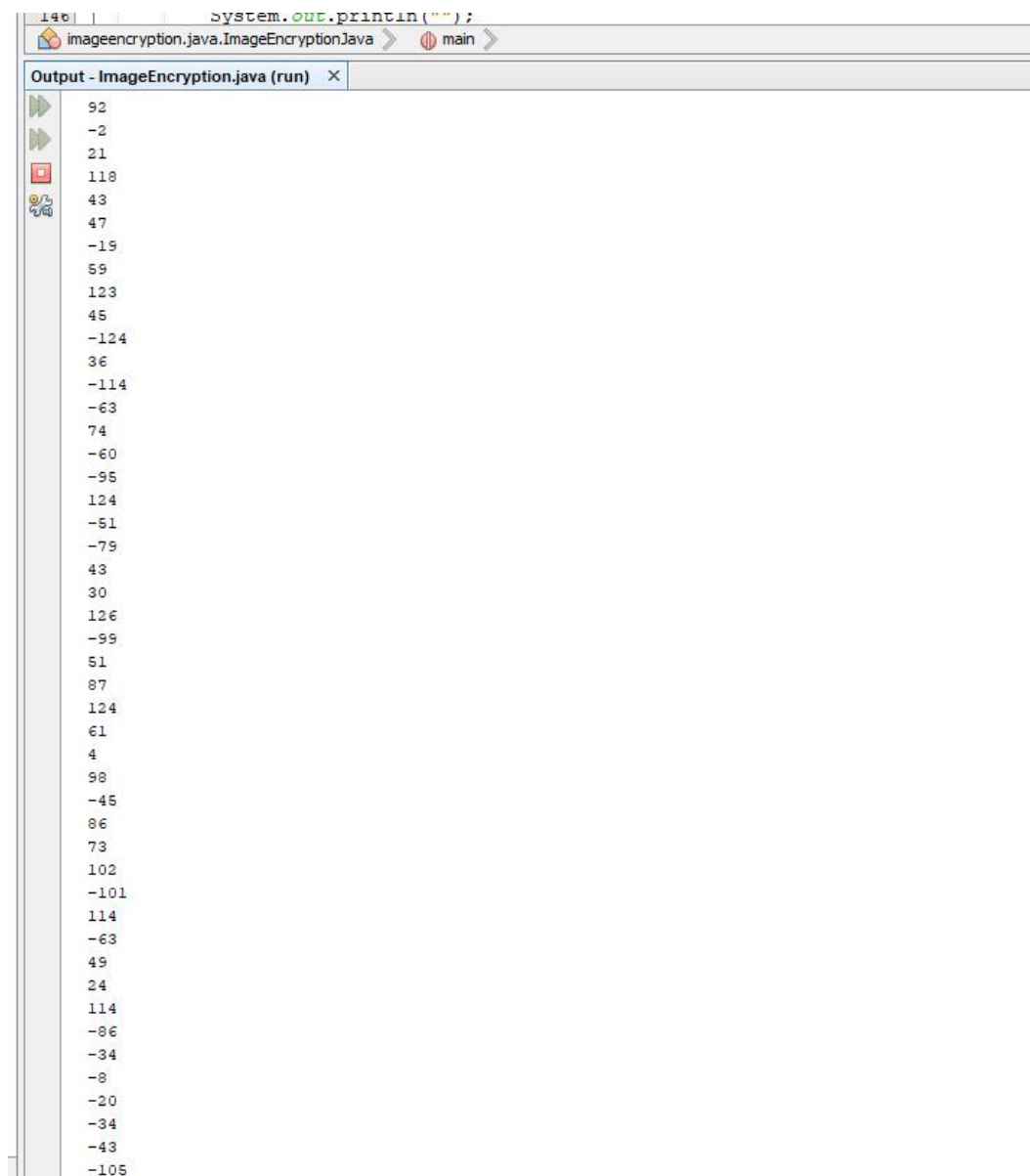
Step3: Encrypt the Image using screate Key. And select image, then encrypt it.



a.jpeg

It appears that we don't support this file format.

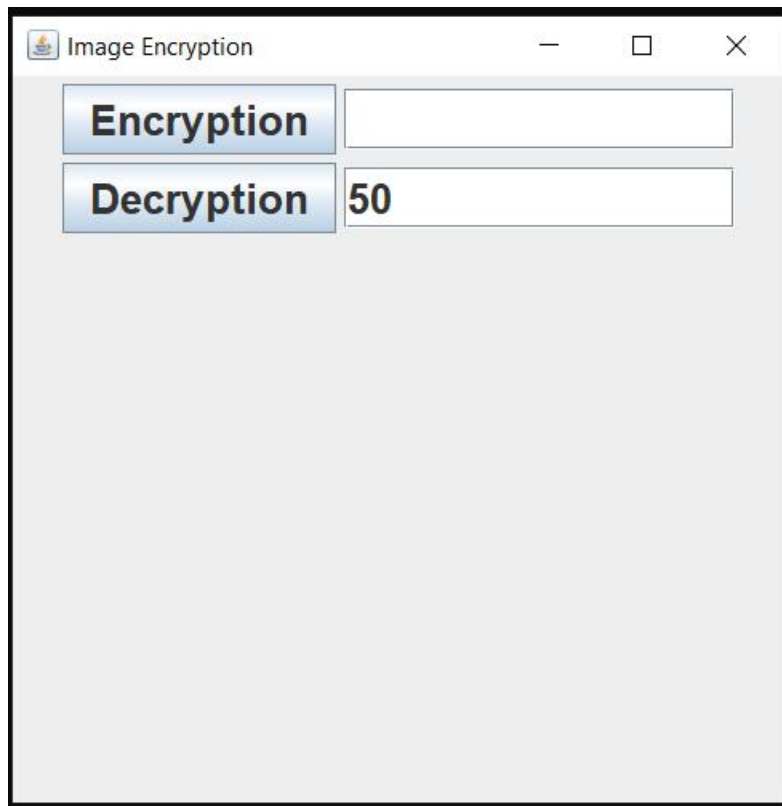
Step4: Encrypted code change into another formate.



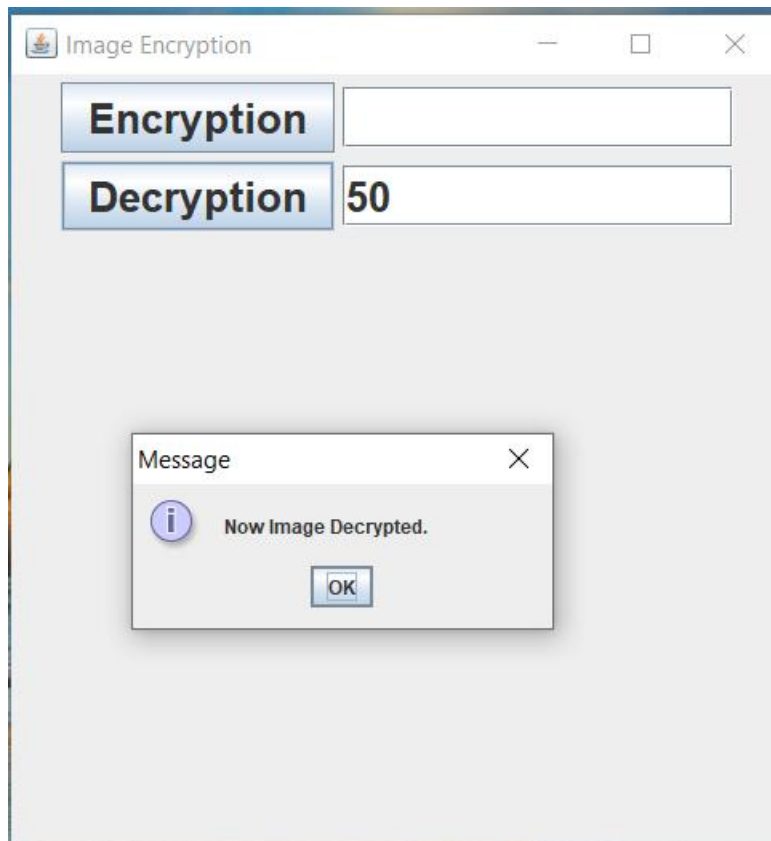
The screenshot shows an IDE window with a tab titled "imageencryption.java.ImageEncryptionJava" and a sub-tab "main". Below the tabs is a panel titled "Output - ImageEncryption.java (run)". The panel displays a list of integers, each preceded by a green arrow icon. The integers are: 92, -2, 21, 118, 43, 47, -19, 59, 123, 45, -124, 36, -114, -63, 74, -60, -95, 124, -51, -79, 43, 30, 126, -99, 51, 87, 124, 61, 4, 98, -45, 86, 73, 102, -101, 114, -63, 49, 24, 114, -86, -34, -8, -20, -34, -43, -105.

```
146 | System.out.println("");  
imageencryption.java.ImageEncryptionJava > main >  
Output - ImageEncryption.java (run) X  
92  
-2  
21  
118  
43  
47  
-19  
59  
123  
45  
-124  
36  
-114  
-63  
74  
-60  
-95  
124  
-51  
-79  
43  
30  
126  
-99  
51  
87  
124  
61  
4  
98  
-45  
86  
73  
102  
-101  
114  
-63  
49  
24  
114  
-86  
-34  
-8  
-20  
-34  
-43  
-105
```

Step5: After Encryption, need to decrypt the image.







Step6: Then our image will be , vissible again.

## How it Works (Mechanism):

```
C:\>jshell
| Welcome to JShell -- Version 16.0.1
| For an introduction type: /help intro

jshell>

jshell> int a = 15
a ==> 15

jshell> int key = 10
key ==> 10

jshell> int r = a^key
r ==> 5

jshell> r^a
$4 ==> 10

jshell>
```

Taking a simple example,  
Let take a variable(a) value = 15  
Then, take key value = 10

Using Relation operator I.e 'XOR' to calculate it's xor value. Which will change it value. And store that value in new variable®(r) due to which, image form will be change and our image will be encrypted.

Then, again use 'xor' operator, which will return our original value.I.e, our image will be decrypted.

*Thank  
you*

