UNIVERSITY OF MUMBAI

A DISSERTATION REPORT ON

"Encryption and Decryption Using AES Algorithm"

SUBMITTED IN PARTIAL FULFILMENT FOR

THE REQUIREMENTS OF THE DEGREE

BACHELOR OF ENGINEERING

IN

COMPUTER ENGINEERING

GROUP MEMEBERS

Rajshree Borkar (03)

Ashish Gavade (08)

Shubham Khedeker (21)

SK Mohd Amjad Raza (49)

UNDER THE GUIDANCE OF

Rajshri Chaudhari

DEPARTMENT OF COMPUTER ENGINEERING

G.V. ACHARYA INSTITUTE OF ENGINEERING AND TECHNOLOGY

UNIVERSITY OF MUMBAI

2020-2021

DECLARATION

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Date:

Place: SHELU

PROJECT REPORT APPROVAL FOR S.E.

This project report entitled **"Encryption and Decryption Using AES Algorithm"** by **"Rajshree Borkar (03)", "Ashish Gavade (08)", "Shubham Khedeker (21)", "SK Mohd Amjad Raza (49)"** is approved for the degree of **"Second Year of Computer Engineering".**

Examiners

1._____

2. _____

Date:

Place: SHELU

CERTIFICATE

This is to certify that the project entitled **"Encryption and Decryption Using AES Algorithm"** is a bonafide work of **"Rajshree Borkar (03)", "Ashish Gavade (08)", "Shubham Khedeker (21)", "SK Mohd Amjad Raza (49)",** "submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **"Undergraduate"** in **"Second Year of Computer Engineering".**

Prof. Rajshri Chaudhari.	Prof. Roshan Bauskar.
Project Guide	Head of Department
Department of Computer Engineering	Computer Engineering
G.V.A.I.E.T	G.V.A.I.E.T

Prof. Rajshri Chaudhari.	Dr. Prashant Sonare.
Project Co-Ordinator	Principle
Department of Computer Engineering	G.V.A.I.E.T
G.V.A.I.E.T	

ACKNOWLEDGEMENT

We have immense pleasure in presenting the report for our project entitled “Encryption and Decryption Using AES Algorithm”.

We would like to take this opportunity to express our gratitude to a number of people who have been sources of help & encouragement during the courses of this project.

We are very graceful and indebted to our project guide & our respected HOD for providing their enduring patience, guidance, & suggestions. They were the one who never let our morale down & always supported us through our thick & thin. They were the constant source of inspiration for us & took at most interest in our project.

We would also like to thank all the staff members for their invaluable co-operation & permitting us to work in the computer lab.

We are also thankful to all the students for giving us their useful advice & immense co-operation. Their support made the working of this project very pleasant.

Group Members

Rajshree Borkar (03)

Ashish Gavade (08)

Shubham Khedekar (21)

SK Mohd Amjad Raza Mohd Irfan (49)

TABLE OF CONTENT

ABSTRACT

Data security is a major issue which we are facing today in this digital world of communication. As we know that today hackers are almost at every corner in search of our useful data which can be hacked by them for their different purposes. Even the risk gets doubled when come to the data of any country's government. So, a system or terminology is must require to make that data safe forever by any means during communication.

Data protection can be accomplished by changing the original data by any means to some other un useful data so that if someone gets that data then also it must remain in un useful bits. This process can be achieved by Encrypting that data by some means of algorithms which are known to the sender and the similar Decryption algorithms to be known to only the desired receiver so that it can convert that encrypted data back to the user understandable form. Today as it is a need to develop such kind of applications which performs the specified task but along with it should be very much user friendly so that no special skills need to be required to learn in order to use that application or project.

In the application the user has to select either wants to send something by encrypting or wants to receive by decrypting. If it wants to send then it has to select source file previously designed or type some message which is to Encrypt and then transfer. Whereas on the receiver side again the receiver has to select the file which is to be received from the sender along with a decryption key to decrypt that message. Decryption key can be selected either manually if told by the sender or selecting key sent by the sender along with the encrypted data to avoid further delay in processing the message.

INTRODUCTION

1.1 Overview:

This project is all about the Data Security. How we can secure our data from hackers. The symmetric-key block cipher plays an important role in data encryption. It means that the same key is used for both encryption and decryption. The Advanced Encryption Standard (AES) is a widely used symmetric-key encryption algorithm.

In this project, we'll see how to implement AES encryption and decryption using the Java Cryptography Architecture (JCA) within the JDK.

1.2 Aim and Objective:

As we know that today hackers are almost at every corner in search of our useful data which can be hacked by them for their different purposes. Even the risk gets doubled when come to the data of any country's government. So, a system or terminology is must require to make that data safe forever by any means during communication. So the main aim of the project is to secure our important data from hackers and keep it safe.

1.3 Problem Statement:

Many organizations today are facing difficulties in choosing the best encryption and decryption algorithms for incorporating data security, confidentiality, and integrity in their businesses. Organizations need huge budget for analyzing the encryption processing overhead of a variety of cryptographic algorithms that are available in the market today

1.4 Features of AES Algorithm

- Block encryption implementation.
- 128-bit group encryption with 128, 192 and 256-bit key lengths.
- Symmetric algorithm requiring only one encryption and decryption key.
- Data security for 20-30 years.
- Worldwide access.
- Easy overall implementation.

1.5 System Requirements:

- Compatible with windows
- Latest version of JDK (Java Development Kit)
- Terminal (Apache NetBeans 12.0)
- RAM Minimum 4G

IMPLEMENTED SYSTEM

2.1 Introduction:

Encryption:

Encryption is a process used to protect data stored in the cloud from unauthorized users. In other words, the primary purpose of encryption is confidentiality. This technique is advancing day by day. The modern encryption along with confidentiality, they provide key elements of security. They are listed below:

1. Authentication
2. Integrity
3. Non- repudiation

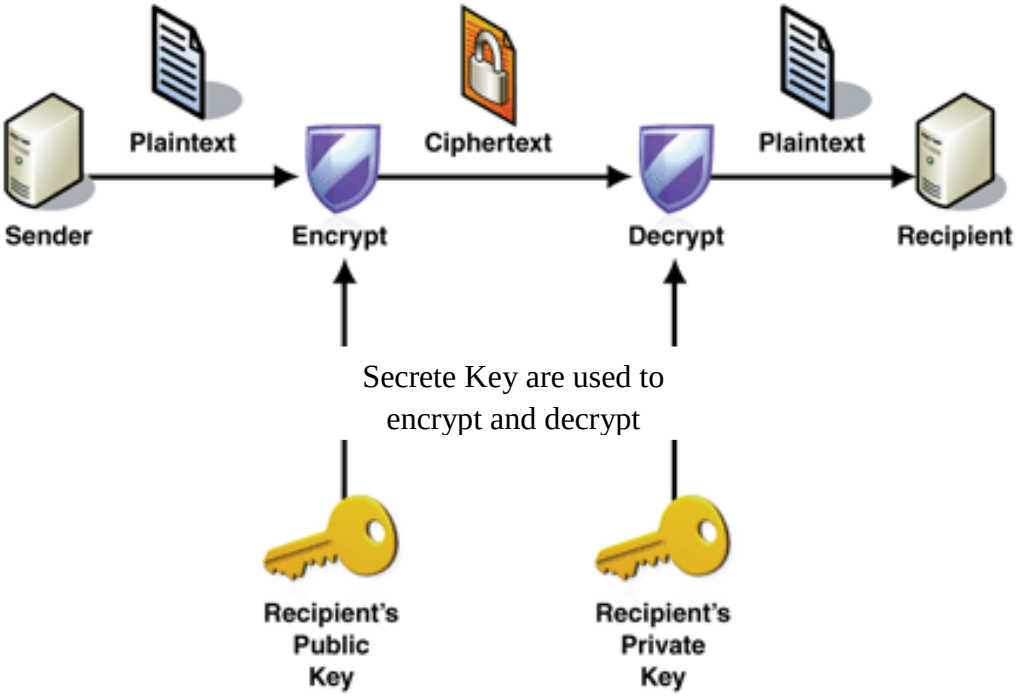
Encryption is also referred as cryptography, which uses a cipher system to change plaintext transforming it into a non-intelligible text. An authorized user can be able to easily decipher the message with the key provided by the owner to recipients, but not the unauthorized interceptors can decipher the message.

Figure shows the working of encryption technique.

Decryption:

Decryption is a methodology of changing encrypted data back into its original form. This process requires a public or private key.

2.2 SYSTEM ARCHITECTURE



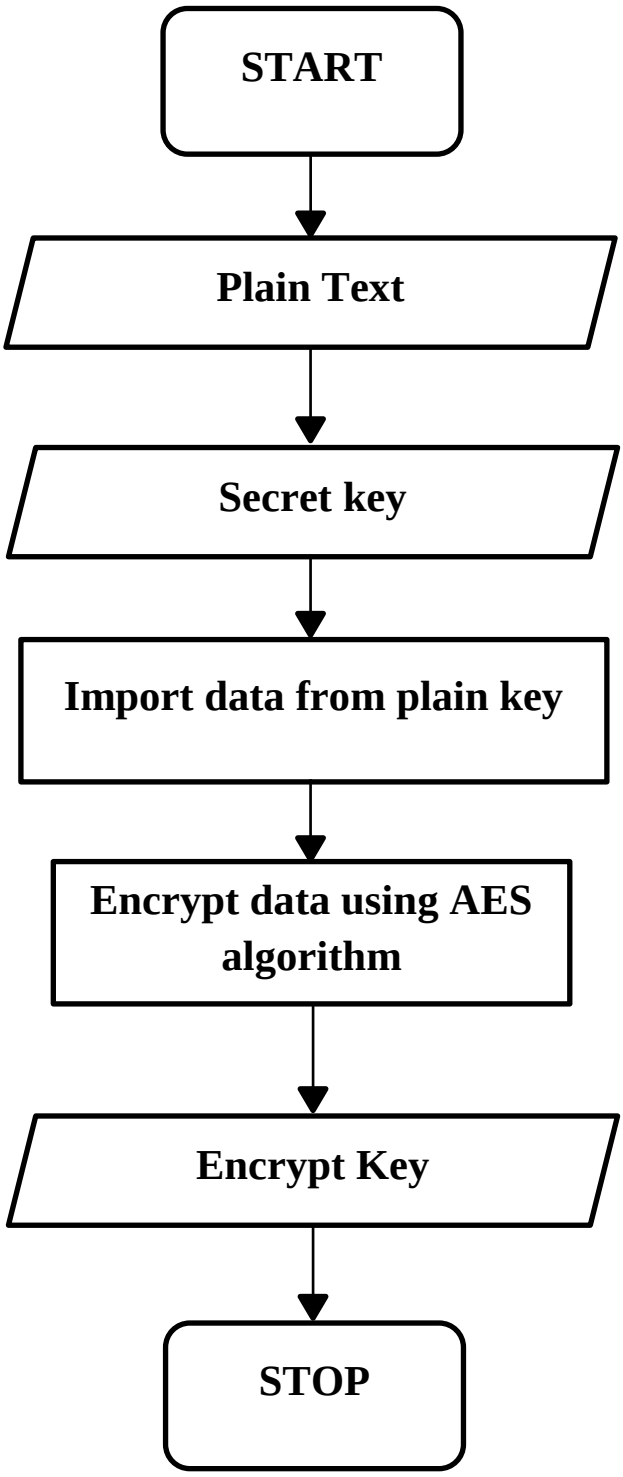
System architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.

In our system architecture, sender send the plain text, the text which we have to encrypt and we have to set an encryption key for encrypted method. After encryption the cipher text is used for decryption of message in which we have to copy the encrypted message and then enter the key set while encryption and the encrypted message will be decrypted

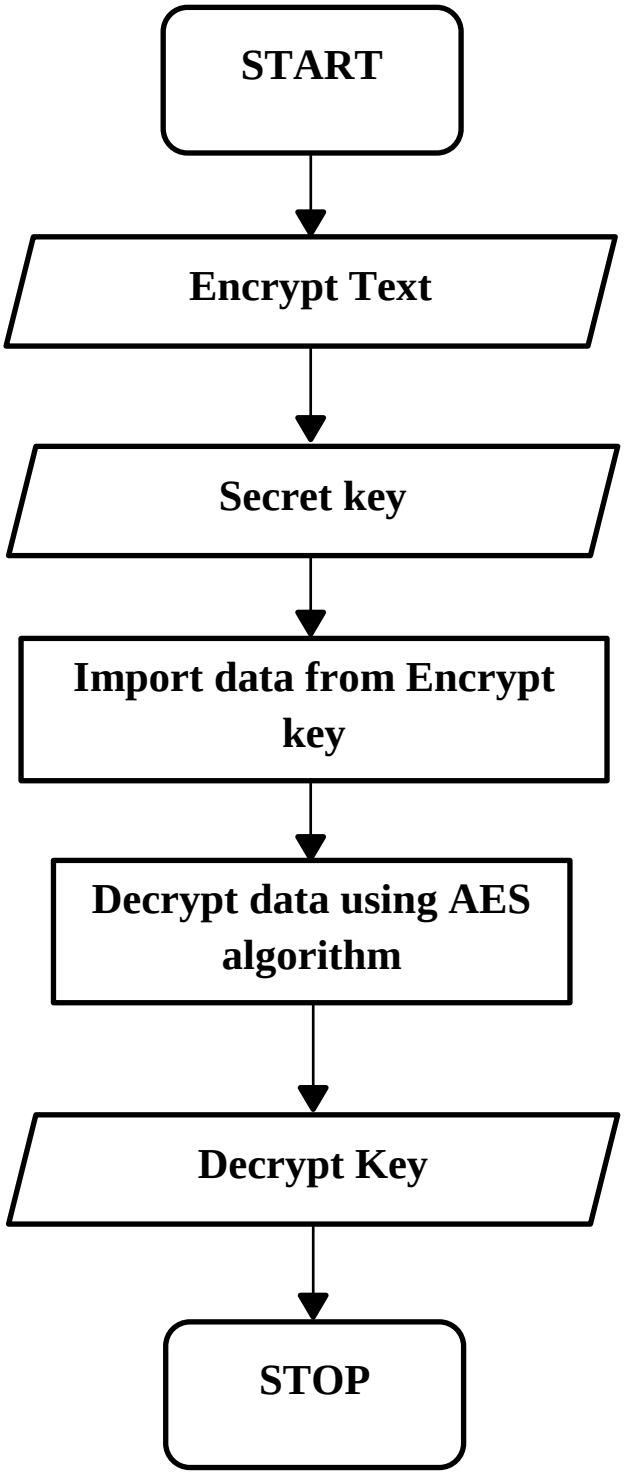
PROJECT DESIGN

3.1 Flow Chart

Encryption:



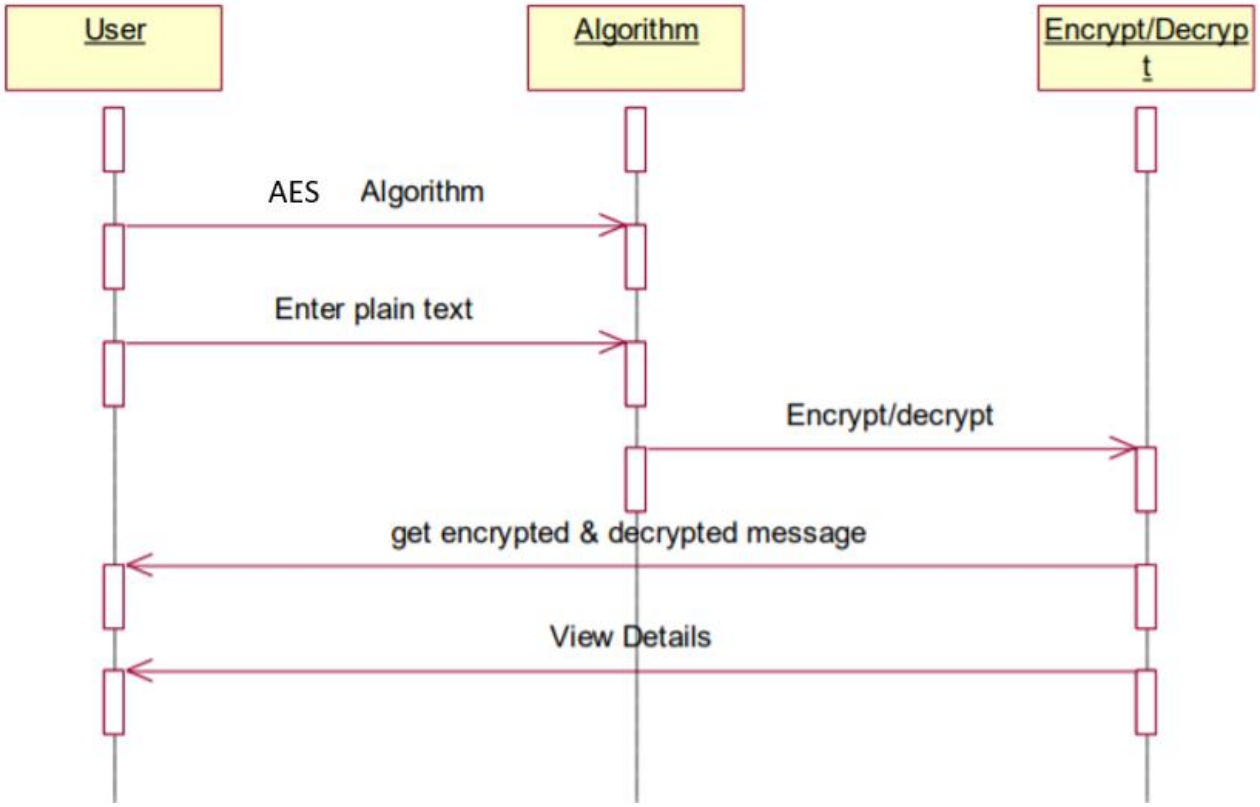
Decryption:-


START
Encrypt Text
Secret key
Import data from Encrypt key
Decrypt data using AES algorithm
Decrypt Key
STOP

3.2 Sequence Diagram:

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function.

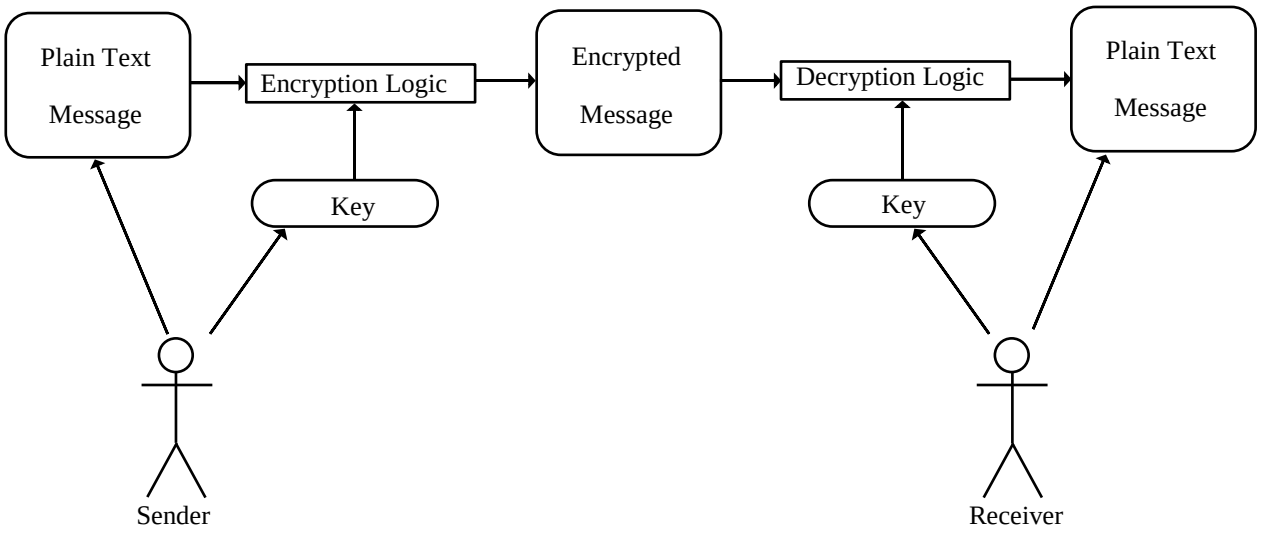
In this sequential diagram sender send the plain text, the text which we have to encrypt and we have to set an encryption key for encrypted method. After encryption the cipher text is used for decryption of message in which we have to copy the encrypted message and then enter the key set while encryption and the encrypted message will be decrypted



3.3 Use -Case Diagram:

A use case is a methodology used in system analysis to identify, clarify and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal.

In this, sender sends the plain text message, the text which we have to encrypt and we have to set an encryption key for encryption logic. After encryption the cipher text is used for decryption of message in which we have to copy the encrypted message and then enter the key set while encryption and the encrypted message will be decrypted.



CODING

Main File:-

```
package ecryptiondecryption;

public class EcryptionDecryption {

    /**

     * @param args the command line arguments

     */

    public static void main(String[] args) {

        EDcrypt run=new EDcrypt();

        run.setVisible(true);

    }

}
```

Classes File:-

```
package ecryptiondecryption;
import java.awt.Frame;
import java.awt.Toolkit;
import java.awt.datatransfer.StringSelection;
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;
import java.util.Base64;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
public class EDcrypt extends javax.swing.JFrame {
    /**
     * Creates new form EDcrypt
     */
    private static SecretKeySpec secretKey;
    private static byte[] key;
    public static void setKey(String myKey)
    {
        MessageDigest sha = null;
        try {
            key = myKey.getBytes("UTF-8");
```

```
            sha = MessageDigest.getInstance("SHA-1");
            key = sha.digest(key);
            key = Arrays.copyOf(key, 16);
            secretKey = new SecretKeySpec(key, "AES");
        }
        catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
    }
    public EDcrypt() {
        initComponents();
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN:initComponents
    private void initComponents() {

        jScrollPane1 = new javax.swing.JScrollPane();
        text1 = new javax.swing.JTextArea();
        jScrollPane2 = new javax.swing.JScrollPane();
        text2 = new javax.swing.JTextArea();
        jScrollPane3 = new javax.swing.JScrollPane();
        text3 = new javax.swing.JTextArea();
        jScrollPane4 = new javax.swing.JScrollPane();
        text4 = new javax.swing.JTextArea();
        msg1 = new javax.swing.JTextField();
        msg2 = new javax.swing.JTextField();
        encrypt = new javax.swing.JButton();
        decrypt = new javax.swing.JButton();
        copyencrypt = new javax.swing.JButton();
        copydecrypt = new javax.swing.JButton();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel1 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        message1 = new javax.swing.JLabel();
        message2 = new javax.swing.JLabel();
        mainsection = new javax.swing.JLabel();
```

```
        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Encryption and Decryption ");
        setAlwaysOnTop(true);
        setUndecorated(true);
        setResizable(false);
        getContentPane().setLayout(null);

        text1.setColumns(20);
        text1.setFont(new java.awt.Font("Dialog", 0, 14)); // NOI18N
        text1.setRows(5);
        text1.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(102,
102, 102), 2));
        jScrollPane1.setViewportView(text1);

        getContentPane().add(jScrollPane1);
        jScrollPane1.setBounds(80, 80, 300, 120);

        text2.setColumns(20);
        text2.setFont(new java.awt.Font("Dialog", 0, 14)); // NOI18N
        text2.setRows(5);
        text2.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(102,
102, 102), 2));
        jScrollPane2.setViewportView(text2);

        getContentPane().add(jScrollPane2);
        jScrollPane2.setBounds(80, 322, 300, 120);

        text3.setColumns(20);
        text3.setFont(new java.awt.Font("Dialog", 0, 14)); // NOI18N
        text3.setRows(5);
        text3.setToolTipText("");
        text3.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(102,
102, 102), 2));
        jScrollPane3.setViewportView(text3);

        getContentPane().add(jScrollPane3);
        jScrollPane3.setBounds(470, 80, 320, 120);

        text4.setColumns(20);
        text4.setFont(new java.awt.Font("Dialog", 0, 14)); // NOI18N
        text4.setRows(5);
        text4.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(102,
102, 102), 2));
```

```
jScrollPane4.setViewportView(text4);

getContentPane().add(jScrollPane4);
jScrollPane4.setBounds(470, 320, 320, 120);

msg1.setHorizontalAlignment(javax.swing.JTextField.CENTER);
msg1.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      msg1ActionPerformed(evt);
   }
});
getContentPane().add(msg1);
msg1.setBounds(200, 220, 180, 30);

msg2.setHorizontalAlignment(javax.swing.JTextField.CENTER);
getContentPane().add(msg2);
msg2.setBounds(595, 220, 190, 30);

encrypt.setBackground(new java.awt.Color(0, 51, 51));
encrypt.setFont(new java.awt.Font("Dialog", 1, 14)); // NOI18N
encrypt.setForeground(new java.awt.Color(255, 255, 255));
encrypt.setText("Encrypt");
encrypt.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      encryptActionPerformed(evt);
   }
});
getContentPane().add(encrypt);
encrypt.setBounds(80, 275, 90, 30);

decrypt.setBackground(new java.awt.Color(0, 51, 51));
decrypt.setFont(new java.awt.Font("Dialog", 1, 14)); // NOI18N
decrypt.setForeground(new java.awt.Color(255, 255, 255));
decrypt.setText("Decrypt");
decrypt.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      decryptActionPerformed(evt);
   }
});
getContentPane().add(decrypt);
decrypt.setBounds(470, 275, 90, 30);

copyencrypt.setBackground(new java.awt.Color(102, 0, 0));
copyencrypt.setFont(new java.awt.Font("Dialog", 1, 12)); // NOI18N
```

```
copyencrypt.setForeground(new java.awt.Color(255, 255, 255));
copyencrypt.setText("Copy Encryption");
copyencrypt.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        copyencryptActionPerformed(evt);
    }
});
getContentPane().add(copyencrypt);
copyencrypt.setBounds(240, 275, 140, 30);

copydecrypt.setBackground(new java.awt.Color(102, 0, 0));
copydecrypt.setFont(new java.awt.Font("Dialog", 1, 12)); // NOI18N
copydecrypt.setForeground(new java.awt.Color(255, 255, 255));
copydecrypt.setText("Copy Decryption");
copydecrypt.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        copydecryptActionPerformed(evt);
    }
});
getContentPane().add(copydecrypt);
copydecrypt.setBounds(633, 275, 150, 30);

jLabel2.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
jLabel2.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mousePressed(java.awt.event.MouseEvent evt) {
        jLabel2MousePressed(evt);
    }
});
getContentPane().add(jLabel2);
jLabel2.setBounds(810, 5, 30, 20);

jLabel3.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
jLabel3.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mousePressed(java.awt.event.MouseEvent evt) {
        jLabel3MousePressed(evt);
    }
});
getContentPane().add(jLabel3);
jLabel3.setBounds(780, 5, 30, 20);

jLabel1.setFont(new java.awt.Font("Dialog", 1, 14)); // NOI18N
jLabel1.setForeground(new java.awt.Color(204, 51, 0));
jLabel1.setText("Encryption Key");
```

```
        jLabel1.setBorder(javax.swing.BorderFactory.createLineBorder(new
    java.awt.Color(102, 102, 102)));
        getContentPane().add(jLabel1);
        jLabel1.setBounds(80, 220, 110, 30);

        jLabel4.setFont(new java.awt.Font("Dialog", 1, 14)); // NOI18N
        jLabel4.setForeground(new java.awt.Color(204, 51, 0));
        jLabel4.setText("Decryption Key");
        jLabel4.setBorder(javax.swing.BorderFactory.createLineBorder(new
    java.awt.Color(102, 102, 102)));
        getContentPane().add(jLabel4);
        jLabel4.setBounds(470, 220, 110, 30);

        jLabel5.setFont(new java.awt.Font("Dialog", 1, 18)); // NOI18N
        jLabel5.setForeground(new java.awt.Color(0, 0, 51));
        jLabel5.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel5.setText("Message to Decrypt");
        getContentPane().add(jLabel5);
        jLabel5.setBounds(470, 50, 300, 30);

        jLabel6.setFont(new java.awt.Font("Dialog", 1, 18)); // NOI18N
        jLabel6.setForeground(new java.awt.Color(0, 0, 51));
        jLabel6.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel6.setText("Message to Encrypt ");
        getContentPane().add(jLabel6);
        jLabel6.setBounds(80, 50, 300, 30);

        message1.setForeground(new java.awt.Color(204, 0, 0));
        message1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        getContentPane().add(message1);
        message1.setBounds(80, 450, 300, 20);

        message2.setForeground(new java.awt.Color(204, 0, 0));
        message2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        getContentPane().add(message2);
        message2.setBounds(470, 450, 320, 20);

        mainsection.setForeground(new java.awt.Color(153, 0, 0));
        mainsection.setIcon(new
    javax.swing.ImageIcon(getClass().getResource("/image/edcrypt.png"))); // NOI18N
        mainsection.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        getContentPane().add(mainsection);
        mainsection.setBounds(0, 0, 850, 500);
```

```
        setSize(new java.awt.Dimension(850, 499));
        setLocationRelativeTo(null);
    }// </editor-fold>//GEN-END:initComponents

    private void jLabel2MousePressed(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_jLabel2MousePressed
        // TODO add your handling code here:
        System.exit(0);

    }//GEN-LAST:event_jLabel2MousePressed

    private void jLabel3MousePressed(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_jLabel3MousePressed
        // TODO add your handling code here:
        this.setState(Frame.ICONIFIED);
    }//GEN-LAST:event_jLabel3MousePressed

    private void encryptActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_encryptActionPerformed
        // TODO add your handling code here:
        String strToEncrypt;
        String secret;
        try
        {
            strToEncrypt=text1.getText();
            secret=msg1.getText();
            setKey(secret);
            Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
            cipher.init(Cipher.ENCRYPT_MODE, secretKey);

text2.setText(Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes("U
TF-8"))));

        }
        catch (Exception e)
        {
            text2.setText("Please fill up the right secret key");

        }
    }//GEN-LAST:event_encryptActionPerformed

    private void copyencryptActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_copyencryptActionPerformed
        // TODO add your handling code here:
```

```
        Toolkit.getDefaultToolkit().getSystemClipboard().setContents(new
StringSelection(text2.getText()),null);
        message1.setText("Your encryption result is copied!");
        message2.setText("");
    }//GEN-LAST:event_copyencryptActionPerformed

    private void decryptActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_decryptActionPerformed
        // TODO add your handling code here:
        String secret;
        String strToDecrypt;
        try
        {
            secret=msg2.getText();
            strToDecrypt=text3.getText();
            setKey(secret);
            Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");
            cipher.init(Cipher.DECRYPT_MODE, secretKey);
            text4.setText(new
String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt))));
            //return new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));
        }
        catch (Exception e)
        {
            text4.setText("Please fill up the right secret key");
        }
    }//GEN-LAST:event_decryptActionPerformed

    private void copydecryptActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_copydecryptActionPerformed
        // TODO add your handling code here:
        Toolkit.getDefaultToolkit().getSystemClipboard().setContents(new
StringSelection(text4.getText()),null);
        message2.setText("Your decryption result is copied!");
        message1.setText("");
    }//GEN-LAST:event_copydecryptActionPerformed

    private void msg1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_msg1ActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_msg1ActionPerformed

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

```
    */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and
feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(EDcrypt.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(EDcrypt.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(EDcrypt.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(EDcrypt.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new EDcrypt().setVisible(true);
            }
        });
    }
```

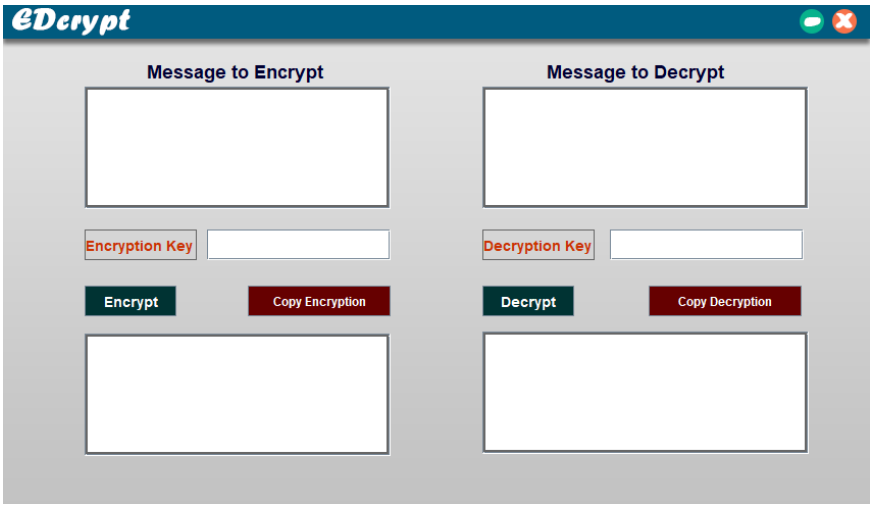
```
    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton copydecrypt;
    private javax.swing.JButton copyencrypt;
    private javax.swing.JButton decrypt;
    private javax.swing.JButton encrypt;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JScrollPane jScrollPane3;
    private javax.swing.JScrollPane jScrollPane4;
    private javax.swing.JLabel mainsection;
    private javax.swing.JLabel message1;
    private javax.swing.JLabel message2;
    private javax.swing.JTextField msg1;
    private javax.swing.JTextField msg2;
    private javax.swing.JTextArea text1;
    private javax.swing.JTextArea text2;
    private javax.swing.JTextArea text3;
    private javax.swing.JTextArea text4;
    // End of variables declaration//GEN-END:variables
}
```

JFrame:-

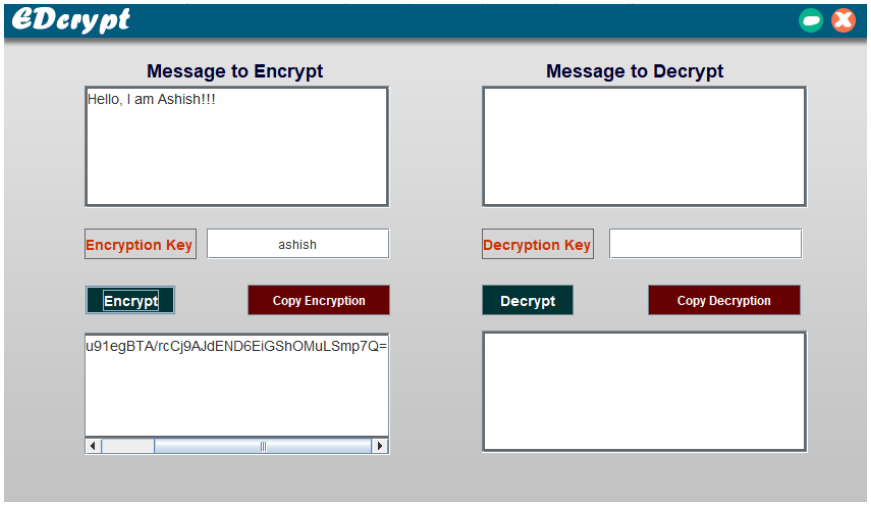
SNAPSHOTS

RESULT ANALYSIS:-

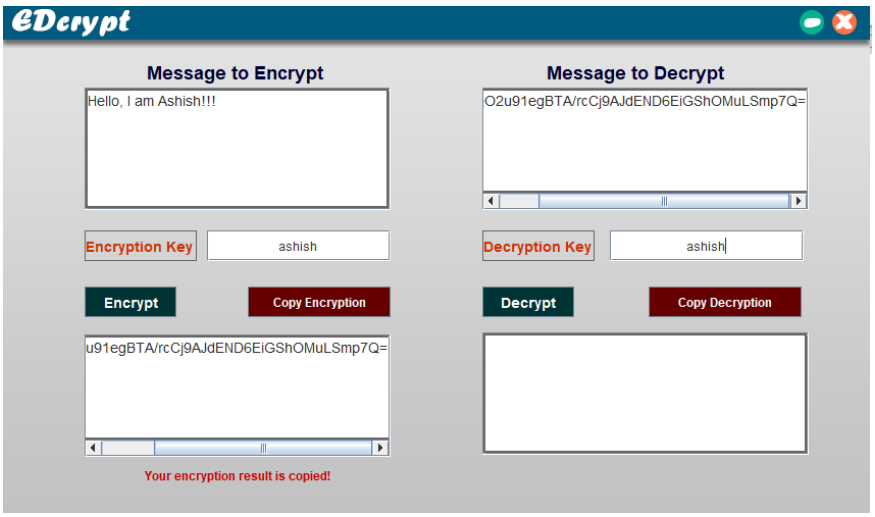
Main window



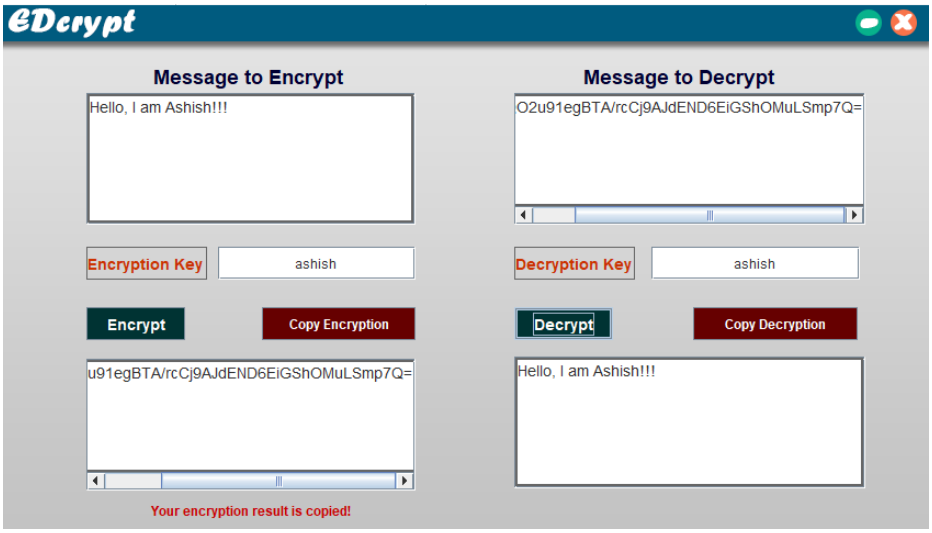
Message and Key for Encryption



Encrypted Message and Secrete key for decryption



Final Decrypted Message:-



SOFTWARE TESTING

6.1 Software Testing:

Software testing is an investigation conducted to provide stakeholders with information about the quality of the software product or service under test.[1] Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding software bugs (errors or other defects) and verifying that the software product is fit for use.

Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test

- Meets the requirements that guided its design and development,
- Responds correctly to all kinds of inputs,
- Performs its functions within an acceptable time,
- Is sufficiently usable,
- Can be installed and run in its intended environments, and
- Achieves the general result its stakeholder's desire.

As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources. As a result, software testing typically (but not exclusively) attempts to execute a program or application with the intent of finding software bugs (errors or other defects). The job of testing is an iterative process as when one bug is fixed; it can illuminate other, deeper bugs, or can even create new ones.

Software testing can provide objective, independent information about the quality of software and risk of its failure to users or sponsors.

Software testing can be conducted as soon as executable software (even if partially complete) exists. The overall approach to software development often determines when and how testing is conducted. For example, in a phased process, most testing occurs after system requirements have been defined and then implemented in testable programs. In contrast, under an agile approach, requirements, programming, and testing are often done concurrently.

6.1.1 System Testing

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic.[1]

As a rule, system testing takes, as its input, all of the "integrated" software components that have passed integration testing and also the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called assemblages) or between any of the assemblages and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

6.1.2 Unit Testing

Unit Testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method.

Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process. It forms the basis for component testing. Ideally, each test case is independent from the others. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended.

6.1.3 Integration Testing

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a

group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

6.1.4 Functional Testing

Functional testing is a quality assurance (QA) process and a type of black-box testing that bases its test cases on the specifications of the software component under test. Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered Functional testing usually describes what the system does.

Functional testing does not imply that you are testing a function (method) of your module or class. Functional testing tests a slice of functionality of the whole system.

Function 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.

Procedures: Interfacing system or procedures must be invoked.

6.2 Testing Method:

6.2.1 Black Box Testing

Black-box testing treats the software as a "black box", examining functionality without any knowledge of internal implementation, without seeing the source code. The testers are only aware of what the software is supposed to do, not how it does it.[15] Black-box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing, and specification-based testing.

Specification-based testing aims to test the functionality of software according to the applicable requirements. This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behavior), either "is" or "is not" the same as the expected value specified in the test case. Test cases are built around specifications and requirements, i.e., what the application is supposed to do. It uses external descriptions of the software, including specifications, requirements, and designs to derive test cases. These tests can be functional or non-functional, though usually functional.

6.1.2 White Box Testing

White-box testing (also known as clear box testing, glass box testing, and transparent box testing and structural testing, by seeing the source code) tests internal structures or workings of a program, as opposed to the functionality exposed to the end-user. In white-box testing, internal perspectives of the system, as well as programming skills, are used to design test cases.

The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

ADVANTAGES & DISADVANTAGES

Benefits or advantages of AES

Following are the benefits or advantages of AES:

- As it is implemented in both hardware and software, it is most robust security protocol.
- It uses higher length key sizes such as 128, 192 and 256 bits for encryption. Hence it makes AES algorithm more robust against hacking.
- It is most common security protocol used for wide various of applications such as wireless communication, financial transactions, e-business, encrypted data storage etc.
- It is one of the most spread commercial and open source solutions used all over the world.
- No one can hack your personal information.
- For 128 bit, about 2^{128} attempts are needed to break. This makes it very difficult to hack it as a result it is very safe protocol.

Drawbacks or disadvantages of AES

Following are the disadvantages of AES:

- It uses too simple algebraic structure.
- Every block is always encrypted in the same way.
- Hard to implement with software.
- AES in counter mode is complex to implement in software taking both performance and security into considerations.

CONCLUSION & FUTURE SCOPE

8.1 Conclusion:

It can be concluded that this project works efficiently for offline purpose at any place provided the requirements for running environment remains same. Users need not to worry about how the algorithm actually works inside the instruction set performed by the compiler.

8.2 Future Scope:

1. In Future, Data Encryption and Decryption system can be updated further which can provide more secured encryption algorithms like DES, AES etc.
2. This project can be made more attractive and easier to use by implementing GUI (Graphical User Interface) with the help of many coding platforms like Eclipse etc.
3. In Future, We could work on selection of a larger key size which would make the algorithm is more secure, and a larger input block to increase the throughput.

REFERENCES

- Yenuguvanilanka. J., & Elkeelany, O. (2008. April). Performance evaluation of hardware models of Advanced Encryption Standard (AES) algorithm. In Southeastcon, 2008. IEEE (pp. 222-225).
- "AES Website", https://en.wikipedia.org/wiki/Advanced_Encryption_Standard
- "Java Website", http://www.en.wikipedia.org/wiki/Java_(programming_language)
- Lu. C. C.. & Tseng, S. Y. (2002). Integrated design of AES (Advanced Encryption Standard) encrypter and decrypter. In Application-Specific Systems. Architectures and Processors. 2002. Proceedings. The IEEE International Conference on (pp. 277-285).
- Mohamed. A. A., & Madian, A. H. (2010. December). A Modified Rijndael Algorithm and its Implementation using FPGA. In Electronics. Circuits, and Systems (ICECS), 2010 17th IEEE International Conference on (pp. 335-338).
- Deshpande. H. S., Karande, K. J., & Mulani, A. 0. (2014. April). Efficient implementation of AES algorithm on FPGA. In Communications and Signal Processing (ICCSP), 2014 IEEE International Conference on (pp. 1895-1899).
- Nadeem, H (2006). A performance comparison of data encryption algorithms," IEEE Information and Communication Technologies, (pp. 84-89).