TEXT FILE ENCRYPTION AND DECRYPTION

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

Bachelor of Technology/Master of Technology

In

Computer Science and Engineering

School of Engineering and Sciences

Submitted by

D Sai Dhanush-AP21110010397

Under the Guidance of

Ms. Kavitha Rani

SRM University–AP

Neerukonda, Mangalagiri, Guntur

Andhra Pradesh – 522 240

[06, 2023]

Certificate

Date:02-06-2023

This is to certify that the work present in this Project entitled "**TEXT FILE ENCRYPTION AND DECRYPTION**" has been carried out by **D Sai Dhanush** under my/our supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology/Master of Technology in **School of Engineering and Sciences**.

Supervisor

(Signature)

Prof. / Dr. [Name]

Designation,

Affiliation.

Co-supervisor

(Signature)

Prof. / Dr. [Name]

Designation,

Affiliation.

Acknowledgements

In completing this project we have been fortunate enough to have help, support and encouragement from many people. I would like to acknowledge them for their cooperation.

We would also like to acknowledge the contribution of faculty member of the department for their kind assistance, suggestions and cooperation throughout the development of the project.

Finally, we would like to thank our classmates for the encouragement and help during the project.

Table of Contents

Abstract

This text file encryption and decryption is a daily life useful application. In our daily life we always shares information from one user to another user, but sometimes the sending information is hacked by other users here our secret information is going for all or other person. To prevent this problem we developed a application in java that encryption and decryption of text file here user can encrypt the data(the original data is converted into unreadable form) he need to send to other person. After encryption he need to set a password and can send this encrypted file to other person along with password he set. The person who receive this file can decrypt the file using decryption option here he need to enter correct password set by first user or sender. Then only the file will be decrypted and he can access the content in the file. In case he enter wrong password the file cannot be accessed. Or by mistake file is gone to wrong person he also cannot open it because he don't know the password. We had also used some graphic interface to make our project look good. And it can also access the all folder in the device.

1. Introduction

Text File Encryption and Decryption is a Java project that provides a secure way of encrypting and decrypting text files using AES (Advanced Encryption Standard). AES is a widely used encryption algorithm that provides strong security and is suitable for various applications.

The project allows users to select a text file and choose the AES encryption algorithm and key length to encrypt the file. The user can also provide a password to encrypt the file, which will be required to decrypt the file later. The encrypted file can be saved at a user-specified location.

To decrypt the file, the user can select the encrypted file and provide the same password used during encryption. The project will then decrypt the file using the chosen AES encryption algorithm and key length, and the decrypted file can be saved at a user-specified location.

The project's user interface is simple and easy to use, making it suitable for users with minimal technical knowledge. The project is useful in various industries where data privacy and security are critical, including healthcare, finance, and government. Overall, the Text File Encryption and Decryption Java project is an essential tool for users who want to keep their data secure and confidential using the AES encryption algorithm.

2. Methodology

Using the AES encryption algorithm, a plain text message is converted into a cipher text with the help of a secret key that is only known to the sender and receiver of the message. Encrypting or decrypting a message or a string is supported by Java Cryptographic Extension (JCE) framework in Java.

The Java Cryptographic Extension framework provides different packages for encryption and decryption.

- java.security
- java.security.cert
- java.security.spec
- java.security.interfaces
- javax.crypto
- javax.crypto.spec
- javax.crypto.interfaces

While decrypting a message, the reverse process of encryption is followed. It requires the value of the secret key in order to acquire the original message. The Cipher class in Java is used for the encryption and decryption process. The init() method of the Cipher class initializes the cipher using the public key from the given transformation type.

3. Concluding Remarks

Now a days all we are using accounts in many daily life activities and using each account we are sharing data to friend and other persons. This encryption and decryption are very important for securing your data in or securing your social media account from hacking or cyber attacks or from theft. Here password play a major role in security. So small password with length of 4 and 6 are easily hacked and your data is in danger. So our aim is to encrypt the file user provide into unreadable form and set a password provided by user . To decrypt file the password is need it cannot open if password is not provided.

By this your data will be secured and it can't be hacked easily. And we had final completed our project and this very useful for all and now a days all social media apps like whatsapp , Instagram and etc are using encryption and decryption to secure there users data from cyber attacks.

CODE:

```
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import javax.swing.*;
import java.awt.*;
import java.io.*;
import java.security.MessageDigest;
import java.util.Arrays;

public class javapro extends JFrame {
 public javapro() {
  super();
  setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  JFrame Frame=new JFrame("JEncrypt");
  JPanel panel = new JPanel() {
   @Override
   protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    Image image = new ImageIcon().getImage();
    g.drawImage(image, 0, 0, getWidth(), getHeight(), this);
   }
  };
  panel.setLayout(null);
  panel.setOpaque(false);

  Image icon =
Toolkit.getDefaultToolkit().getImage("C:\\Users\\DELL\\Documents\\operating
systems\\icon.png");

  JButton en = new JButton("ENCRYPT A FILE");
  en.setBounds(75, 100, 250, 40);
```

```
en.addActionListener(e -> {
 try {
  String inputFile = getFile("Select a file to encrypt");
  String outputFile = getSaveFile("Select a location to save the encrypted file");
  String key = getKey();
  encryptFile(inputFile, outputFile, key);
  JOptionPane.showMessageDialog(null, "File encrypted successfully!");
 } catch (Exception ex) {
  JOptionPane.showMessageDialog(null, "Error encrypting file: " + ex.getMessage());
 }
});

JButton de = new JButton("DECRYPT A FILE");
de.setBounds(75, 150, 250, 40);
de.addActionListener(e -> {
 try {
  String inputFile = getFile("Select a file to decrypt");
  String outputFile = getSaveFile("Select a location to save the decrypted file");
  String key = JOptionPane.showInputDialog(null, "Enter the secret key for the encrypted
file:");
  decryptFile(inputFile, outputFile, key);
  JOptionPane.showMessageDialog(null, "File decrypted successfully!");
 } catch (Exception ex) {
  JOptionPane.showMessageDialog(null, "Error decrypting file: " + ex.getMessage());
 }
});

JButton exit = new JButton("EXIT");
exit.setBounds(75, 200, 250, 40);
exit.addActionListener(e -> System.exit(0));
```

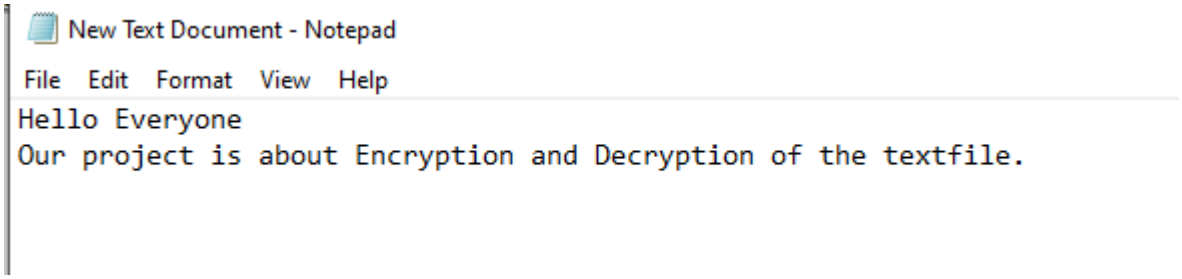
```
 panel.add(en);
 panel.add(de);
 panel.add(exit);
 Frame.setIconImage(icon);
 Frame.add(panel);
 Frame.setVisible(true);
 Frame.setContentPane(panel);
 Frame.setSize(400, 500);
 Frame.setLocationRelativeTo(null);
}

private static String getFile(String message) {
 JFileChooser chooser = new JFileChooser();
 chooser.setDialogTitle(message);
 int result = chooser.showOpenDialog(null);
 if (result == JFileChooser.APPROVE_OPTION) {
  return chooser.getSelectedFile().getAbsolutePath();
 } else {
  throw new RuntimeException("File not selected.");
 }
}

private static String getSaveFile(String message) {
 JFileChooser chooser = new JFileChooser();
 chooser.setDialogTitle(message);
 int result = chooser.showSaveDialog(null);
 if (result == JFileChooser.APPROVE_OPTION) {
  return chooser.getSelectedFile().getAbsolutePath();
 } else {
  throw new RuntimeException("Location not selected.");
```

```
 }
 }

 private static String getKey() {
  String key = JOptionPane.showInputDialog(null, "Enter a secret key for the file (must be at least 16 characters):");
  if (key == null || key.length() < 16) {
   throw new RuntimeException("Invalid secret key.");
  }
  return key;
 }

 private static void encryptFile(String inputFile, String outputFile, String key) throws Exception {
  File inFile = new File(inputFile);
  FileInputStream fis = new FileInputStream(inFile);
  byte[] inputBytes = new byte[(int) inFile.length()];
  fis.read(inputBytes);
  fis.close();

  MessageDigest sha = MessageDigest.getInstance("SHA-256");
  byte[] keyBytes = Arrays.copyOf(sha.digest(key.getBytes("UTF-8")), 16);
  SecretKeySpec secretKeySpec = new SecretKeySpec(keyBytes, "AES");

  Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
  cipher.init(Cipher.ENCRYPT_MODE, secretKeySpec);

  byte[] outputBytes = cipher.doFinal(inputBytes);

  File outFile = new File(outputFile);
  FileOutputStream fos = new FileOutputStream(outFile);
```

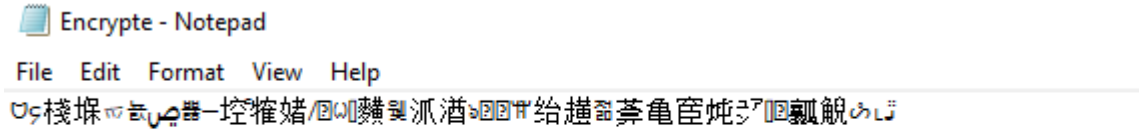
```
  fos.write(outputBytes);
  fos.close();
 }

 private static void decryptFile(String inputFile, String outputFile, String key) throws
Exception {
  File inFile = new File(inputFile);
  FileInputStream fis = new FileInputStream(inFile);
  byte[] inputBytes = new byte[(int) inFile.length()];
  fis.read(inputBytes);
  fis.close();

  MessageDigest sha = MessageDigest.getInstance("SHA-256");
  byte[] keyBytes = Arrays.copyOf(sha.digest(key.getBytes("UTF-8")), 16);
  SecretKeySpec secretKeySpec = new SecretKeySpec(keyBytes, "AES");

  Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
  cipher.init(Cipher.DECRYPT_MODE, secretKeySpec);

  byte[] outputBytes = cipher.doFinal(inputBytes);

  File outFile = new File(outputFile);
  FileOutputStream fos = new FileOutputStream(outFile);
  fos.write(outputBytes);
  fos.close();
 }

 public static void main(String[] args) {
  new javapro();
 }
}
```

OUTPUT:

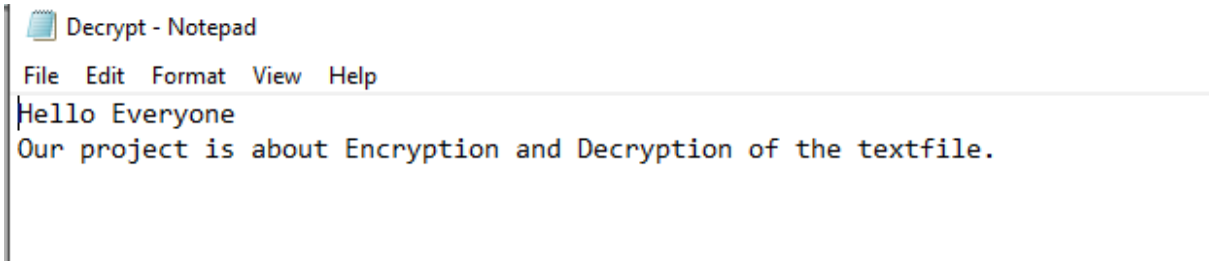
Selected folder



Encrypted form of selected folder



Decrypted form of selected folder



THANK YOU