RAJALAKSHMI ENGINEERING COLLEGE

[PROJECT REPORT]

CS23333-ObJECT ORIENTED PROGRAMING USING JAVA

Department of CYBER SECURITY
Semester 3

TEXT FILE ENCRYPTION DECRYPTION

A MINI-PROJECT REPORT

Submitted by

2116241901067 - NAVANEETHA KRISHNAN E
2116241901114 - SURESVARAKUMAR J S
2116241901502 - ARILLI JAGADEESH A
2116241901506 - MUGHIL SANKAR N
2116241901507 - SACHIN KRISHNA B
2116241901508 - SANJAY KRISHNA S
2116241901509 - YUDISH VENKAT V

in partial fulfillment of the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING (CYBER SECURITY)

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI
NOVEMBER 2025

BONAFIDE CERTIFICATE

CERTIFIED THAT THIS PROJECT **"TEXT FILE ENCRYPTION DECRYPTION"** IS THE BONAFIDE WORK OF **"NAVANEETHA KRISHNAN E, SURVESVARAKUMAR J S, ARILLI JAGADEESH A, MUGHIL SANKAR N, SACHIN KRISHNA B, SANJAY KRISHNA S & YUDISH VENKAT V"** WHO CARRIED OUT THE PROJECT WORK UNDER MY SUPERVISION.

SIGNATURE

Mrs. Janani. I

ASSISTANT PROFESSOR

Department of Computer Science and Engineering (Cyber Security)

Rajalakshmi Engineering College Chenna

This mini project report is submitted for the viva voce examination to be held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

The *Encryption System* is a Java-based academic project developed to enhance the understanding of network security principles. The system provides a secure mechanism for protecting data transmitted over public networks. It currently supports text-based file formats such as .txt and .java. The application allows a user to convert readable data (plaintext) into a non-readable encrypted format (ciphertext) using a user-defined password known only to the sender and the receiver. This ensures that even if an unauthorized user intercepts the data during transmission, the content remains unintelligible. On the receiver side, the same system and password are used to decrypt the ciphertext back into its original readable form. This project demonstrates the practical implementation of encryption and decryption concepts and can be extended to support additional file types and enhanced security features in the future.

ACKNOWLEDGEMENT

We like to convey our sincere appreciation to all who have supported and mentored us during the successful completion of this project work.

We express our profound gratitude to Mr. Benedict J.N., Associate Professor (SG) and Head of the Department of Computer Science Engineering and Cyber Security at Rajalakshmi Engineering College, for furnishing us with essential resources, support, and an enabling environment to execute this project.

We express our profound gratitude to Ms. L. JANANI , Assistant Professor in the Department of Computer Science Engineering and Cyber Security at Rajalakshmi Engineering College, for her unwavering support, invaluable insights, and collaboration throughout our endeavor.

We would like to convey our gratitude to our faculty members and colleagues for their valuable feedback and encouragement.

We express our gratitude to our families and friends for their steadfast support, patience and encouragement, which were instrumental in the effective execution of this seminar

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

With the rapid growth of digital communication systems, securing data during transmission has become a major concern. Sensitive information transmitted over open or public networks is vulnerable to interception, unauthorized access, and misuse. Encryption is a widely used security technique that protects data by transforming readable information into an unreadable format, ensuring that only authorized users can access it.

This project, *Encryption System*, is developed in Java and focuses on implementing a basic but functional data encryption and decryption process. It allows a user to encrypt .txt and .java files, making their content unreadable for unauthorized individuals. A password is required for both encryption and decryption, ensuring that only the intended recipient can retrieve the original data

1.2 SCOPE OF THE WORK

- The system allows encryption and decryption of .txt and .java files.
- It provides a password-based secured communication approach.
- The application is suitable for secure message transmission between two users over a public network.
- The system serves as a learning and demonstration tool for basic cryptographic concepts

1.3 PROBLEM STATEMENT

When data is transmitted over unsecured networks, there is a high risk of unauthorized interception. Without encryption, any third party may read or misuse the information. Therefore, a secure mechanism is needed to protect sensitive data during transfer.

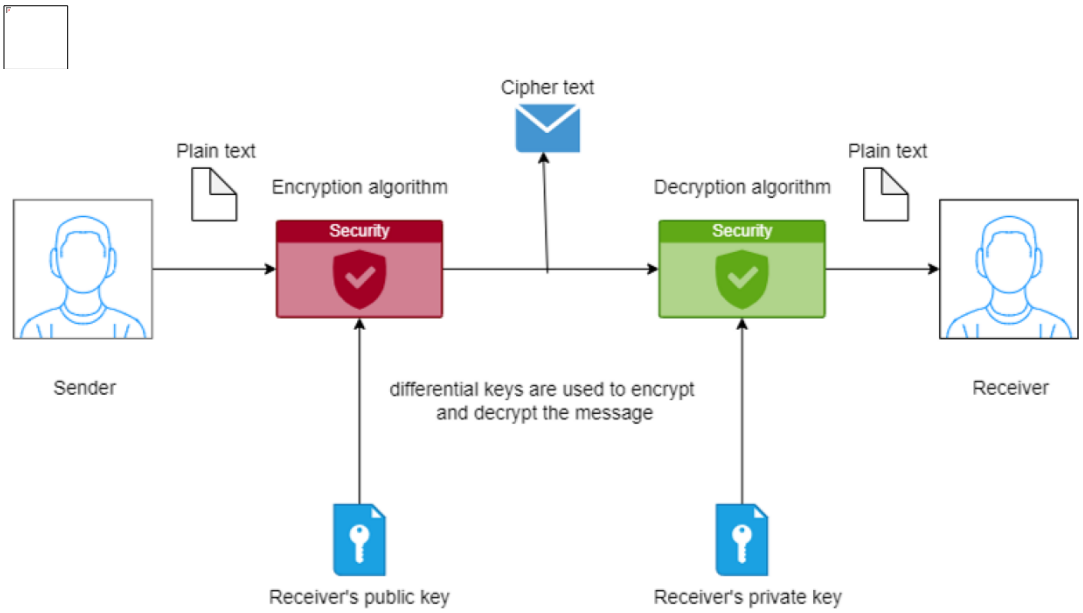
1.4 AIM AND OBJECTIVES OF THE PROJECT

Aim

To develop a Java-based system that securely encrypts and decrypts files using a password, ensuring safe transmission over public networks.

Objectives

- To convert plaintext into unreadable ciphertext using encryption algorithms.
- To implement password-based access control for encryption and decryption.
- To allow users to securely exchange files without the risk of unauthorized access.
- To provide a user-friendly interface demonstrating the practical use of network security concepts.



CHAPTER 2

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS

· Processor: Intel i3 or above

· RAM: Minimum 4 GB

· Hard Disk: Minimum 500 MB of free storage

· Input Devices: Keyboard and Mouse

· Display: Standard Monitor

2.2 SOFTWARE SPECIFICATIONS

· Operating System: Windows / Linux

· Programming Language: Java

· IDE/Editor: NetBeans / Eclipse / IntelliJ IDEA

· JDK: Java Development Kit (version 8 or above)

CHAPTER 3

The system is divided into the following modules:

1. File Input Module

Allows the user to select .txt or .java files for processing.

2. Encryption Module

Converts plaintext data into ciphertext using an algorithm and user-provided password.

3. Decryption Module

Reverts ciphertext to its original readable form using the same password.

4. Password Validation Module

Ensures that encryption and decryption can be performed only when the correct password is entered.

5. Output Generation Module

Generates the encrypted or decrypted file and saves it to the system.

CHAPTER 4

SAMPLE CODING

```
package in. CYBER. Encryption System;

import java.awt. *;
import java.awt. image.BufferedImage;
import java.io. *;
import javax. imageio. Image IO;
import javax. swing. Frame;
import javax. swing. *;



public class Main {
ImageIcon image;
Dimension d = null;
  public Main () {
            JFrame frame = new JFrame ("Encryption & Decryption Software");
            d=Toolkit.getDefaultToolkit(). getScreenSize();
            frame. set Size (width/2, height/2);
            frame. set Location (width/4, d. height/4);
            frame. set Resizable(false);
frame. setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
JPanel panel1=new JPanel ();
JPanel panel2=new JPanel ();
JPanel panel3=new JPanel ();
panel1.setPreferredSize(new Dimension (100,100));
panel2.setPreferredSize(new Dimension (100,100));
panel3.setPreferredSize(new Dimension (100,100));
panel1.setOpaque(true);
panel2.setOpaque(true);
panel3.setOpaque(true);
panel1.setBackground(Color.yellow);
panel1.setForeground(Color.red);
panel2.setBackground(Color.green);
panel2.setForeground(Color.red);
panel3.setBackground(Color.yellow);
panel3.setForeground(Color.red);
image=new ImageIcon("wait.GIF");
JLabel im= new JLabel(image);
JLabel msg=new JLabel("LOADING.....");
panel1.add(msg);
panel2.add(im);
frame.add (panel1, BorderLayout.NORTH);
frame.add (panel2, BorderLayout.CENTER);
```

```
frame.add (panel3, BorderLayout.SOUTH);
//frame. set Visible(true);
//try
//{
        //Thread.sleep(4000);
        new Cryptography ();
        //frame. set Visible(false);
        //frame. dispose ();
//}
//catch (InterruptedException ec) {}
}
public static void main (String args []) throws Exception {
        new Main ();
  }
}
```

Sample 1

```
 package in. CYBER. Encryption System;


import java.nio. *;
import java.nio. channels. *;
import java.io. *;
import java.awt. *;
import java.awt. event. *;
import java.awt. Color. *;
import javax. swing. *;
import javax. swing. filechooser. File Filter;
import java.util.*;
import java.awt. image. Buffered Image;
import javax. imageio. Image IO;
class JavaFileFilter extends FileFilter {
        public boolean accept (File file)
        {
                if (file. get Name (). ends with(".txt")) return true;
                if (file. get Name (). ends with(".java")) return true;
                if (file. is Directory ()) return true;
                return false;
        }
        public String getDescription ()
        {
                return "Java and Text file for Encryption and Dencryption";
        }
}

class SaveJavaFileFilter extends FileFilter
 {
   public boolean accept (File f)
```

```
   {
       if (f. isDirectory ())
             return true;

        String s = f. getName ();

       return s. endsWith(".java");
   }

   public String getDescription ()
   {
      return "*.java";
   }

}

class SaveTextFileFilter extends FileFilter
 {
   public boolean accept (File f)
   {
       if (f. isDirectory ())
             return true;
        String s = f. getName ();

       return s. endsWith(".txt");
   }

   public String getDescription ()
   {
      return "*.txt";
   }

}
public class Cryptography extends JFrame implements ActionListener
{
             public JButton browse, enc, denc, cancel;
    private JLabel label;
    private JTextField filename;
    private JFileChooser jfc;
    private File file;
    Dimension d = null;
FileInputStream fin;
FileOutputStream fout;
FileChannel fichan, fochan;
long fsize;
ByteBuffer mbuf, ombuf;
long key=0;
String ext="";
```

```
JScrollPane displayScrollPane;
ImageIcon image;
int wdth, hight;

   public Cryptography () {
            super ("Encryption and Decryption Software");
            d=Toolkit.getDefaultToolkit(). getScreenSize ();
            setBackground (Color.yellow);
                     setForeground (Color.red);
            wdth=d. width/2;
            hight=d. height/2;
        setSize (wdth, hight);
        setLocation (d. width/4, d. height/4);
        setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
        setResizable(false);
        label = new JLabel ("CHOOSE THE TEXT OR JAVA FILE FOR
ENCRYPTION OR DECRYPTION");
        filename = new JTextField (50);
        filename. set Editable(false);
        browse=new JButton("Browse");
        browse. set Foreground (Color. BLUE);
        enc=new JButton("Encrypt");
        enc. setForeground (Color.BLUE);
        enc. setEnabled(false);
        denc=new JButton("Dencrypt");
        denc. set Foreground (Color. BLUE);
        denc. set Enabled(false);
        cancel=new JButton("Cancel");
        cancel. set Foreground (Color. RED);
        jfc=new JFileChooser ();
        jfc. setFileFilter(new JavaFileFilter());
        JPanel buttonPanel1 = new JPanel();
        JPanel buttonPanel2 = new JPanel();
        JPanel loadPanel = new JPanel();
        buttonPanel1.setPreferredSize(new Dimension(100,100));
        buttonPanel2.setPreferredSize(new Dimension(100,100));
        loadPanel.setPreferredSize(new Dimension(100,100));
        buttonPanel1.setOpaque(true);
                        buttonPanel2.setOpaque(true);
                        loadPanel.setOpaque(true);
                        loadPanel.setBackground(Color.yellow);
                        buttonPanel1.setBackground(Color.green);
                        buttonPanel1.setForeground(Color.red);
                        buttonPanel2.setBackground(Color.green);
                        buttonPanel2.setForeground(Color.red);

            buttonPanel1.setBorder(BorderFactory.createLineBorder(Color.BLUE));
```

```
        buttonPanel2.setBorder(BorderFactory.createLineBorder(Color.BLUE));
                image=new ImageIcon("logo.GIF");
JLabel im= new JLabel(image);
loadPanel.add(im);

                buttonPanel1.add(label);
      buttonPanel1.add(filename);
      buttonPanel1.add(browse);
      buttonPanel2.add(enc);
      buttonPanel1.add(denc);
      buttonPanel2.add(cancel);
      label.setBounds(105, 10, 400, 25);
          browse.addActionListener(this);
          enc.addActionListener(this);
          denc.addActionListener(this);
          cancel.addActionListener(this);
          add(buttonPanel1, BorderLayout.NORTH);
      add(loadPanel, BorderLayout.CENTER);
      add(buttonPanel2, BorderLayout.SOUTH);
      setVisible(true);
   }
  public void actionPerformed(ActionEvent e) {
          if(e.getSource() == browse) {
                int result = jfc.showOpenDialog(null);
                        if(result==JFileChooser.APPROVE_OPTION)
                        {
                                label.setText("Selected file is :
"+jfc.getSelectedFile().getName());

          filename.setText(jfc.getSelectedFile().getPath());
                                file=jfc.getSelectedFile();
                                denc.setEnabled(true);
                                enc.setEnabled(true);
                        }
                        else
                        {
                                filename.setText("No file selected");
                                denc.setEnabled(false);
                                enc.setEnabled(false);
                        }
          }
          if(e.getSource()==enc)
          {
                try
                {
                key=enterKey();
                if(key>=1)
```

```
                    {
                    JOptionPane.showMessageDialog(null,"Save the Encrypted
file","ENCRYPTION COMPLETED",JOptionPane.INFORMATION_MESSAGE);
            convert(-key);
                    }
                    else

            JOptionPane.showMessageDialog(null,"Please enter a nonzero
key","WARNING",JOptionPane.WARNING_MESSAGE);
                    }catch(Exception ioee){
                    }
            }
            if(e.getSource()==denc)
            {try
            {
                                key=enterKey();
                                if(key>=1)
                                {
                                JOptionPane.showMessageDialog(null,"Save the
Decrypted file","DECRYPTION
COMPLETED",JOptionPane.INFORMATION_MESSAGE);
                                convert(key);
                                }
                                else

            JOptionPane.showMessageDialog(null,"Please enter a nonzero
key","WARNING",JOptionPane.WARNING_MESSAGE);
            }catch(Exception ex){}
            }
   if(e.getSource() == cancel)             {
                    System.exit(1);
            }
   }
private long enterKey() throws IOException
{
            long k=0;
            String key=JOptionPane.showInputDialog(null,"Enter the Key","SECURE
KEYS",JOptionPane.QUESTION_MESSAGE);
            long intKey=(long)Integer.parseInt(key);
            long temp=intKey;
            checking:
            {
            do
            {
            k=k+(intKey%10);
            intKey=intKey/10;
            }while(intKey>=10);
            k=k+intKey;
```

```
            if(k>32)
            {
                        intKey=temp/10;
                        break checking;
            }
            }
            return k;
}
private void convert(long secureKey)
{
            long Key=secureKey;
try
{
JFileChooser jFileChooser = new JFileChooser();
jFileChooser.addChoosableFileFilter(new SaveJavaFileFilter());
jFileChooser.addChoosableFileFilter(new SaveTextFileFilter());
jFileChooser.setSelectedFile(new File("fileToSave.txt"));
int responce = jFileChooser.showSaveDialog(null);
if(responce==JFileChooser.APPROVE_OPTION)
{
   String extension=jFileChooser.getFileFilter().getDescription();
   if(extension.equals("*.java"))
     {
        ext=".java";
     }
   if(extension.equals("*.txt"))
     {
        ext=".txt";
     }
}

fin=new FileInputStream(file);
fout=new FileOutputStream(jFileChooser.getSelectedFile()+ext);
fichan=fin.getChannel();
fochan=fout.getChannel();
fsize=fichan.size();
mbuf=ByteBuffer.allocate((int)fsize);
ombuf=ByteBuffer.allocate((int)fsize);
fichan.read(mbuf);
mbuf.rewind();
for(int i=0;i<fsize;i++)
{
long data=((long) mbuf.get());
ombuf.put((byte)(data+Key));
}
ombuf.rewind();
fochan.write(ombuf);
fichan.close();
```

```
fin.close();
fochan.close();
fout.close();
}
catch(IOException e)
{
System.out.println(e);
System.exit(1);
}
catch(BufferUnderflowException uf)
{
        System.out.println(uf);
}
 }
 public static void main(String args[]){
        SwingUtilities.invokeLater(new Runnable()
                {
                        public void run()
                        {
                                new Cryptography();
                        }
                });
        }

}
```

CHAPTER 5

SCREEN SHOTS

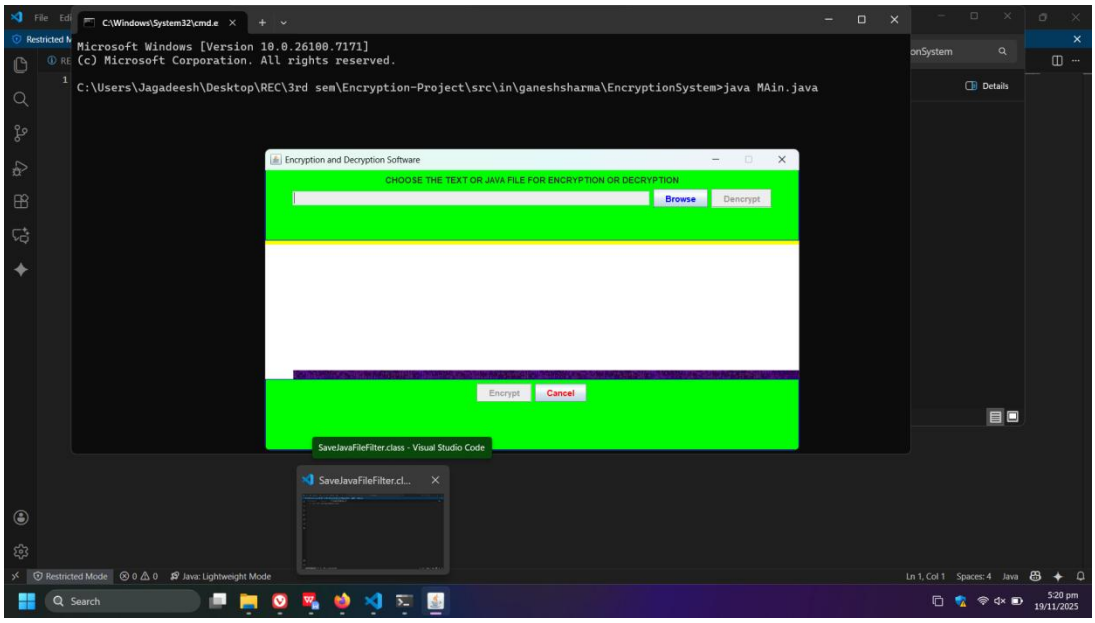


Fig 5.1 – Introduction Page

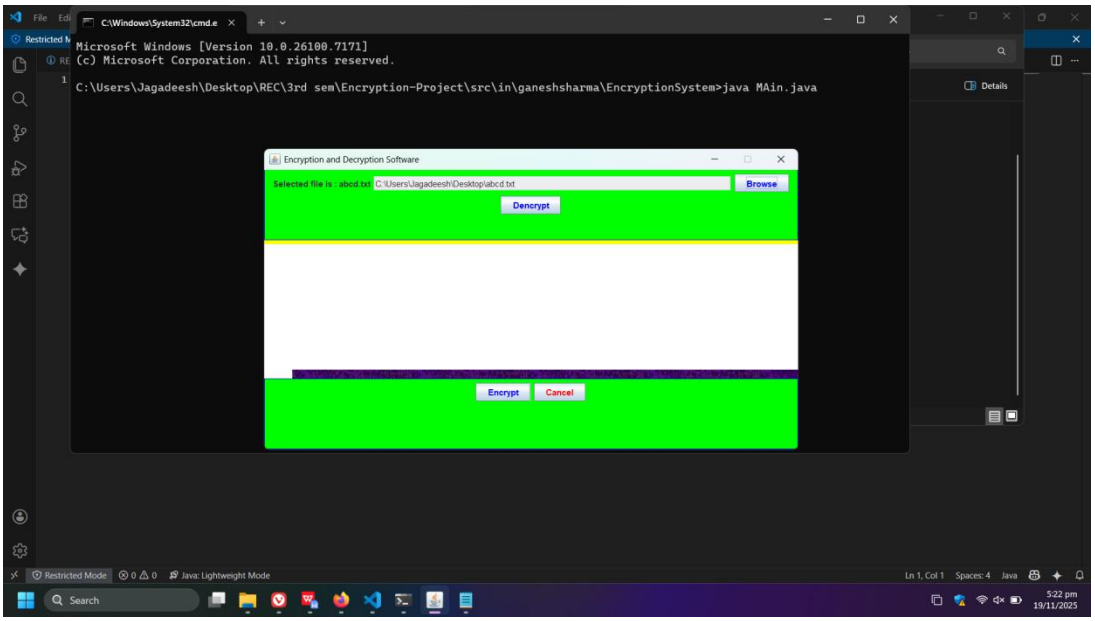


Fig 5.2 – Key for Encryption

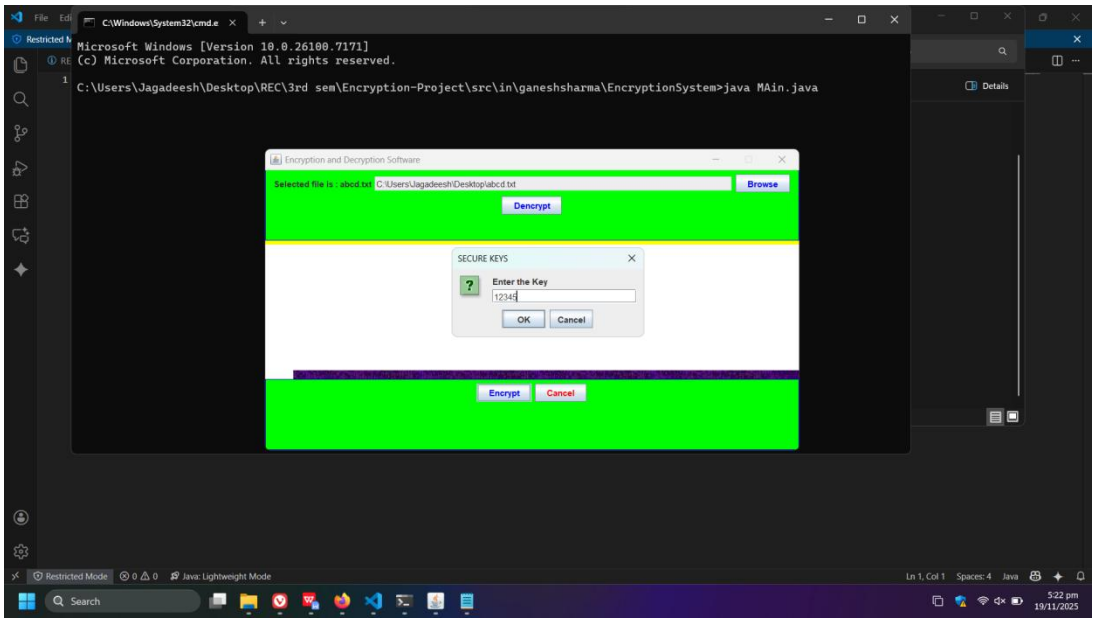


Fig 5.3 – Encryption Completed

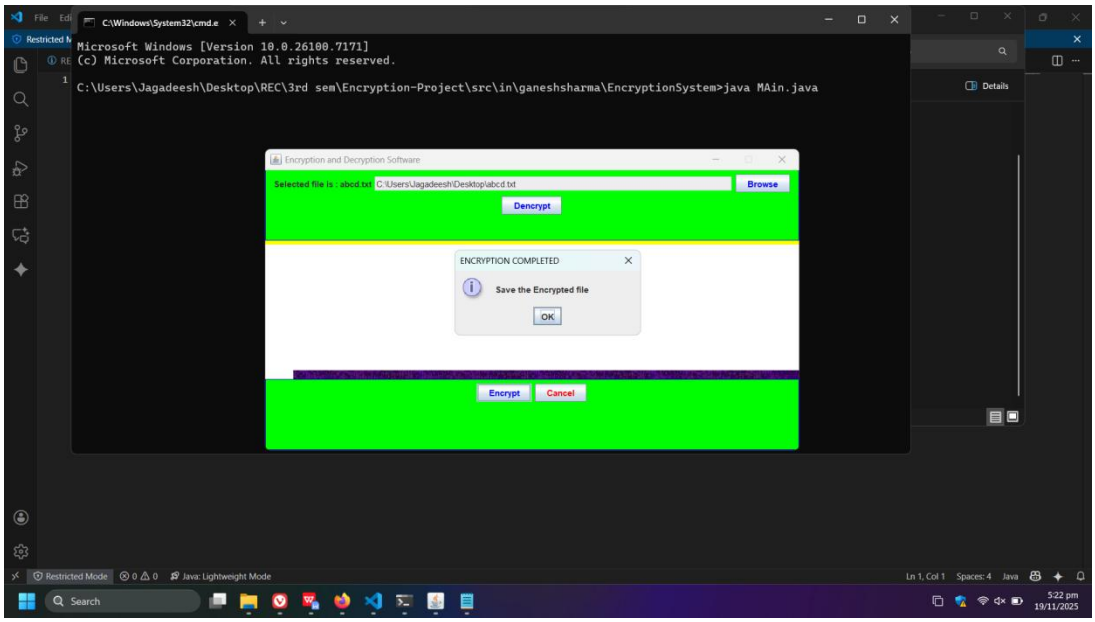


Fig 5.4 Encrypted (Hashed) File

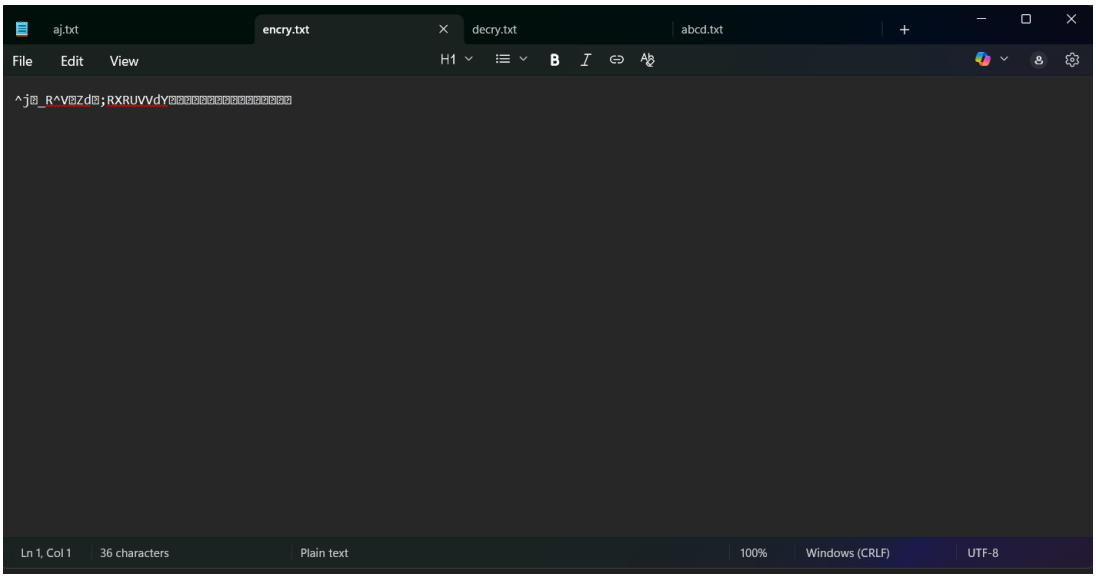
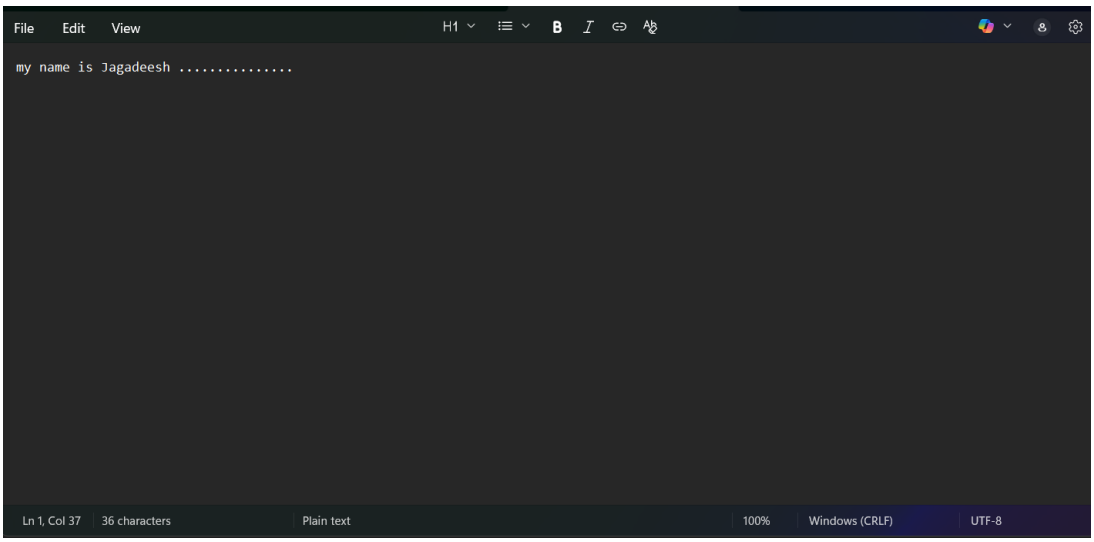


Fig 5.5 Decryption



CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

The Encryption System successfully demonstrates a basic implementation of cryptographic principles using Java. By encrypting files with a secure password, the system prevents unauthorized access to data during transmission. It serves as an effective learning project for understanding encryption, decryption, and secure communication.

FUTURE ENHANCEMENTS

- ✓ Support for additional file formats such as .pdf, .docx, and image files.
- ✓ Implementation of stronger cryptographic algorithms (AES, RSA, etc.).
- ✓ Development of a GUI-based desktop application.
- ✓ Cloud integration for online secure file transfer.
- ✓ Logging and monitoring features for security auditing.

REFERENCES

1. https://www.w3schools.com/sql/
2. https://www.tutorialspoint.com/sqlite/index.htm
3. https://www.wikipedia.org/
4. https://www.learnpython.org/
5. https://www.codecademy.com/learn/learn-python