Practice 4: MODERN SYMMETRIC ENCRYPTION

4.1 OVERVIEW

4.1.1 Introduction

- Lab 4: DES and 3DES Encryption Algorithm
- Practice time: class: 3 study hours, self-study: 3 study hours.
- Requirements: Students using Netbeans Software

4.1.2 Objective

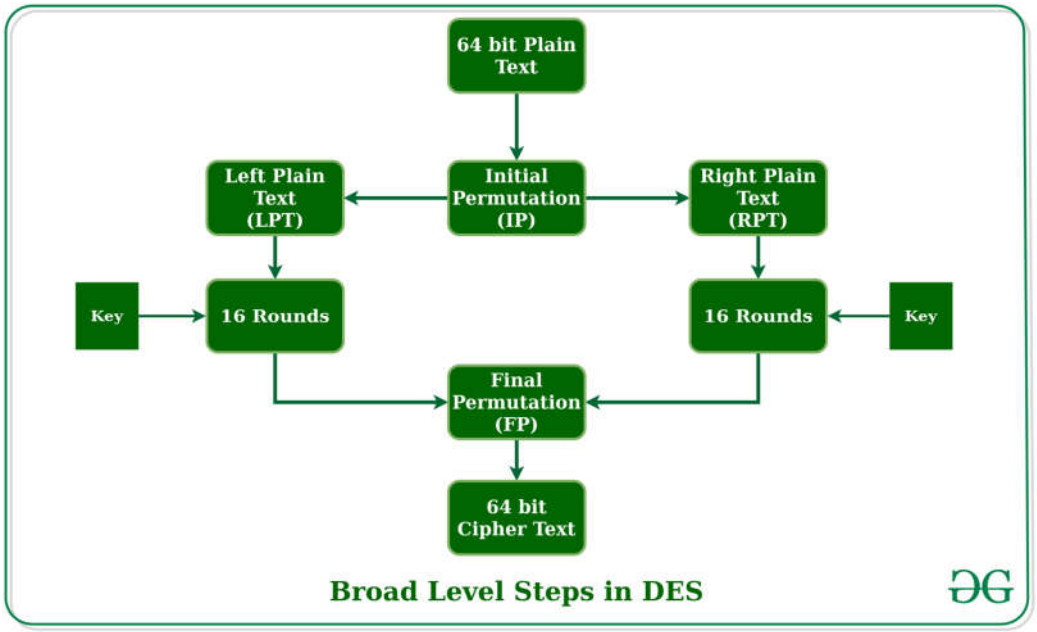
- This course provides students with knowledge of cryptographic algorithms and how they are used in today's world.
- The content emphasizes the principles, topics, approaches, and problem solving related to the underlying technologies and architectures of the field.

4.2 CONTENTS

4.2.1 Basic knowledge

DES is a block cipher that processes each block of plaintext information of a specified length of 64 bits. Before entering the 16 main cycles, the data block to be secured will be split into 64-bit blocks, and each of these 64-bit blocks will in turn be put into 16 rounds of DES encryption for execution. DES encrypt are follow steps:

- In the first step, the 64 bit plain text block is handed over to an initial Permutation (IP) function.
- The initial permutation performed on plain text.
- Next the initial permutation (IP) produces two halves of the permuted block; says Left Plain Text (LPT) and Right Plain Text (RPT).
- Now each LPT and RPT to go through 16 rounds of encryption process.
- In the end, LPT and RPT are rejoined and a Final Permutation (FP) is performed on the combined block
- The result of this process produces 64 bit cipher text.



4.2.2 DES Encrypt

Write a program to encrypt and decrypt text with Caesar encryption algorithm. The program can perform the following functions:

- Allow text input into the system.
- Allows entering text protection keys.
- Allows to write File and open File.

❖ **Step 1: Design Form:**


PROGRAM DES CIPHER
Input Key:
information
Encrypt
Open File
Write File
Plain Text:
I LOVE YOU MORE THAN I CAN SEE
Cipher Text:
Decrypt
All Show

❖ **Step 2: Write code for initialization function:**

```
public class bl_frmDES extends javax.swing.JFrame {

    public bl_frmDES() {
        initComponents();
    }

    private int mode;
    private static  void doCopy(InputStream is,OutputStream os) throws IOException{
        byte[] bytes = new byte[64];
        int numBytes;
        while((numBytes=is.read(bytes))!=-1)
        {
            os.write(bytes,0,numBytes);
        }
        os.flush();
        os.close();
        is.close();
    }
```

```
public static void encrypt(String key, InputStream is, OutputStream os) throws Throwable{
    encryptOrDecrypt(key, Cipher.ENCRYPT_MODE,is, os);
}

public static void decrypt(String key,InputStream is, OutputStream os) throws Throwable{
    encryptOrDecrypt(key,Cipher.ENCRYPT_MODE,is, os);
}

public static void encryptOrDecrypt(String key,int mode,InputStream is,OutputStream os) throws Throwable{
    DESKeySpec dks= new DESKeySpec(key.getBytes());
    SecretKeyFactory skf= SecretKeyFactory.getInstance("DES");
    SecretKey desKey= skf.generateSecret(dks);
    Cipher cipher= Cipher.getInstance("DES");

    if (mode==Cipher.ENCRYPT_MODE) {
        cipher.init(Cipher.ENCRYPT_MODE, desKey);
        CipherInputStream cis = new CipherInputStream(is, cipher);
        doCopy(cis, os);
    } else if (mode == Cipher.DECRYPT_MODE) {
        cipher.init(Cipher.DECRYPT_MODE, desKey);
        CipherOutputStream cos= new CipherOutputStream(os, cipher);
        doCopy(is, cos);
    }

}
```

❖ **Step 3: Write an event handler function:**

3.1 Button Encrypt:

```
private void btnEncryptActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        String key = txtKey.getText();
        FileInputStream fis= new FileInputStream("D:\\Des.txt");
        FileOutputStream fos= new FileOutputStream("D:\\EnDes.txt");
        encrypt(key, fis, fos);
        JOptionPane.showMessageDialog(null, "Encrypted");
    } catch (Throwable e) {
        e.printStackTrace();
    }
}
```

3.2 Button Decrypt:

```
private void btnDecryptActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    FileInputStream fis2 = null;
    try {
            String key = txtKey.getText();
            fis2= new FileInputStream("D:\\Des.txt");
            FileOutputStream fos2 = new FileOutputStream("D:\\EnDes.txt");
            decrypt(key, fis2, fos2);
            BufferedReader br=null;

            br=new BufferedReader(new FileReader("D:\\Des.txt"));
            StringBuffer sb=new StringBuffer();
                JOptionPane.showMessageDialog(null, "Dencrypted");
            char[] ca=new char[5];
            while (br.ready())
        {
               int len=br.read(ca);
               sb.append(ca,0,len);

        }
```

```
            br.close();
            System.out.println("Data is :"+" "+sb);
            String chuoi=sb.toString();
            txtPlainText.setText(chuoi);
        } catch (Throwable ex) {
            Logger.getLogger(bl_frmDES.class.getName()).log(Level.SEVERE,null,ex);
        } finally{
            try {
                fis2.close();
            } catch (IOException ex) {
                Logger.getLogger(bl_frmDES.class.getName()).log(Level.SEVERE,null,ex);
            }
        }

}
```

3.3 Button Write File:

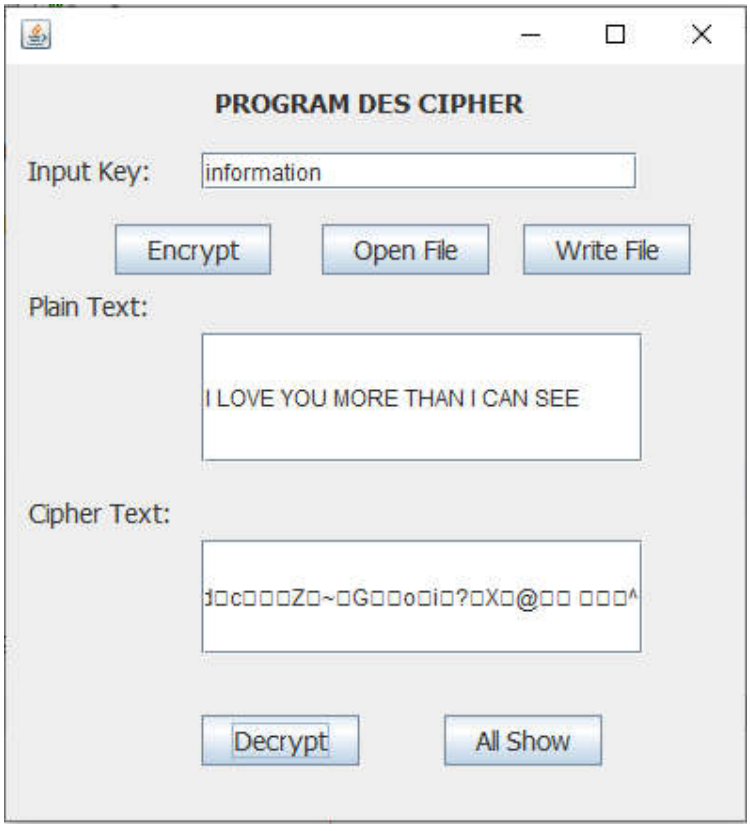
```
private void btnWriteFileActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        BufferedWriter bw=null;
        String fileName="D:\\Des.txt";
        String s = txtPlainText.getText();
        bw = new BufferedWriter(new FileWriter(fileName));
        bw.write(s);
        bw.close();
        JOptionPane.showMessageDialog(null, "Wrote File");
        txtCipherText.setText(s);
    } catch (IOException ex) {
        Logger.getLogger(bl_frmDES.class.getName()).log(Level.SEVERE,null,ex);
    }

}
```

3.4 Button Open File:

```
private void btnOpenFileActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        BufferedReader br=null;
        String fileName="D:\\EnDes.txt";
        br=new BufferedReader(new FileReader(fileName));
        StringBuffer sb= new StringBuffer();
        JOptionPane.showMessageDialog(null, "Opened file");
        char[] ca= new char[5];
        while (br.ready()) {
            int len=br.read(ca);
            sb.append(ca,0,len);
        }
        br.close();
        System.out.println("Data is :"+" "+sb);
        String chuoi=sb.toString();
        txtPlainText.setText(chuoi);
    } catch (IOException ex) {
        Logger.getLogger(bl_frmDES.class.getName()).log(Level.SEVERE,null,ex);
    }
}
```

Result:

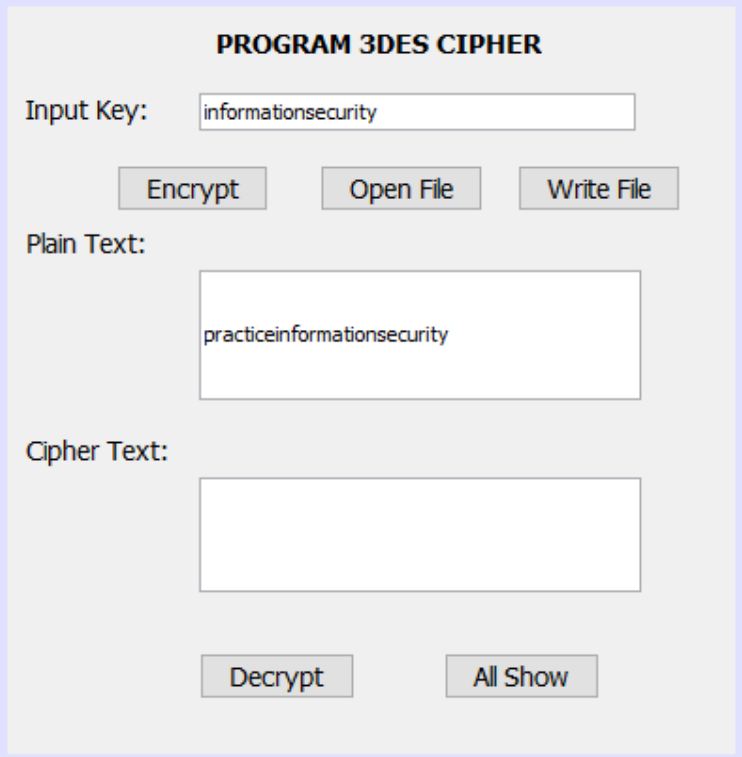


4.2.3 3DES Encrypt

Write a program to encrypt and decrypt text with 3DES encryption algorithm. The program can perform the following functions:

- Allow text input into the system.
- Allows entering text protection keys.
- Allows to write File and open File.

❖ **Step 1: Design Form:**



❖ **Step 2: Write code for initialization function:**

```
public class b2_frm3DES extends javax.swing.JFrame {


    public b2_frm3DES() {
        initComponents();
    }
    private static final String UNICODE_FORMAT="UTF8";
    public  static  final String DESEDE_ENCYPTION_SCHEME ="DESede";
    private KeySpec myKeySpec;
    private SecretKeyFactory mySecretKeyFactory;
    private Cipher cipher;
    byte[] keyAsBytes;
    private String myEncryptionKey;
    private String myEncryptionScheme;
    SecretKey key;
```

❖ **Step 2: Write code for Encrypt function:**

```
public String encrypt(String unencryptedString)
{
    String encryptedString = null;
    try {
        cipher.init(Cipher.ENCRYPT_MODE, key);
        byte[] plainText = unencryptedString.getBytes(UNICODE_FORMAT);
        byte[] encryptedText = cipher.doFinal(plainText);
        BASE64Encoder base64encoder= new BASE64Encoder();
        encryptedString = base64encoder.encode(encryptedText);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return  encryptedString;
}
```

❖ **Step 3: Write code for Encrypt function:**

```
public String decrypt(String encryptedString)
{
    String decryptedText = null;
    try {
        cipher.init(Cipher.DECRYPT_MODE, key);
        BASE64Decoder base64decoder = new BASE64Decoder();
        byte[] encryptedText = base64decoder.decodeBuffer(encryptedString);
        byte[] plainText = cipher.doFinal(encryptedText);
        String a= new String(plainText);
        System.out.println("plainText :"+ a);
        decryptedText = a;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return decryptedText;
}
```

❖ **Step 4: Write an event handler function:**

4.1 Button Encrypt:

```
private void btnEncryptActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        myEncryptionKey = txtKey.getText();
        myEncryptionScheme = DESEDE_ENCYPTION_SCHEME;
        keyAsBytes = myEncryptionKey.getBytes(UNICODE_FORMAT);
        myKeySpec = new DESedeKeySpec(keyAsBytes);
        mySecretKeyFactory = SecretKeyFactory.getInstance(myEncryptionScheme);
        cipher = Cipher.getInstance(myEncryptionScheme);
        key = mySecretKeyFactory.generateSecret(myKeySpec);
        System.out.println("Key k :"+" "+key);
        String plainText=txtPalinText.getText();
        String encrypted = encrypt(plainText);
        System.out.println("Encrypted Value :"+ encrypted);
        txtCipherText.setText(encrypted);
    } catch (Exception ex) {
        ex.printStackTrace();
    }


}
```

3.2 Button Decrypt:

```
private void btbDecryptActionPerformed(java.awt.event.ActionEvent evt) {
        try {
        myEncryptionKey = txtKey.getText();
        myEncryptionScheme = DESEDE_ENCYPTION_SCHEME;
        keyAsBytes = myEncryptionKey.getBytes(UNICODE_FORMAT);
        myKeySpec = new DESedeKeySpec(keyAsBytes);
        mySecretKeyFactory = SecretKeyFactory.getInstance(myEncryptionScheme);
        cipher = Cipher.getInstance(myEncryptionScheme);
        key = mySecretKeyFactory.generateSecret(myKeySpec);
        System.out.println("Key k :"+" "+key);
        String plainText=txtPalinText.getText();
        String encrypted = decrypt(plainText);
        System.out.println("Decrypted Value :"+ encrypted);
        txtCipherText.setText(encrypted);
    } catch (Exception ex) {
        ex.printStackTrace();
    }


}
```

3.3 Button Write File:

```
private void btnWriteFileActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        BufferedWriter bw=null;
        String fileName="D:\\3Des.txt";
        String s = txtPalinText.getText();
        bw = new BufferedWriter(new FileWriter(fileName));
        bw.write(s);
        bw.close();
        JOptionPane.showMessageDialog(null, "Wrote file");
        txtCipherText.setText(s);
    } catch (IOException ex) {
        Logger.getLogger(b2_frm3DES.class.getName()).log(Level.SEVERE,null,ex);
    }
}
```

3.4 Button Open File:

s

```
private void btnOpenFileActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        BufferedReader br=null;
        String fileName="D:\\3Des.txt";
        br=new BufferedReader(new FileReader(fileName));
        StringBuffer sb= new StringBuffer();
        JOptionPane.showMessageDialog(null, "Opened file");
        char[] ca= new char[5];
        while (br.ready()) {
            int len=br.read(ca);
            sb.append(ca,0,len);
        }
        br.close();
        System.out.println("Data is :"+ "+sb);
        String chuoi=sb.toString();
        txtPalinText.setText(chuoi);
    } catch (IOException ex) {
        Logger.getLogger(b2_frm3DES.class.getName()).log(Level.SEVERE,null,ex);
    }

}
```

Result: