

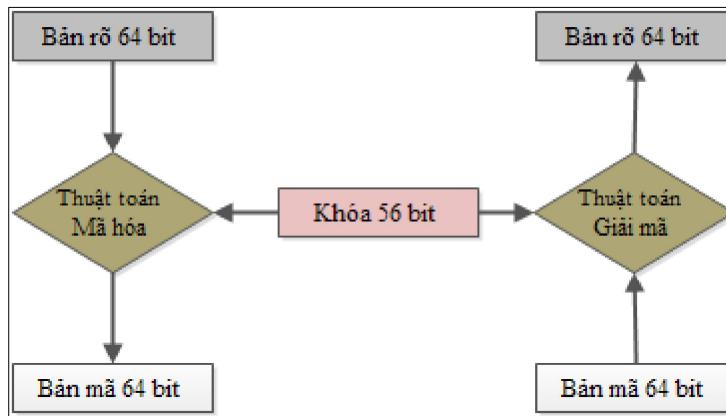
BÀI 2. MÃ HÓA ĐỔI XỨNG HIỆN ĐẠI

2.1 MẬT MÃ DES

2.1.1 Giới thiệu:

DES (Data Encryption Standard) là chuẩn mã hóa dữ liệu đầu tiên trên thế giới, do Cơ quan an ninh Quốc gia Hoa Kỳ (NSA) đề xuất trên cơ sở cải tiến thuật toán Lucifer do hãng IBM công bố năm 1973. DES đã được sử dụng rộng rãi ở Hoa Kỳ và nhiều quốc gia khác trong các thập kỷ 70, 80, 90 cho đến khi được thay thế bởi Tiêu chuẩn mã hóa dữ liệu tiên tiến AES (Advanced Encryption Standard) vào năm 2002.

Đầu vào của DES là khối 64 bit, đầu ra cũng là khối 64 bit. Khóa mã hóa có độ dài 56 bit, nhưng thực chất ban đầu là 64 bit, được lấy đi các bit ở vị trí chia hết cho 8 dùng để kiểm tra tính chẵn lẻ.

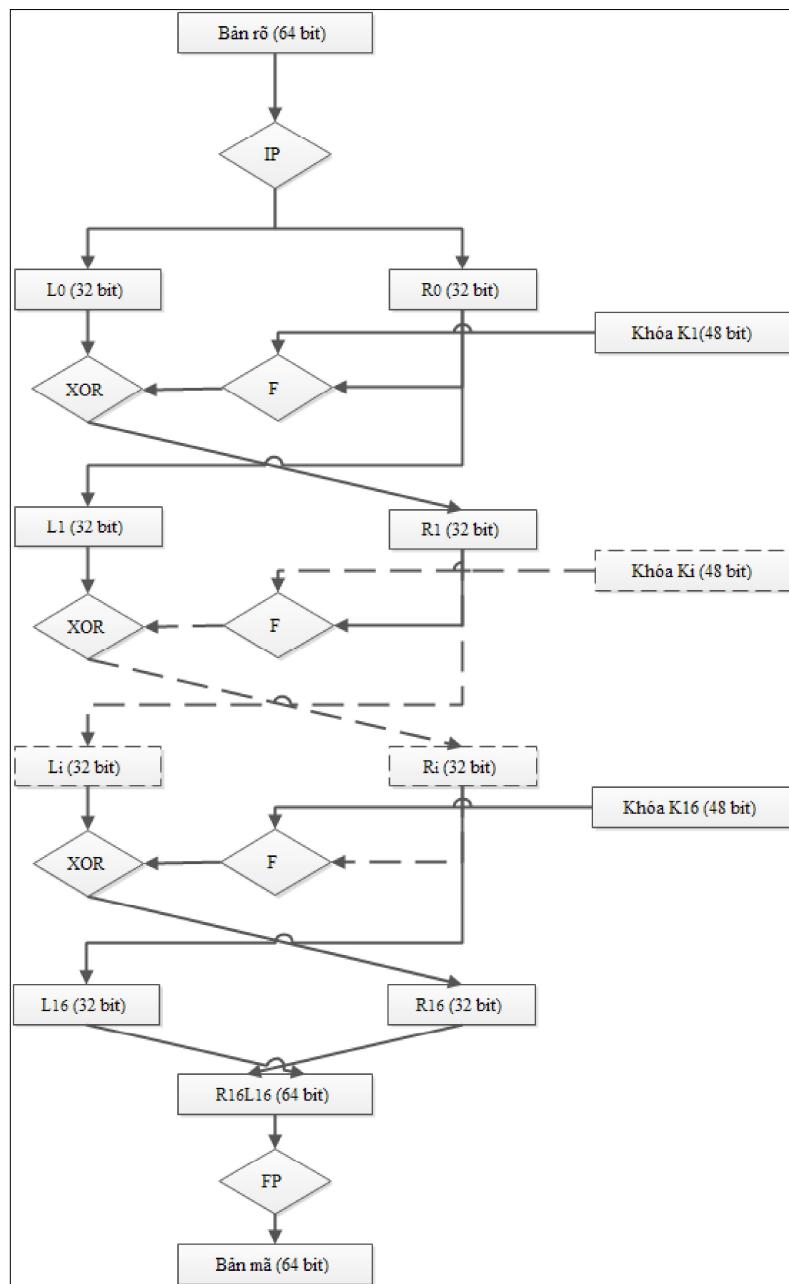


DES là thuật toán mã hóa theo khối, nó xử lý từng khối thông tin của bản rõ có độ dài xác định là 64 bit. Trước khi đi vào 16 chu trình chính, khối dữ liệu cần bảo mật sẽ được tách ra thành từng khối 64 bit, và từng khối 64 bit này sẽ lần lượt được đưa vào 16 vòng mã hóa DES để thực hiện. Input: Bản rõ $M = m_1m_2...m_{64}$ là một khối 64 bit, khóa 64 bit $K = k_1k_2...k_{64}$. Output: Bản mã 64 bit $C = c_1c_2...c_{64}$.

Các bước thuật toán hoạt động như sau:

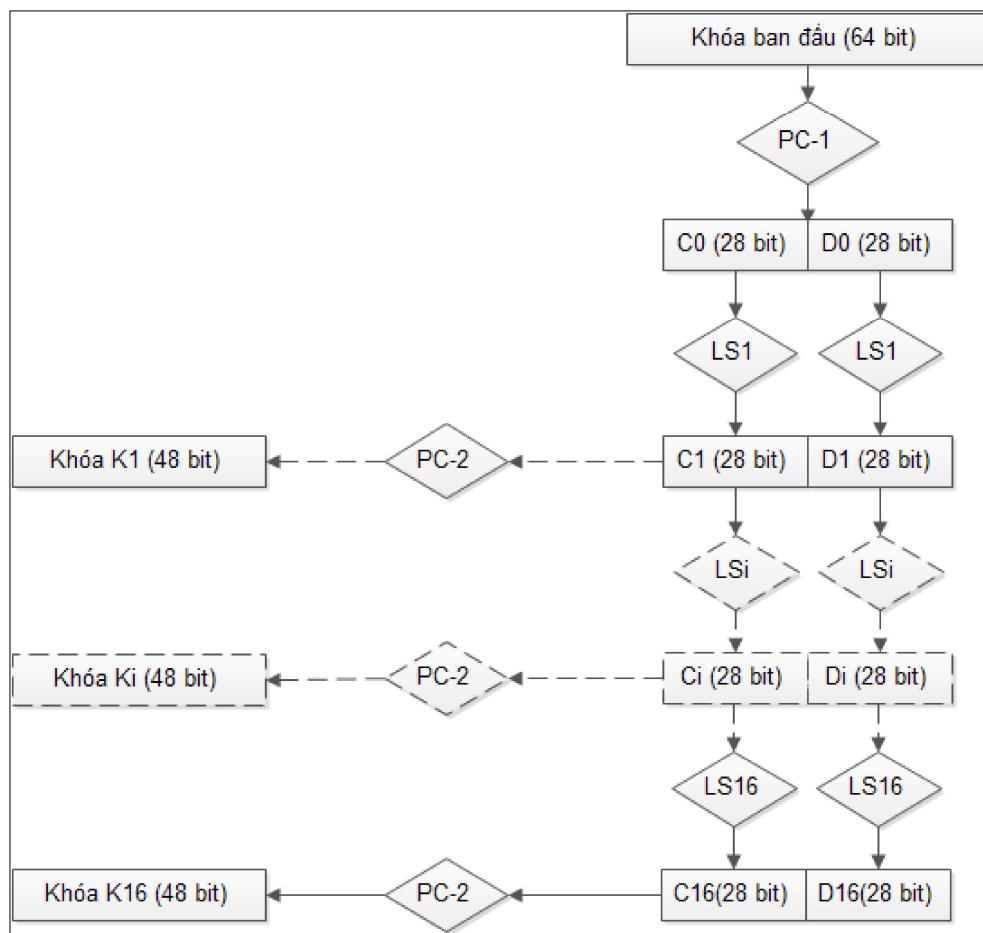
- Bước 1: Sinh khóa con: Sử dụng thuật toán sinh khóa con từ khóa K ta sẽ được 16 khóa con K_1, K_2, \dots, K_{16} .

- Bước 2: Sử dụng phép hoán vị khởi đầu IP (Initial Permutation) để hoán vị các bit của M, kết quả nhận được chia thành 2 nửa là $L_0 = m_{63}m_{62}\dots m_{32}$, $R_0 = m_{31}m_{30}\dots m_0$.
- Bước 3: Với i chạy từ $i = 1$ đến 16 thực hiện: Tính các L_i và R_i theo công thức: $L_i = R_{i-1}$ $R_i = L_{i-1} \text{ XOR } f(R_{i-1}, K_i)$ trong đó $f(R_{i-1}, K_i) = R(S(E(R_{i-1}) \text{ XOR } K_i))$.
- Bước 4: Đổi vị trí khối L_{16} , R_{16} ta được khối $R_{16}L_{16} = b_1b_2\dots b_{64}$.
- Bước 5: Sử dụng phép hoán vị kết thúc FP (Final Permutation – nghịch đảo với hoán vị khởi đầu IP) ta thu được bản mã cần tìm : $C = IP^{-1}(b_1b_2\dots b_{64})$.



Quá trình sinh khóa con:

- 16 vòng lặp của DES chạy cùng thuật toán như nhau nhưng với 16 khóa con khác nhau. Các khóa con đều được sinh ra từ khóa chính của DES bằng thuật toán sinh khóa con. Khóa ban đầu là 1 xâu có độ dài 64 bit, bit thứ 8 của mỗi byte sẽ được lấy ra để kiểm tra phát hiện lỗi, tạo ra chuỗi 56 bit.
- Sau khi bỏ các bit kiểm tra ta sẽ hoán vị chuỗi 56 bit này. Hai bước trên được thực hiện thông qua hoán vị ma trận PC-1 (Permuted choice 1).
- Tiếp theo ta kết quả sau khi PC-1 thành 2 phần : C0 : 28 bit đầu. D0 : 28 bit cuối. Mỗi phần sẽ được xử lý 1 cách độc lập. $C_i = LS_i(C_{i-1})$ $D_i = LS_i(D_{i-1})$ với $1 \leq i \leq 16$. LS_i là biểu diễn phép dịch bit vòng (cyclic shift) sang trái 1 hoặc 2 vị trí tùy thuộc vào i.
- Cuối cùng sử dụng hoán vị cố định PC-2 (Permuted choice 2) để hoán vị chuỗi CiDi 56 bit tạo thành khóa K_i với 48 bit.

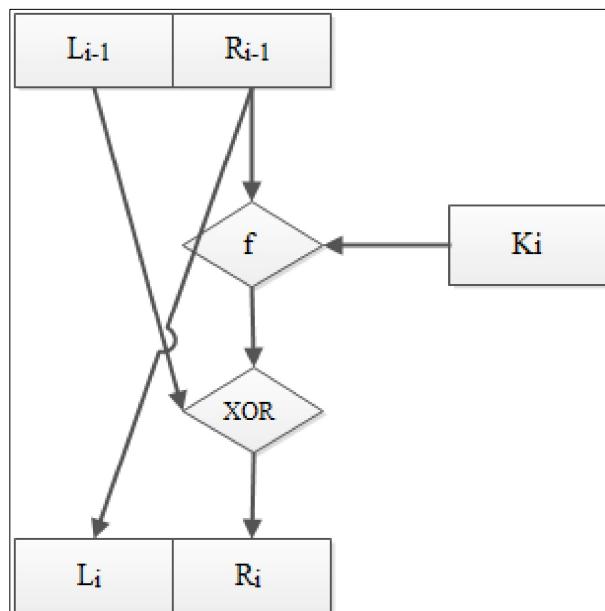


Quá trình mã hóa DES:

- Phase 01: Với bản rõ cho trước x , 1 xâu x' sẽ được tạo ra bằng cách hoán vị các bit của x theo hoán vị ban đầu IP. Tiếp theo x' sẽ được chia thành 2 phần L_0, R_0 . $x' = \text{IP}(x) = L_0R_0$ Trong đó L_0 là 32 bit đầu, R_0 là 32 bit cuối.

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

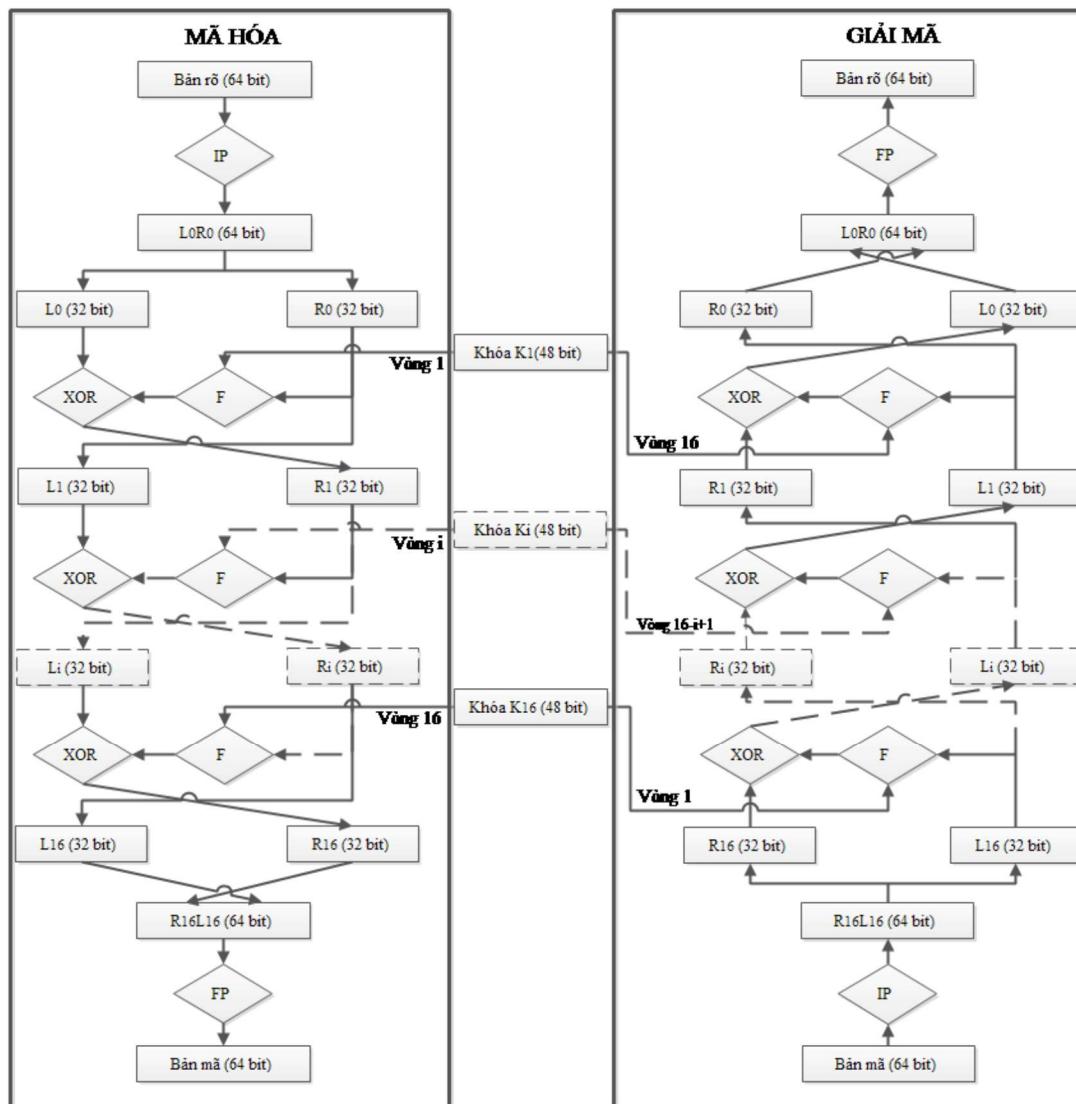
- Phase 02: Tính toán 16 lần bằng 1 hàm xác định. Ta sẽ tính L_i, R_i ($1 \leq i \leq 16$) theo quy tắc: $L_i = R_{i-1}$. $R_i = L_{i-1} \text{ XOR } f(R_{i-1}, K_i)$. Với K_i là khóa được sinh ra ở quá trình tạo khóa.

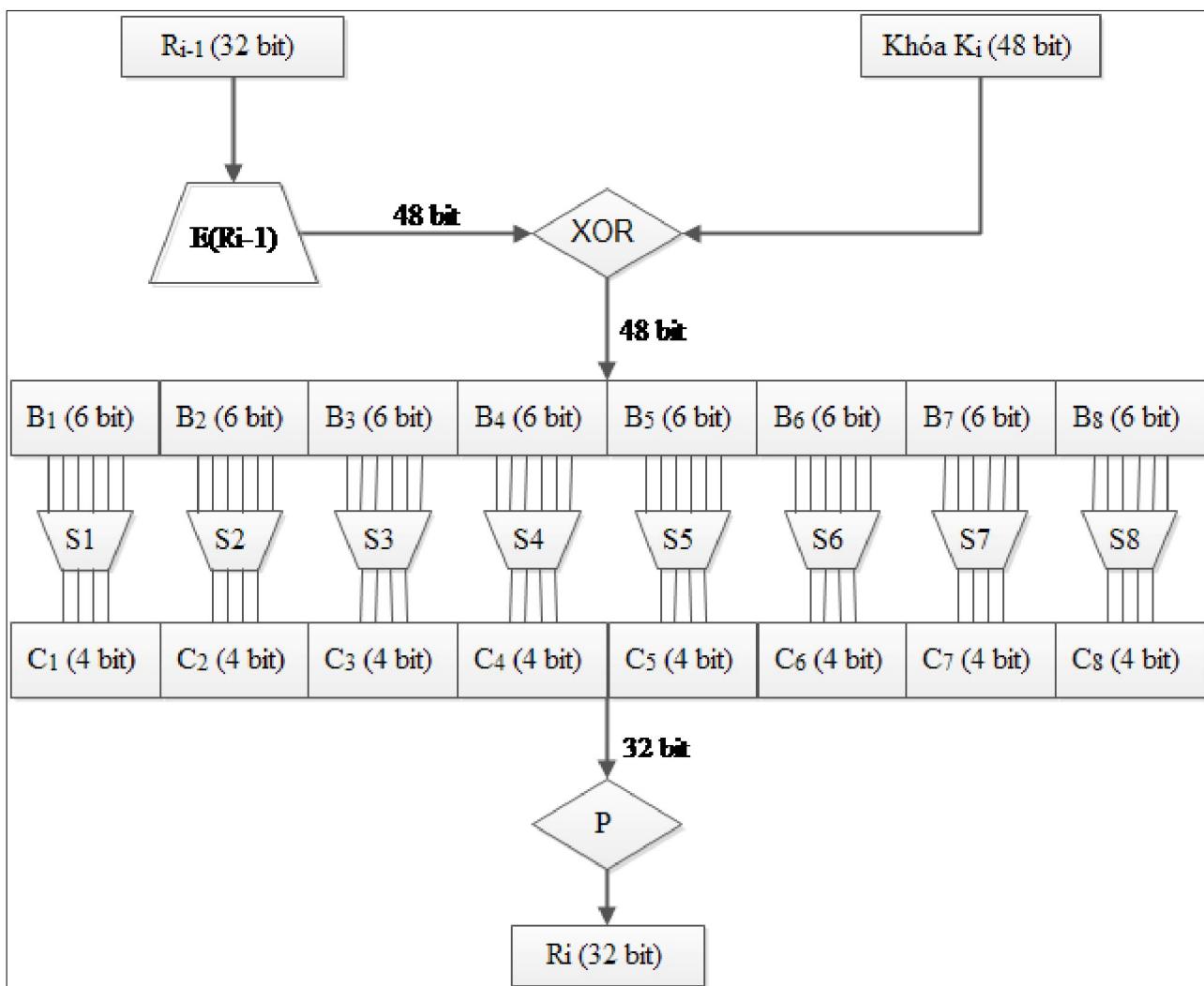


- Phase 03: Áp dụng hoán vị kết thúc FP cho xâu bit $R_{16}L_{16}$ ta thu được bản mã y : $y = \text{FP}(R_{16}L_{16})$.

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	59	17	57	25

Quá trình giải mã DES: Quá trình giải mã của DES cũng tương tự quá trình mã hóa. Chỉ khác nhau ở: $L_i = R_{i-1}$. $R_i = L_{i-1} \text{ XOR } f(R_{i-1}, K_{16-i+1})$. Như vậy khóa K của hàm F sẽ đi từ khóa K_{16} đến khóa K_1 .



Hàm F:**2.1.2 Bài tập thực hành:**

- Tạo mới package “week_02”. Trong package “week_02” tạo class “DESCipher.java”:

```
package week_02;

import javax.crypto.*;
import javax.crypto.spec.DESKeySpec;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.spec.InvalidKeySpecException;
import java.util.Base64;
```

```

public class DESCipher {

    private static final String ALGORITHM = "DES";

    public static String encrypt(String plaintext, String secretKey)
        throws NoSuchAlgorithmException,
        InvalidKeySpecException, InvalidKeySpecException, NoSuchPaddingException,
        BadPaddingException, IllegalBlockSizeException {

        SecretKey key = generateKey(secretKey);
        Cipher cipher = Cipher.getInstance(ALGORITHM);
        cipher.init(Cipher.ENCRYPT_MODE, key);
        byte[] encryptedBytes = cipher.doFinal(plaintext.getBytes());
        return Base64.getEncoder().encodeToString(encryptedBytes);
    }
}

```

```

public static String decrypt(String ciphertext, String secretKey)
    throws NoSuchAlgorithmException,
    InvalidKeySpecException, InvalidKeySpecException, NoSuchPaddingException,
    BadPaddingException, IllegalBlockSizeException {

    SecretKey key = generateKey(secretKey);
    Cipher cipher = Cipher.getInstance(ALGORITHM);
    cipher.init(Cipher.DECRYPT_MODE, key);
    byte[] decodedBytes = Base64.getDecoder().decode(ciphertext);
    byte[] decryptedBytes = cipher.doFinal(decodedBytes);
    return new String(decryptedBytes);
}

private static SecretKey generateKey(String secretKey)
    throws NoSuchAlgorithmException, InvalidKeySpecException {
    DESKeySpec keySpec = new DESKeySpec(secretKey.getBytes());
    SecretKeyFactory keyFactory = SecretKeyFactory.getInstance(ALGORITHM);
    return keyFactory.generateSecret(keySpec);
}
}

```

- Tạo Jframe Form “frm_DES.java”:



- Thêm các gói cần thiết:

```
package week_02;

import javax.swing.*;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
```

- Phương thức xử lý sự kiện nhấn nút “Encrypt”:

```
private void btn_encryptActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        String plaintext = txt_plaintext.getText();
        String secretKey = txt_key.getText();

        String encryptedText = DESCipher.encrypt(plaintext, secretKey);
        txt_ciphertext.setText(encryptedText);
    } catch (Exception e) {
        JOptionPane.showMessageDialog(this, "Error: " + e.getMessage(),
            "Error", JOptionPane.ERROR_MESSAGE);
    }
}
```

- Phương thức xử lý sự kiện nhấn nút “Decrypt”:

```

private void btn_decryptActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        String ciphertext = txt_ciphertext.getText();
        String secretKey = txt_key.getText();

        String decryptedText = DESCipher.decrypt(ciphertext, secretKey);
        txt_plaintext.setText(decryptedText);
    } catch (Exception e) {
        JOptionPane.showMessageDialog(this, "Error: " + e.getMessage(),
            "Error", JOptionPane.ERROR_MESSAGE);
    }
}

```

- Phương thức xử lý sự kiện nhấp nút “Save File”:

```

private void btn_saveFileActionPerformed(java.awt.event.ActionEvent evt) {
    String ciphertext = txt_ciphertext.getText();
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setDialogTitle("Save Ciphertext to File");
    int userSelection = fileChooser.showSaveDialog(this);
    if (userSelection == JFileChooser.APPROVE_OPTION) {
        File fileToSave = fileChooser.getSelectedFile();
        try (FileWriter writer = new FileWriter(fileToSave.getAbsolutePath() + ".txt")) {
            writer.write(ciphertext);
            JOptionPane.showMessageDialog(this, "Ciphertext saved to file successfully.",
                "Success", JOptionPane.INFORMATION_MESSAGE);
        } catch (IOException e) {
            JOptionPane.showMessageDialog(this, "Error saving file: " + e.getMessage(),
                "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
}

```

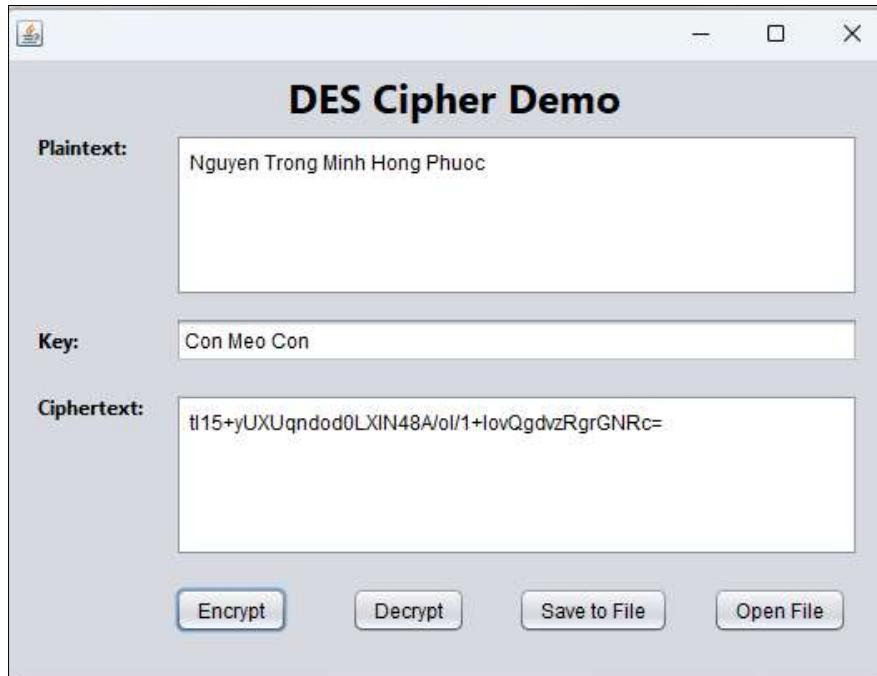
- Phương thức xử lý sự kiện nhấp nút “Open File”:

```

private void btn_openFileActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setDialogTitle("Open File containing Ciphertext");
    int userSelection = fileChooser.showOpenDialog(this);
    if (userSelection == JFileChooser.APPROVE_OPTION) {
        File fileToOpen = fileChooser.getSelectedFile();
        try (BufferedReader reader = new BufferedReader(new FileReader(fileToOpen))) {
            StringBuilder ciphertextBuilder = new StringBuilder();
            String line;
            while ((line = reader.readLine()) != null) {
                ciphertextBuilder.append(line);
            }
            String ciphertext = ciphertextBuilder.toString().trim();
            txt_ciphertext.setText(ciphertext);
        } catch (IOException e) {
            JOptionPane.showMessageDialog(this, "Error opening file: " + e.getMessage(),
                "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
}

```

- Kiểm tra ứng dụng: Chuột phải tại file cần chạy, chọn “Run File” hoặc nhấn tổ hợp phím “Shift + F6”.
- Kiểm tra lần lượt các nút “Encrypt”, “Decrypt”, “Save File”, “Open File”.



2.2 MẬT MÃ TRIPLEDES

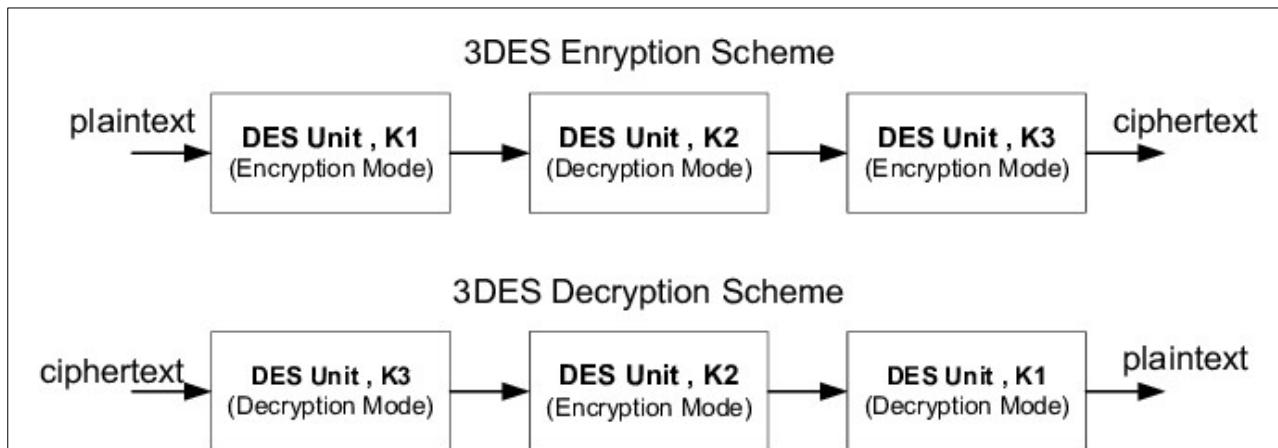
2.2.1 Giới thiệu:

Mật mã 3DES (Triple Data Encryption Standard) là một thuật toán mã hóa cung cấp mức độ bảo mật cao hơn so với DES (Data Encryption Standard) ban đầu. DES đã bị coi là không an toàn do chiều dài khóa ngắn (56 bit), và 3DES được phát triển để cải thiện điều này bằng cách sử dụng ba vòng mã hóa DES để tăng độ dài khóa lên 168 bit ($168 = 3 * 56$).

Cách hoạt động của 3DES:

- Khóa (Keying):
 - + 3DES sử dụng ba khóa DES độc lập (K_1, K_2, K_3), mỗi khóa có độ dài 56 bit, tạo thành một khóa chung dài 168 bit.
 - + Có hai cách sắp xếp các khóa 3DES:

- Cách sắp xếp 1 (EDE1): Sử dụng cùng một khóa để mã hoá và giải mã: $E(K1, D(K2, E(K3, plaintext)))$.
 - Cách sắp xếp 2 (EDE2): Sử dụng khóa khác nhau cho mã hoá và giải mã: $E(K1, D(K2, E(K1, plaintext)))$.
- Mã hoá và giải mã:
- + Mã hoá (Encryption): Plaintext được mã hoá bằng cách áp dụng ba vòng mã hóa DES lần lượt với các khóa K1, K2, K3.
 - + Giải mã (Decryption): Ciphertext được giải mã bằng cách áp dụng ba vòng giải mã DES ngược lại với các khóa K3, K2, K1.
- Đặc điểm nổi bật:
- + 3DES có độ bảo mật cao hơn so với DES ban đầu nhờ vào độ dài khóa lớn hơn và sử dụng thuật toán DES trong ba vòng.
 - + Nó vẫn được sử dụng rộng rãi trong các hệ thống và ứng dụng bảo mật, đặc biệt trong các môi trường yêu cầu mức độ bảo mật cao và yêu cầu tương thích với hệ thống sử dụng DES cũ.



3DES là một trong những thuật toán mã hóa phổ biến trong lĩnh vực bảo mật và vẫn được sử dụng rộng rãi mặc dù đã có các thuật toán mã hóa mới như AES (Advanced Encryption Standard) có hiệu suất và độ bảo mật cao hơn.

2.2.2 Bài tập thực hành:

- Trong package “week_02” tạo class “TripleDESCipher.java”:

```
package week_02;

import javax.crypto.*;
import javax.crypto.spec.SecretKeySpec;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.Base64;

public class TripleDESCipher {

    private static final String ALGORITHM = "DESede";

    public static String encrypt(String plaintext, String secretKey)
        throws NoSuchAlgorithmException,
        InvalidKeyException, NoSuchPaddingException,
        BadPaddingException, IllegalBlockSizeException {

        SecretKey key = generateKey(secretKey);
        Cipher cipher = Cipher.getInstance(ALGORITHM);
        cipher.init(Cipher.ENCRYPT_MODE, key);
        byte[] encryptedBytes = cipher.doFinal(plaintext.getBytes());
        return Base64.getEncoder().encodeToString(encryptedBytes);
    }
}
```

```
public static String decrypt(String ciphertext, String secretKey)
    throws NoSuchAlgorithmException,
    InvalidKeyException, NoSuchPaddingException,
    BadPaddingException, IllegalBlockSizeException {

    SecretKey key = generateKey(secretKey);
    Cipher cipher = Cipher.getInstance(ALGORITHM);
    cipher.init(Cipher.DECRYPT_MODE, key);
    byte[] decodedBytes = Base64.getDecoder().decode(ciphertext);
    byte[] decryptedBytes = cipher.doFinal(decodedBytes);
    return new String(decryptedBytes);
}
```

```

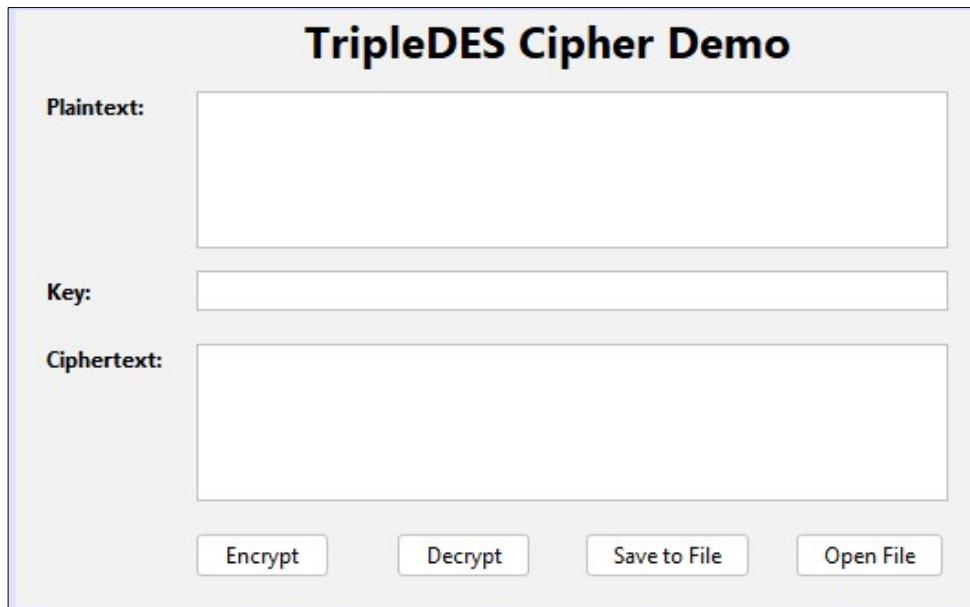
private static SecretKey generateKey(String secretKey)
    throws NoSuchAlgorithmException {
    // Key must be 24 bytes (192 bits) long for Triple DES
    byte[] keyBytes = secretKey.getBytes();
    byte[] validKeyBytes = new byte[24];

    // Truncate or pad keyBytes to make it 24 bytes long
    for (int i = 0; i < validKeyBytes.length; i++) {
        if (i < keyBytes.length) {
            validKeyBytes[i] = keyBytes[i];
        } else {
            validKeyBytes[i] = 0; // Or any other value as padding
        }
    }

    SecretKeySpec keySpec = new SecretKeySpec(validKeyBytes, ALGORITHM);
    return keySpec;
}
}

```

- Tạo Jframe Form “frm_TripleDES.java”:



- Thêm các gói cần thiết:

```
package week_02;

import javax.swing.*;
import java.io.*;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.spec.InvalidKeySpecException;
import javax.crypto.BadPaddingException;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
```

- Phương thức xử lý sự kiện nhấp nút “Encrypt”:

```
private void btn_encryptActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        String plaintext = txt_plaintext.getText();
        String secretKey = txt_key.getText();

        if (secretKey.length() == 24) {
            String encryptedText = TripleDESCipher.encrypt(plaintext, secretKey);
            txt_ciphertext.setText(encryptedText);
        } else {
            JOptionPane.showMessageDialog(this, "Secret Key must be 24 characters long.",
                "Error", JOptionPane.ERROR_MESSAGE);
        }
    } catch (InvalidKeyException | NoSuchAlgorithmException | BadPaddingException
        | IllegalBlockSizeException | NoSuchPaddingException e) {
        JOptionPane.showMessageDialog(this, "Error: " + e.getMessage(),
            "Error", JOptionPane.ERROR_MESSAGE);
    }
}
```

- Phương thức xử lý sự kiện nhấp nút “Decrypt”:

```
private void btn_decryptActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        String ciphertext = txt_ciphertext.getText();
        String secretKey = txt_key.getText();

        if (secretKey.length() == 24) {
            String decryptedText = TripleDESCipher.decrypt(ciphertext, secretKey);
            txt_plaintext.setText(decryptedText);
        } else {
            JOptionPane.showMessageDialog(this, "Secret Key must be 24 characters long.",
                "Error", JOptionPane.ERROR_MESSAGE);
        }
    } catch (InvalidKeyException | NoSuchAlgorithmException | BadPaddingException
        | IllegalBlockSizeException | NoSuchPaddingException e) {
        JOptionPane.showMessageDialog(this, "Error: " + e.getMessage(),
            "Error", JOptionPane.ERROR_MESSAGE);
    }
}
```

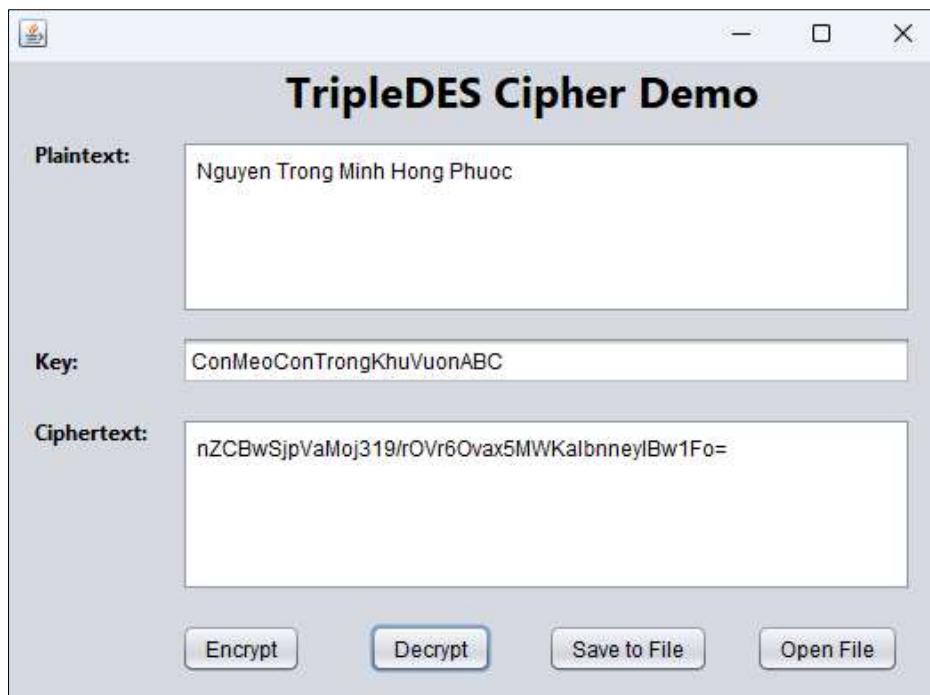
- Phương thức xử lý sự kiện nhấn nút “Save File”:

```
private void btn_saveFileActionPerformed(java.awt.event.ActionEvent evt) {
    String ciphertext = txt_ciphertext.getText();
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setDialogTitle("Save Ciphertext to File");
    int userSelection = fileChooser.showSaveDialog(this);
    if (userSelection == JFileChooser.APPROVE_OPTION) {
        File fileToSave = fileChooser.getSelectedFile();
        try (FileWriter writer = new FileWriter(fileToSave.getAbsolutePath() + ".txt")) {
            writer.write(ciphertext);
            JOptionPane.showMessageDialog(this, "Ciphertext saved to file successfully.",
                "Success", JOptionPane.INFORMATION_MESSAGE);
        } catch (IOException e) {
            JOptionPane.showMessageDialog(this, "Error saving file: " + e.getMessage(),
                "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
}
```

- Phương thức xử lý sự kiện nhấn nút “Open File”:

```
private void btn_openFileActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setDialogTitle("Open File containing Ciphertext");
    int userSelection = fileChooser.showOpenDialog(this);
    if (userSelection == JFileChooser.APPROVE_OPTION) {
        File fileToOpen = fileChooser.getSelectedFile();
        try (BufferedReader reader = new BufferedReader(new FileReader(fileToOpen))) {
            StringBuilder ciphertextBuilder = new StringBuilder();
            String line;
            while ((line = reader.readLine()) != null) {
                ciphertextBuilder.append(line);
            }
            String ciphertext = ciphertextBuilder.toString().trim();
            txt_ciphertext.setText(ciphertext);
        } catch (IOException e) {
            JOptionPane.showMessageDialog(this, "Error opening file: " + e.getMessage(),
                "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
}
```

- Kiểm tra ứng dụng: Chuột phải tại file cần chạy, chọn “Run File” hoặc nhấn tổ hợp phím “Shift + F6”.
- Kiểm tra lần lượt các nút “Encrypt”, “Decrypt”, “Save File”, “Open File”.



2.3 MẬT MÃ AES

2.3.1 Giới thiệu:

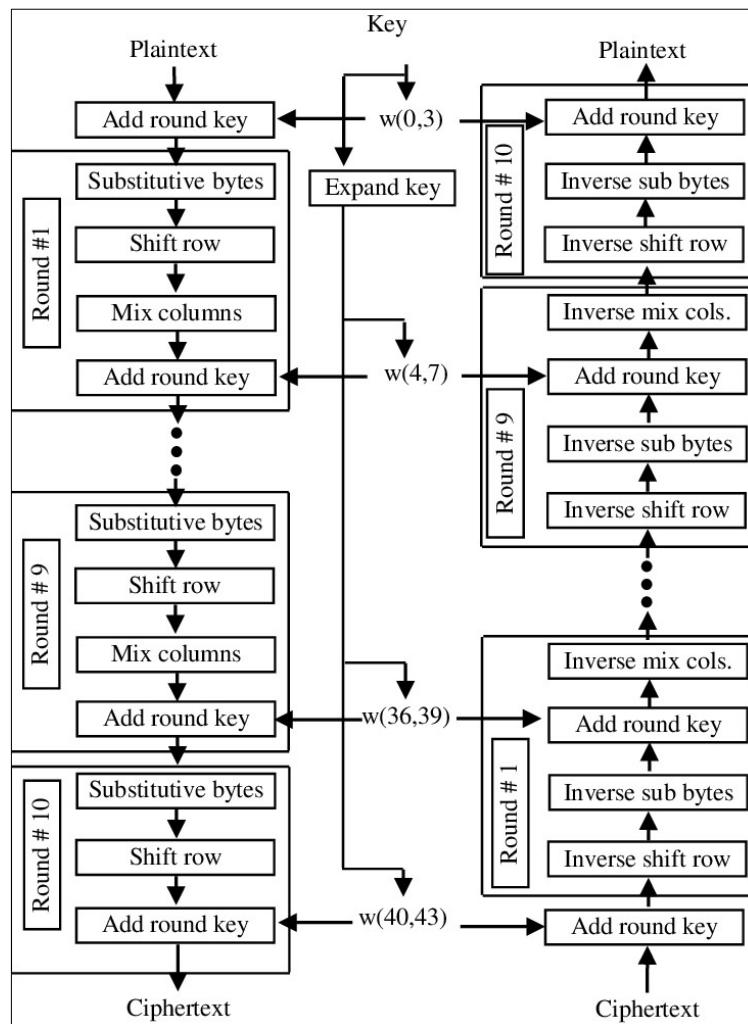
Vào những năm 1990, nhận thấy nguy cơ của mã hóa DES là kích thước khóa ngắn, có thể bị phá mã trong tương lai gần, Cục tiêu chuẩn quốc gia Hoa Kỳ đã kêu gọi xây dựng một phương pháp mã hóa mới. Cuối cùng một thuật toán có tên là Rijndael được chọn và đổi tên thành Advanced Encryption Standard hay AES. Có thể nói rằng mã hóa AES với khóa có kích thước 256 bit là an toàn tuyệt đối.

- Đặc điểm: An ninh hơn và nhanh hơn 3DES.
- Kích thước khối: 128 bit.
- Kích thước khóa: 128/192/256 bit.
- Số vòng: 10/12/14.
- Cấu trúc mạng S-P, nhưng không theo hệ Feistel.

AES là một thuật toán mã hóa khối (block cipher) được sử dụng rộng rãi trong các ứng dụng mật mã hóa và bảo mật thông tin. Nó được chọn làm chuẩn mã hóa bởi Chính

phủ Hoa Kỳ và đã trở thành một trong những thuật toán mã hóa phổ biến nhất trên thế giới. Cách hoạt động của AES:

- **Khởi tạo:** Bắt đầu với một khóa được chọn và chuẩn bị dữ liệu cần mã hóa thành các khối cố định kích thước 128-bit.
- **SubBytes:** Mỗi byte trong khối dữ liệu được thay thế bằng một byte khác thông qua một bảng thay thế (S-box).
- **ShiftRows:** Các hàng trong khối dữ liệu được dịch chuyển sang trái theo một số lượng byte xác định.
- **MixColumns:** Các cột trong khối dữ liệu được kết hợp thông qua các phép nhân ma trận.
- **AddRoundKey:** Mỗi khối dữ liệu được kết hợp với một khóa con được tạo từ khóa chính.



2.3.2 Bài tập thực hành:

- Trong package “week_02” tạo class “AESCipher.java”:

```
package week_02;

import javax.crypto.*;
import javax.crypto.spec.SecretKeySpec;
import java.io.*;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.Base64;

public class AESCipher {

    private static final String ALGORITHM = "AES";
    private static final String ENCRYPTION_KEY = "encryptionKey"; // Key for AES encryption (16 bytes)

    public static String encrypt(String plaintext, String secretKey) throws NoSuchAlgorithmException,
            InvalidKeyException, NoSuchPaddingException,
            BadPaddingException, IllegalBlockSizeException {

        SecretKey key = generateKey(secretKey);
        Cipher cipher = Cipher.getInstance(ALGORITHM);
        cipher.init(Cipher.ENCRYPT_MODE, key);
        byte[] encryptedBytes = cipher.doFinal(plaintext.getBytes());
        return Base64.getEncoder().encodeToString(encryptedBytes);
    }
}
```

```
public static String decrypt(String ciphertext, String secretKey) throws NoSuchAlgorithmException,
        InvalidKeyException, NoSuchPaddingException,
        BadPaddingException, IllegalBlockSizeException {

    SecretKey key = generateKey(secretKey);
    Cipher cipher = Cipher.getInstance(ALGORITHM);
    cipher.init(Cipher.DECRYPT_MODE, key);
    byte[] decodedBytes = Base64.getDecoder().decode(ciphertext);
    byte[] decryptedBytes = cipher.doFinal(decodedBytes);
    return new String(decryptedBytes);
}

private static SecretKey generateKey(String secretKey) throws NoSuchAlgorithmException {
    byte[] keyBytes = secretKey.getBytes();
    SecretKeySpec keySpec = new SecretKeySpec(keyBytes, ALGORITHM);
    return keySpec;
}
```

```

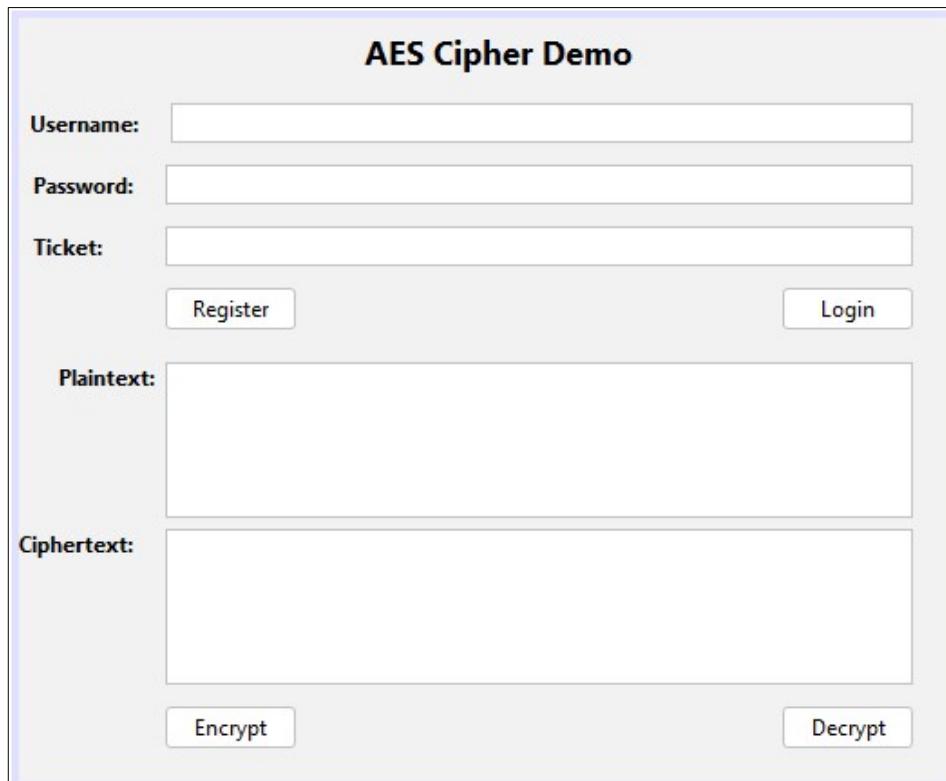
public static String generateRegistrationKey(String username, String password) {
    String registrationKey = username + ":" + password + ":" + ENCRYPTION_KEY;
    return registrationKey;
}

public static void saveRegistrationKeyToFile(String registrationKey, String filename)
    throws IOException {
    try (FileOutputStream fos = new FileOutputStream(filename);
        ObjectOutputStream oos = new ObjectOutputStream(fos)) {
        oos.writeObject(registrationKey);
    }
}

public static String readRegistrationKeyFromFile(String filename)
    throws IOException, ClassNotFoundException {
    try (FileInputStream fis = new FileInputStream(filename);
        ObjectInputStream ois = new ObjectInputStream(fis)) {
        return (String) ois.readObject();
    }
}
}

```

- Tạo Jframe Form "frm_AES.java":



- Thêm các gói cần thiết và khởi tạo:

```

package week_02;

import javax.swing.*;
import java.io.IOException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import javax.crypto.BadPaddingException;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;

public class frm_AES extends javax.swing.JFrame {

    private static final String REGISTRATION_FILE = "registration.key";
    private boolean isLoggedIn = false;

    public frm_AES() {
        initComponents();
        updateButtonState();
    }
}

```

- Phương thức cập nhật trạng thái của các nút:

```

private void updateButtonState() {
    btn_encrypt.setEnabled(isLoggedIn);
    btn_decrypt.setEnabled(isLoggedIn);
}

```

- Phương thức xử lý sự kiện cho nút “Register”:

```

private void btn_registerActionPerformed(java.awt.event.ActionEvent evt) {
    String username = txt_username.getText();
    String password = new String(txt_password.getPassword());

    if (!username.isEmpty() && !password.isEmpty()) {
        try {
            String registrationKey =
                AESCipher.generateRegistrationKey(username, password);
            AESCipher.saveRegistrationKeyToFile(registrationKey, REGISTRATION_FILE);
            JOptionPane.showMessageDialog(this,
                "Registration Successful. Registration Key saved.",
                "Success", JOptionPane.INFORMATION_MESSAGE);
        } catch (IOException e) {
            JOptionPane.showMessageDialog(this,
                "Error saving registration key: " + e.getMessage(),
                "Error", JOptionPane.ERROR_MESSAGE);
        }
    } else {
        JOptionPane.showMessageDialog(this, "Username and Password cannot be empty.",
            "Error", JOptionPane.ERROR_MESSAGE);
    }
}

```

- Phương thức xử lý sự kiện cho nút “Login”:

```

private void btn_loginActionPerformed(java.awt.event.ActionEvent evt) {
    String username = txt_username.getText();
    String password = new String(txt_password.getPassword());

    if (!username.isEmpty() && !password.isEmpty()) {
        try {
            String registrationKeyFromFile =
                AESCipher.readRegistrationKeyFromFile(REGISTRATION_FILE);
            String registrationKey =
                AESCipher.generateRegistrationKey(username, password);

            if (registrationKeyFromFile.equals(registrationKey)) {
                JOptionPane.showMessageDialog(this,
                    "Login Successful with Registration Key: " + registrationKey,
                    "Success", JOptionPane.INFORMATION_MESSAGE);
                txt_ticket.setText(registrationKey);
                isLoggedIn = true;
                updateButtonState();
            } else {
                JOptionPane.showMessageDialog(this, "Invalid Username or Password.",
                    "Error", JOptionPane.ERROR_MESSAGE);
            }
        } catch (IOException | ClassNotFoundException e) {
            JOptionPane.showMessageDialog(this, "Error reading registration key: "
                + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        }
    } else {
        JOptionPane.showMessageDialog(this, "Username and Password cannot be empty.",
            "Error", JOptionPane.ERROR_MESSAGE);
    }
}

```

- Phương thức xử lý sự kiện cho nút “Encrypt”:

```

private void btn_encryptActionPerformed(java.awt.event.ActionEvent evt) {
    String plaintext = txt_plaintext.getText();
    String secretKey = new String(txt_password.getPassword());

    if (!plaintext.isEmpty() && !secretKey.isEmpty()) {
        try {
            String encryptedText = AESCipher.encrypt(plaintext, secretKey);
            txt_ciphertext.setText(encryptedText);
        } catch (InvalidKeyException | NoSuchAlgorithmException | BadPaddingException |
            IllegalBlockSizeException | NoSuchPaddingException e) {
            JOptionPane.showMessageDialog(this, "Error encrypting: " + e.getMessage(),
                "Error", JOptionPane.ERROR_MESSAGE);
        }
    } else {
        JOptionPane.showMessageDialog(this, "Input and Secret Key cannot be empty.",
            "Error", JOptionPane.ERROR_MESSAGE);
    }
}

```

- Phương thức xử lý sự kiện cho nút “Decrypt”:

```

private void btn_decryptActionPerformed(java.awt.event.ActionEvent evt) {
    String ciphertext = txt_ciphertext.getText();
    String secretKey = new String(txt_password.getPassword());

    if (!ciphertext.isEmpty() && !secretKey.isEmpty()) {
        try {
            String decryptedText = AESCipher.decrypt(ciphertext, secretKey);
            txt_plaintext.setText(decryptedText);
        } catch (InvalidKeyException | NoSuchAlgorithmException | BadPaddingException |
                 IllegalBlockSizeException | NoSuchPaddingException e) {
            JOptionPane.showMessageDialog(this, "Error decrypting: " + e.getMessage(),
                                         "Error", JOptionPane.ERROR_MESSAGE);
        }
    } else {
        JOptionPane.showMessageDialog(this, "Output and Secret Key cannot be empty.",
                                         "Error", JOptionPane.ERROR_MESSAGE);
    }
}

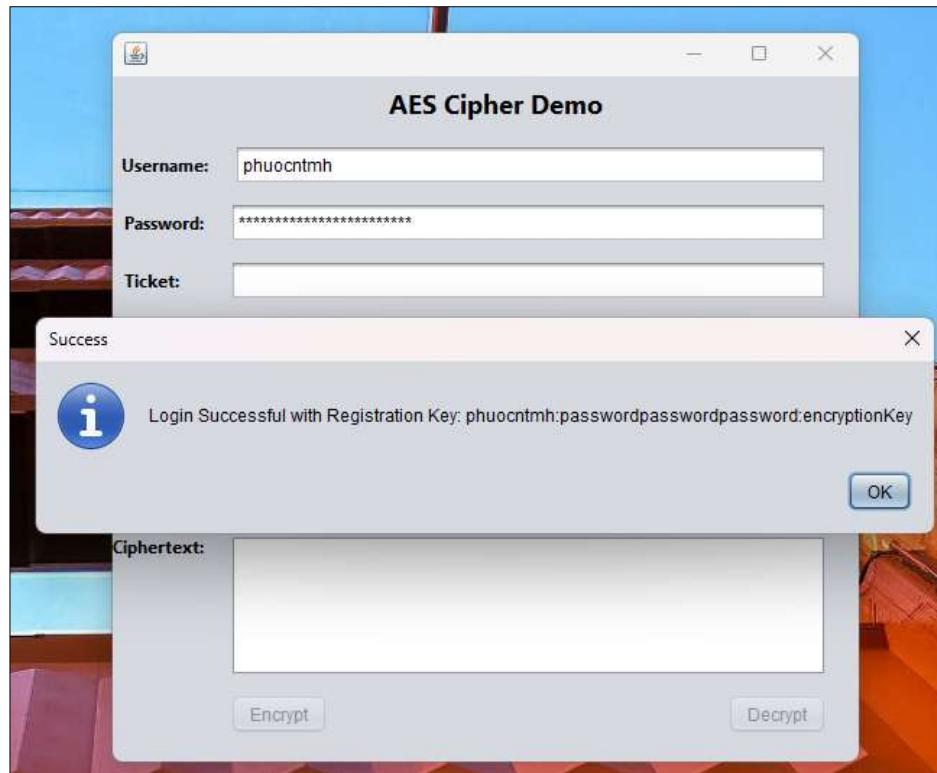
```

- Kiểm tra ứng dụng: Chuột phải tại file cần chạy, chọn “Run File” hoặc nhấn tổ hợp phím “Shift + F6”.
- Kiểm tra lần lượt các nút “Register”, “Login”, “Encrypt”, “Decrypt”.



- Khi đăng ký thành công, kiểm tra trong thư mục chứa dự án sẽ có file “registration.key”. File này lưu dữ liệu người dùng đã đăng ký.

- Khi đăng nhập thành công, sẽ tạo ra 1 ticket để dùng mã hoá và giải mã dữ liệu.



- Kiểm tra mã hoá và giải mã dữ liệu.



2.4 MẬT MÃ RC4

2.4.1 Giới thiệu:

RC4 (Rivest Cipher 4) là một thuật toán mã hoá dùng cho mục đích bảo mật dữ liệu. Nó được phát triển bởi Ronald Rivest vào năm 1987 và nhanh chóng trở thành một trong những thuật toán mã hoá phổ biến nhất trên thế giới.

Cơ chế hoạt động của RC4 có các đặc tính sau:

- Đơn vị mã hóa của RC4 là một byte 8 bít.
- Mảng S và T gồm 256 số nguyên 8 bít
- Khóa K là một dãy gồm N số nguyên 8 bít với N có thể lấy giá trị từ 1 đến 256.
- Bộ sinh số mỗi lần sinh ra một byte để sử dụng trong phép XOR.

Hai giai đoạn của RC4 là:

a) *Giai đoạn khởi tạo:*

```
/* Khoi tao day S va T*/
for i = 0 to 255 do
    S[i] = i;
    T[i] = K[i mod N];
next i
/* Hoan vi day S */
j = 0;
for i = 0 to 255 do
    j = (j + S[i] + T[i]) mod 256;
    Swap(S[i], S[j]);
next i
```

b) *Giai đoạn sinh số:*

```
i, j = 0;
while (true)
    i = (i + 1) mod 256;
    j = (j + S[i]) mod 256;
    Swap (S[i], S[j]);
    t = (S[i] + S[j]) mod 256;
    k = S[t];
end while;
```

Quá trình sinh số của RC4 cũng sinh ra dãy số ngẫu nhiên, khó đoán trước, vì vậy RC4 đạt được mức độ an toàn cao theo tinh thần của mã hóa OTP. Mã hóa RC4 hoàn

toàn được thực hiện trên các số nguyên một byte do đó tối ưu cho việc thiết lập bằng phần mềm và tốc độ thực hiện nhanh hơn so với mã khôi.

Cách hoạt động cụ thể của RC4:

- Khởi tạo S-box (S):
 - + Bước đầu tiên, RC4 sử dụng một khóa (key) để khởi tạo một hoán vị ban đầu của các byte từ 0 đến 255. Đây được gọi là S-box (hoặc S).
 - + Mỗi giá trị trong S-box là một số từ 0 đến 255, ban đầu được sắp xếp theo thứ tự từ 0 đến 255.
- Hoán vị S-box: Dựa trên khóa (key), S-box được hoán vị một cách nhanh chóng để tạo ra một trạng thái ban đầu.
- Sinh keystream:
 - + Sau khi hoán vị xong, RC4 sử dụng thuật toán để sinh ra một keystream (dãy byte ngẫu nhiên) từ S-box.
 - + Keystream là dãy byte dùng để mã hóa dữ liệu bằng phép XOR.

Mã hóa và giải mã:

- Khi mã hóa, mỗi byte của plaintext được XOR với từng byte tương ứng của keystream để tạo ra ciphertext.
- Khi giải mã, ciphertext được XOR với keystream để khôi phục lại plaintext.

2.4.2 Bài tập thực hành:

- Trong package “week_02” tạo class “RC4Cipher.java”:

```
package week_02;

public class RC4Cipher {

    private byte[] S = new byte[256];
    private byte[] T = new byte[256];
    private int keylen;
```

```

public RC4Cipher(byte[] key) {
    keylen = key.length;
    for (int i = 0; i < 256; i++) {
        S[i] = (byte) i;
        T[i] = key[i % keylen];
    }

    int j = 0;
    for (int i = 0; i < 256; i++) {
        j = (j + S[i] + T[i]) & 0xFF;
        swap(S, i, j);
    }
}

private void swap(byte[] arr, int i, int j) {
    byte temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}

```

```

public byte[] encrypt(byte[] plaintext) {
    byte[] ciphertext = new byte[plaintext.length];
    int i = 0, j = 0;

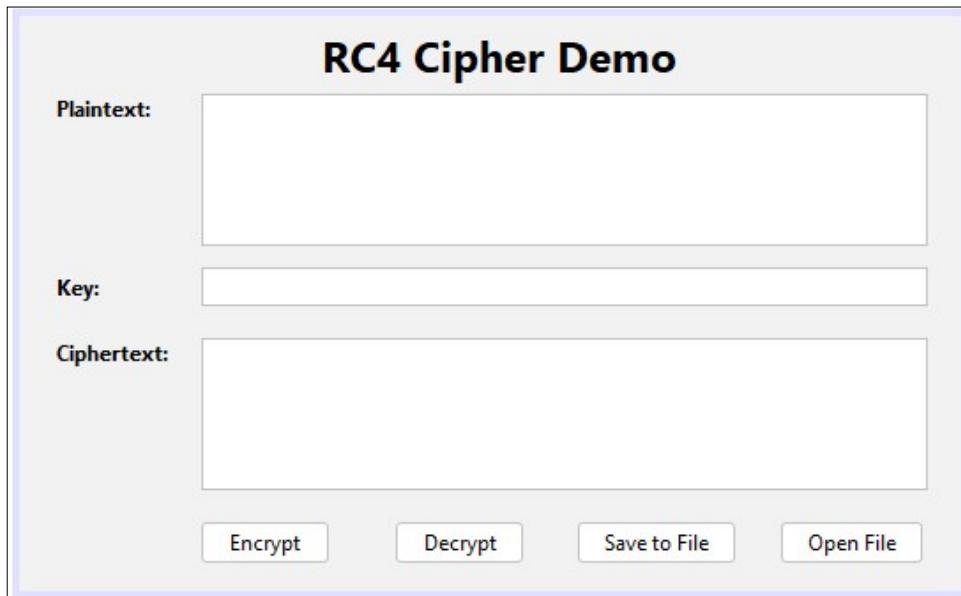
    for (int k = 0; k < plaintext.length; k++) {
        i = (i + 1) & 0xFF;
        j = (j + S[i]) & 0xFF;
        swap(S, i, j);
        int t = (S[i] + S[j]) & 0xFF;
        ciphertext[k] = (byte) (plaintext[k] ^ S[t]);
    }

    return ciphertext;
}

public byte[] decrypt(byte[] ciphertext) {
    return encrypt(ciphertext); // RC4 encryption and decryption are symmetric
}

```

- Tạo mới Jframe Form “frm_RC4.java”:



- Thêm các gói cần thiết và khởi tạo:

```
package week_02;

import javax.swing.*;
import java.io.*;
import java.nio.charset.StandardCharsets;

public class frm_RC4 extends javax.swing.JFrame {

    private RC4Cipher rc4;

    public frm_RC4() {
        initComponents();
    }
}
```

- Phương thức xử lý sự kiện cho nút “Encrypt”:

```
private void btn_encryptActionPerformed(java.awt.event.ActionEvent evt) {
    String plaintext = txt_plaintext.getText();
    byte[] plaintextBytes = plaintext.getBytes(StandardCharsets.UTF_8);
    String key = txt_key.getText();
    rc4 = new RC4Cipher(key.getBytes());
    byte[] ciphertextBytes = rc4.encrypt(plaintextBytes);
    String ciphertext = bytesToHexString(ciphertextBytes);
    txt_ciphertext.setText(ciphertext);
}
```

- Phương thức xử lý sự kiện cho nút “Decrypt”:

```

private void btn_decryptActionPerformed(java.awt.event.ActionEvent evt) {
    String ciphertext = txt_ciphertext.getText();
    byte[] ciphertextBytes = hexStringToByteArray(ciphertext);
    String key = txt_key.getText();
    rc4 = new RC4Cipher(key.getBytes());
    byte[] decryptedBytes = rc4.decrypt(ciphertextBytes);
    String decryptedText = new String(decryptedBytes, StandardCharsets.UTF_8);
    txt_plaintext.setText(decryptedText);
}

```

- Phương thức xử lý sự kiện cho nút “Save to File”:

```

private void btn_saveFileActionPerformed(java.awt.event.ActionEvent evt) {
    String ciphertext = txt_ciphertext.getText();
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setDialogTitle("Save Ciphertext to File");
    int userSelection = fileChooser.showSaveDialog(this);
    if (userSelection == JFileChooser.APPROVE_OPTION) {
        File fileToSave = fileChooser.getSelectedFile();
        try (FileWriter writer =
            new FileWriter(fileToSave.getAbsolutePath() + ".txt")) {
            writer.write(ciphertext);
            JOptionPane.showMessageDialog(this,
                "Ciphertext saved to file successfully.",
                "Success", JOptionPane.INFORMATION_MESSAGE);
        } catch (IOException e) {
            JOptionPane.showMessageDialog(this, "Error saving file: " + e.getMessage(),
                "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
}

```

- Phương thức xử lý sự kiện cho nút “Open File”:

```

private void btn_openFileActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setDialogTitle("Open File containing Ciphertext");
    int userSelection = fileChooser.showOpenDialog(this);
    if (userSelection == JFileChooser.APPROVE_OPTION) {
        File fileToOpen = fileChooser.getSelectedFile();
        try (BufferedReader reader =
            new BufferedReader(new FileReader(fileToOpen))) {
            StringBuilder ciphertextBuilder = new StringBuilder();
            String line;
            while ((line = reader.readLine()) != null) {
                ciphertextBuilder.append(line);
            }
        }
    }
}

```

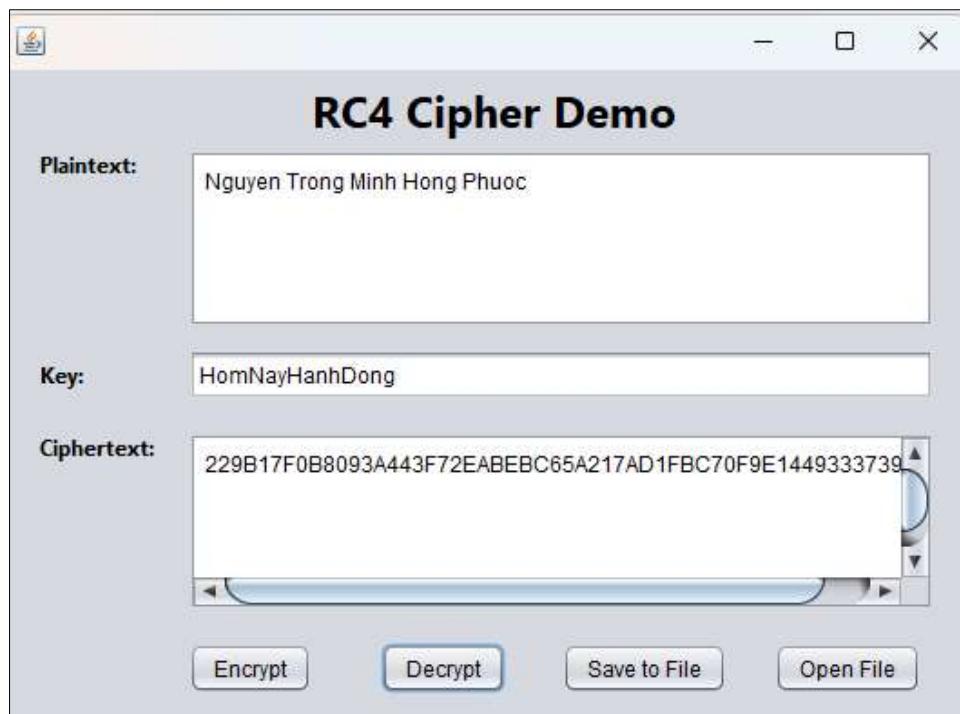
```
        string ciphertext = ciphertextBuilder.toString().trim();
        txt_ciphertext.setText(ciphertext);
    } catch (IOException e) {
        JOptionPane.showMessageDialog(this,
            "Error opening file: " + e.getMessage(),
            "Error", JOptionPane.ERROR_MESSAGE);
    }
}
```

- Các phương thức chuyển đổi ByteArray \leftrightarrow hexString:

```
private static byte[] hexStringToByteArray(String s) {
    int len = s.length();
    byte[] data = new byte[len / 2];
    for (int i = 0; i < len; i += 2) {
        data[i / 2] = (byte) ((Character.digit(s.charAt(i), 16) << 4)
            + Character.digit(s.charAt(i + 1), 16));
    }
    return data;
}

private static String bytesToHexString(byte[] bytes) {
    StringBuilder sb = new StringBuilder();
    for (byte b : bytes) {
        sb.append(String.format("%02X", b));
    }
    return sb.toString();
}
```

- Kiểm tra ứng dụng: Chuột phải tại file cần chạy, chọn “Run File” hoặc nhấn tổ hợp phím “Shift + F6”.
 - Kiểm tra lần lượt các nút “Encrypt”, “Decrypt”, “Save File”, “Open File”.



2.5 MẬT MÃ TWOFISH

2.5.1 Giới thiệu:

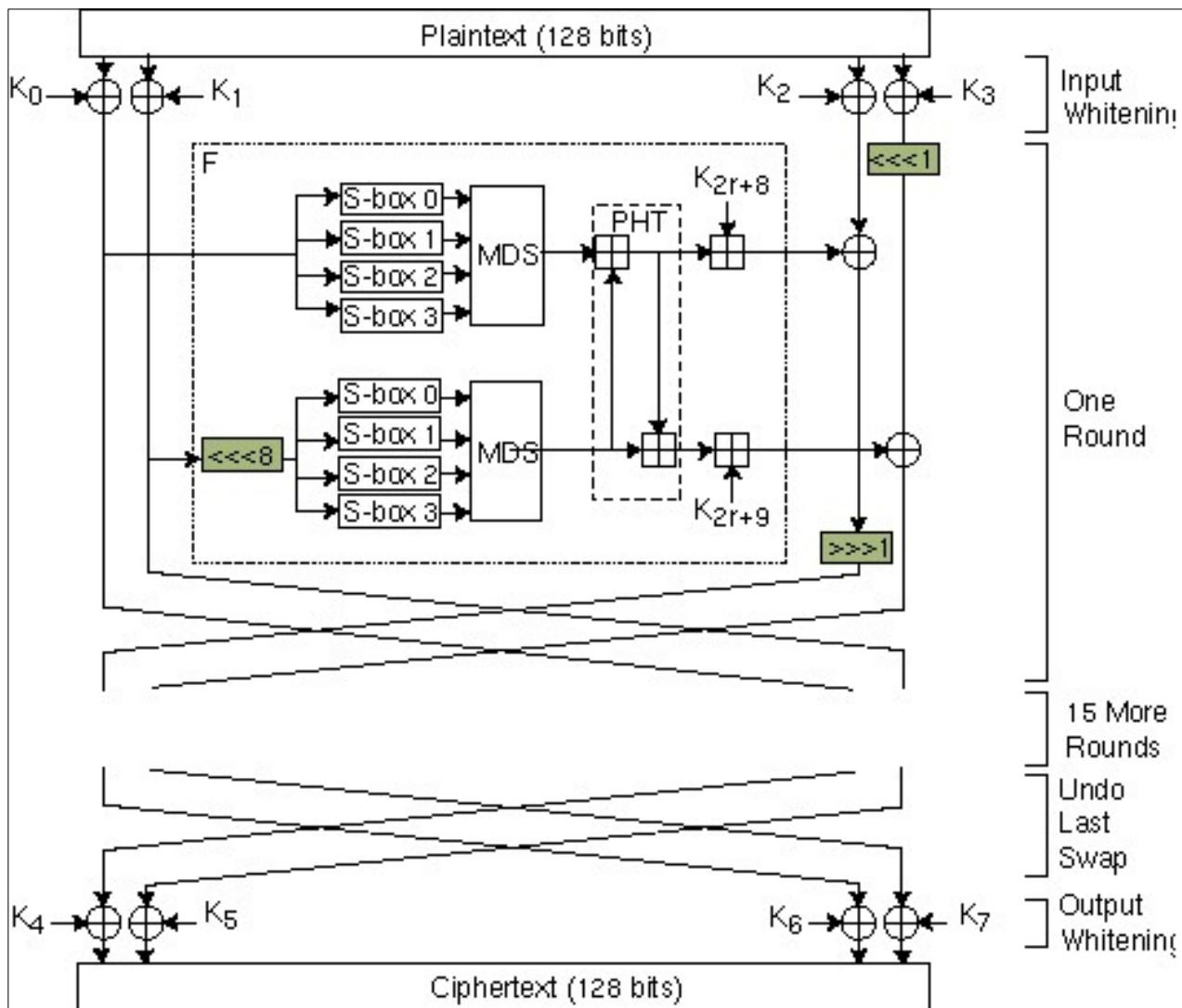
Twofish là một thuật toán mã hoá đối xứng (block cipher) được thiết kế để cung cấp mức độ bảo mật cao và hiệu suất tốt. Nó được phát triển bởi Bruce Schneier và đội ngũ của ông vào những năm 1990, và đã tham gia vào cuộc thi AES (Advanced Encryption Standard) để lựa chọn một thuật toán thay thế cho DES. Mặc dù không được chọn làm AES, Twofish vẫn được coi là một trong những lựa chọn mã hoá rất mạnh.

Cách hoạt động của Twofish:

- Kích thước block và khóa:
 - + Twofish mã hoá dữ liệu theo từng block có kích thước là 128 bit.
 - + Độ dài khóa có thể lên đến 256 bit, cho phép nó cung cấp mức độ bảo mật cao và khả năng chống lại các cuộc tấn công brute-force.
- Cấu trúc mạng Feistel: Twofish sử dụng một cấu trúc mạng Feistel với 16 vòng mã hoá. Cấu trúc này cho phép mã hoá và giải mã được thực hiện bằng cùng một quy trình.

- Phép XOR và hàm hộp S (S-boxes):
 - + Trong mỗi vòng mã hoá, Twofish sử dụng phép XOR và các hàm hộp S để biến đổi dữ liệu theo khóa và dữ liệu đầu vào.
 - + Các hàm hộp S được sử dụng để thay đổi các giá trị byte đầu vào thành các giá trị khác nhau, làm phức tạp hóa quá trình mã hoá.
- Khối hóa học (Mixing operations): Trong mỗi vòng, Twofish sử dụng các phép trộn (mixing operations) để cải thiện tính ngẫu nhiên của dữ liệu và đảm bảo tính bảo mật cao.

Twofish được coi là một trong những thuật toán mã hoá mạnh và đa dụng, với sự đa dạng hóa và phức tạp hóa quá trình mã hoá. Nó thường được sử dụng trong các ứng dụng yêu cầu mức độ bảo mật cao như phần mềm bảo mật, mạng và lưu trữ.



2.5.2 Bài tập thực hành:

- Để thực hiện bài tập này, chúng ta cần sử dụng thư viện “org.bouncycastle.crypto”. Đầu tiên mở file “pom.xml” và cập nhật các “dependencies” như sau:

```
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>22</maven.compiler.source>
    <maven.compiler.target>22</maven.compiler.target>
</properties>

<dependencies>
    <dependency>
        <groupId>org.bouncycastle</groupId>
        <artifactId>bcp prov-jdk15on</artifactId>
        <version>1.70</version>
    </dependency>
    <dependency>
        <groupId>org.bouncycastle</groupId>
        <artifactId>bcp kix-jdk15on</artifactId>
        <version>1.70</version>
    </dependency>
</dependencies>
</project>
```

- Sau đó, tiến hành “Clean anh Build” lại dự án để Maven tìm và tải xuống thư viện còn thiếu.
- Trong package “week_02” tạo class “TwofishCipher.java”:

```
package week_02;

import org.bouncycastle.crypto.BlockCipher;
import org.bouncycastle.crypto.engines.TwofishEngine;
import org.bouncycastle.crypto.modes.CBCBlockCipher;
import org.bouncycastle.crypto.paddings.PaddedBufferedBlockCipher;
import org.bouncycastle.crypto.params.KeyParameter;
import org.bouncycastle.crypto.params.ParametersWithIV;
import java.nio.charset.StandardCharsets;
```

```

public class TwofishCipher {

    private static final int BLOCK_SIZE = 16; // Twofish block size in bytes
    private BlockCipher cipher = new CBCBlockCipher(new TwofishEngine());

    public byte[] encrypt(String plaintext, byte[] key, byte[] iv) throws Exception {
        PaddedBufferedBlockCipher paddedCipher = new PaddedBufferedBlockCipher(cipher);
        ParametersWithIV parameters = new ParametersWithIV(new KeyParameter(key), iv);
        paddedCipher.init(true, parameters);

        byte[] plaintextBytes = plaintext.getBytes(StandardCharsets.UTF_8);
        int minSize = paddedCipher.getOutputSize(plaintextBytes.length);
        byte[] outBuf = new byte[minSize];
        int length1 = paddedCipher.processBytes(plaintextBytes, 0,
            plaintextBytes.length, outBuf, 0);
        int length2 = paddedCipher.doFinal(outBuf, length1);

        byte[] ciphertext = new byte[length1 + length2];
        System.arraycopy(outBuf, 0, ciphertext, 0, ciphertext.length);
        return ciphertext;
    }
}

```

```

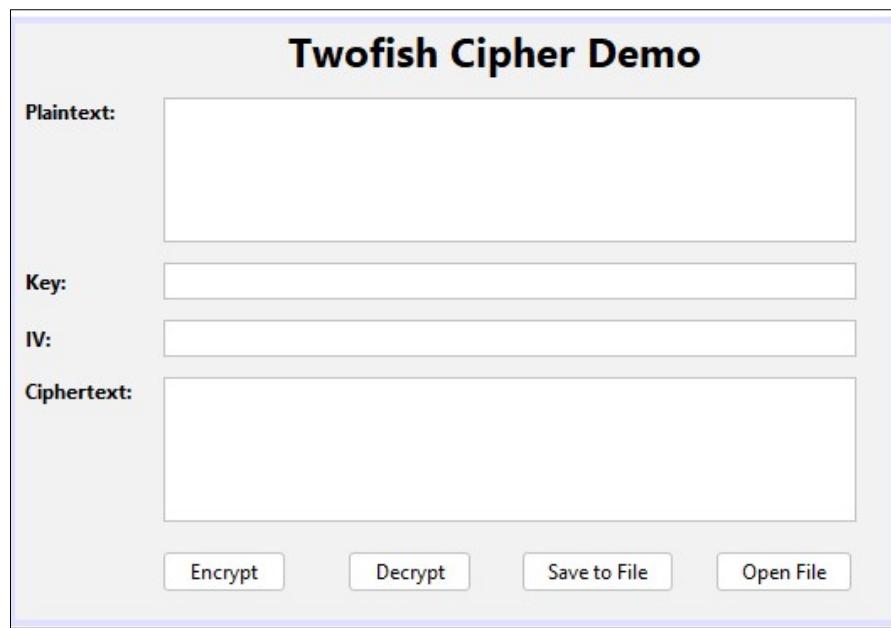
public String decrypt(byte[] ciphertext, byte[] key, byte[] iv) throws Exception {
    PaddedBufferedBlockCipher paddedCipher = new PaddedBufferedBlockCipher(cipher);
    ParametersWithIV parameters = new ParametersWithIV(new KeyParameter(key), iv);
    paddedCipher.init(false, parameters);

    int minSize = paddedCipher.getOutputSize(ciphertext.length);
    byte[] outBuf = new byte[minSize];
    int length1 = paddedCipher.processBytes(ciphertext, 0,
        ciphertext.length, outBuf, 0);
    int length2 = paddedCipher.doFinal(outBuf, length1);

    byte[] plaintextBytes = new byte[length1 + length2];
    System.arraycopy(outBuf, 0, plaintextBytes, 0, plaintextBytes.length);
    return new String(plaintextBytes, StandardCharsets.UTF_8);
}
}

```

- Tạo mới Jframe Form “frm_Twofish.java”:



- Thêm các gói cần thiết và khởi tạo:

```
package week_02;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.nio.charset.StandardCharsets;
import java.util.Base64;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
```

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

    TwofishCipher twofish = new TwofishCipher();

    public frm_Twofish() {
        initComponents();
    }
}
```

- Phương thức xử lý sự kiện cho nút "Encrypt":

```

private void btn_encryptActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        String plaintext = txt_plaintext.getText();
        byte[] key = txt_key.getText().getBytes(StandardCharsets.UTF_8);
        byte[] iv = txt_IV.getText().getBytes(StandardCharsets.UTF_8);

        byte[] ciphertext = twofish.encrypt(plaintext, key, iv);
        String encryptedText = Base64.getEncoder().encodeToString(ciphertext);

        txt_ciphertext.setText(encryptedText);
    } catch (UnsupportedEncodingException e) {
        JOptionPane.showMessageDialog(this, "Unsupported encoding: " + e.getMessage(),
                "Error", JOptionPane.ERROR_MESSAGE);
    } catch (Exception ex) {
        Logger.getLogger(frm_Twofish.class.getName()).log(Level.SEVERE, null, ex);
        JOptionPane.showMessageDialog(this, "Error encrypting: " + ex.getMessage(),
                "Error", JOptionPane.ERROR_MESSAGE);
    }
}

```

- Phương thức xử lý sự kiện cho nút “Decrypt”:

```

private void btn_decryptActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        String ciphertext = txt_ciphertext.getText();
        byte[] key = txt_key.getText().getBytes(StandardCharsets.UTF_8);
        byte[] iv = txt_IV.getText().getBytes(StandardCharsets.UTF_8);

        byte[] decodedCiphertext = Base64.getDecoder().decode(ciphertext);
        String decryptedText = twofish.decrypt(decodedCiphertext, key, iv);

        txt_plaintext.setText(decryptedText);
    } catch (Exception ex) {
        Logger.getLogger(frm_Twofish.class.getName()).log(Level.SEVERE, null, ex);
        JOptionPane.showMessageDialog(this, "Error decrypting: " + ex.getMessage(),
                "Error", JOptionPane.ERROR_MESSAGE);
    }
}

```

- Phương thức xử lý sự kiện cho nút “Save File”:

```

private void btn_saveFileActionPerformed(java.awt.event.ActionEvent evt) {
    String ciphertext = txt_ciphertext.getText();
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setDialogTitle("Save Ciphertext to File");
    int userSelection = fileChooser.showSaveDialog(this);
    if (userSelection == JFileChooser.APPROVE_OPTION) {
        File fileToSave = fileChooser.getSelectedFile();
        try (FileWriter writer =
                new FileWriter(fileToSave.getAbsolutePath() + ".txt")) {
            writer.write(ciphertext);
            JOptionPane.showMessageDialog(this,
                "Ciphertext saved to file successfully.", "Success",
                JOptionPane.INFORMATION_MESSAGE);
        } catch (IOException e) {
            JOptionPane.showMessageDialog(this, "Error saving file: " +
                e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
}

```

- Phương thức xử lý sự kiện cho nút “Open File”:

```

private void btn_openFileActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setDialogTitle("Open File containing Ciphertext");
    int userSelection = fileChooser.showOpenDialog(this);
    if (userSelection == JFileChooser.APPROVE_OPTION) {
        File fileToOpen = fileChooser.getSelectedFile();
        try (BufferedReader reader = new BufferedReader(new FileReader(fileToOpen))) {
            StringBuilder ciphertextBuilder = new StringBuilder();
            String line;
            while ((line = reader.readLine()) != null) {
                ciphertextBuilder.append(line);
            }
            String ciphertext = ciphertextBuilder.toString().trim();
            txt_ciphertext.setText(ciphertext);
        } catch (IOException e) {
            JOptionPane.showMessageDialog(this, "Error opening file: " +
                e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
}

```

- Kiểm tra ứng dụng: Chuột phải tại file cần chạy, chọn “Run File” hoặc nhấn tổ hợp phím “Shift + F6”.
- Kiểm tra lần lượt các nút “Encrypt”, “Decrypt”, “Save File”, “Open File”.



2.6 BÀI TẬP NÂNG CAO

- ❖ **Bài tập 01:** Viết một chương trình sử dụng AES với chế độ Counter (CTR) để mã hóa và giải mã văn bản rõ (plaintext) sử dụng một khóa (key) và một nonce (số hóa, số một lần) được chọn ngẫu nhiên. Hãy bao gồm cả việc sinh ra nonce và khóa trong chương trình.
- ❖ **Bài tập 02:** Viết một chương trình Python sử dụng AES trong chế độ CBC (Cipher Block Chaining) để mã hóa và giải mã một văn bản rõ (plaintext) có độ dài bất kỳ sử dụng một khóa (key) và một IV (Initialization Vector) được chọn ngẫu nhiên. Bao gồm cả việc sinh ra IV và khóa trong chương trình.