

BỘ GIÁO DỤC VÀ ĐÀO TẠO
ĐẠI HỌC CÔNG NGHỆ TP.HCM

THỰC HÀNH

BẢO MẬT THÔNG TIN

Biên soạn:

TS. Văn Thiên Hoàng

ThS. Dương Minh Chiến

www.hutech.edu.vn

THỰC HÀNH BẢO MẬT THÔNG TIN

Ấn bản 2020

*Các ý kiến đóng góp về tài liệu học tập này, xin gửi về e-mail của ban biên tập :
tailieuhoctap@hutech.edu.vn*

MỤC LỤC

MỤC LỤC.....	I
HƯỚNG DẪN	III
BÀI 1. MÃ HÓA ĐỐI XỨNG CỔ ĐIỂN	1
1.1 THUẬT TOÁN MÃ HÓA CEASAR	1
1.2 THUẬT TOÁN MÃ HÓA VIGENERE.....	5
1.3 THUẬT TOÁN MÃ HÓA RAIL FENCE	9
1.4 THUẬT TOÁN MÃ HÓA PLAYFAIR.....	12
1.5 THUẬT TOÁN MÃ HÓA TRANSPOSITION CIPHER	17
BÀI 2. MÃ HÓA ĐỐI XỨNG HIỆN ĐẠI.....	23
2.1 THUẬT TOÁN MÃ HÓA DES.....	23
2.1.1 Hướng dẫn mã hóa DES:	23
2.1.2 Hướng dẫn thực hành	30
2.2 THUẬT TOÁN MÃ HÓA 3DES.....	35
2.2.1 Hướng dẫn thuật toán TRIPLEDES:	35
2.2.2 Hướng dẫn thực hành	36
2.3 THUẬT TOÁN MÃ HÓA AES.....	39
2.3.1 Hướng dẫn thuật toán AES	40
2.3.2 Hướng dẫn thực hành	42
BÀI 3. MÃ HÓA KHÓA CÔNG KHAI RSA.....	48
3.1 GIỚI THIỆU THUẬT TOÁN RSA	48
3.2 HƯỚNG DẪN THỰC HÀNH	50
3.3 BÀI TẬP NÂNG CAO	61
BÀI 4. QUẢN LÝ KHÓA DÙNG MÃ KHÓA CÔNG KHAI.....	65
4.1 HƯỚNG DẪN THUẬT TOÁN TRAO ĐỔI KHÓA DIFFIE-HELLMAN	65
4.2 HƯỚNG DẪN THỰC HÀNH	66
4.2.1 Tạo Class CryptoUtil viết phương thức sau	66
4.2.2 Bên Alice	67
4.2.3 Bên Bob	70
4.2.4 Viết Frame DESCS.....	73
4.2.5 Kết Quả.....	76
BÀI 5. HÀM BẮM.....	79
5.1 HÀM BẮM MD5.....	79
5.1.1 Giới thiệu thuật toán MD5	79

5.1.2 Thuật toán MD5	80
5.2 HƯỚNG DẪN THỰC HÀNH.....	86
5.3 THUẬT TOÁN SHA	91

HƯỚNG DẪN

MÔ TẢ MÔN HỌC

Môn học này cung cấp cho sinh viên những kiến thức về các thuật toán mã hóa và ứng dụng trong thực tế hiện nay.

Nội dung môn học nhấn mạnh đến các nguyên tắc, các chủ đề, các phương pháp tiếp cận và giải quyết vấn đề liên quan đến các công nghệ và kiến trúc cơ bản của lĩnh vực này.

NỘI DUNG MÔN HỌC

BÀI 1. MÃ HÓA ĐỐI XỨNG CỔ ĐIỂN

BÀI 2. MÃ HÓA ĐỐI XỨNG HIỆN ĐẠI

BÀI 3. MÃ HÓA KHÓA CÔNG KHAI RSA

BÀI 4. QUẢN LÝ KHÓA DÙNG MÃ KHÓA CÔNG KHAI

BÀI 5. HÀM BĂM

KIẾN THỨC TIỀN ĐỀ

Sinh viên có kiến thức căn bản về kiến trúc máy tính, kỹ thuật lập trình, một số hệ điều hành như Windows, Linux.

YÊU CẦU MÔN HỌC

Người học phải dự học đầy đủ các buổi lên lớp và làm bài tập đầy đủ ở nhà.

CÁCH TIẾP CẬN NỘI DUNG MÔN HỌC

Để học tốt môn này, người học cần đọc trước các nội dung chưa được học trên lớp; tham gia đều đặn và tích cực trên lớp; hiểu các khái niệm, tính chất và ví dụ tại lớp học. Sau khi học xong, cần ôn lại bài đã học, làm các bài tập và câu hỏi. Tìm đọc thêm các tài liệu khác liên quan đến bài học và làm thêm bài tập.

PHƯƠNG PHÁP ĐÁNH GIÁ MÔN HỌC

- Điểm quá trình: 50%. Điểm kiểm tra thường xuyên trong quá trình học tập. Điểm kiểm tra giữa học phần, điểm làm bài tập trên lớp, hoặc điểm chuyên cần.
- Điểm thi: 50%. Hình thức bài thi tự luận trong 90 phút, không được mang tài liệu vào phòng thi. Nội dung gồm các câu hỏi và bài tập tương tự như các câu hỏi và bài tập về nhà.

BÀI 1. MÃ HÓA ĐỐI XỨNG CỔ ĐIỂN

1.1 THUẬT TOÁN MÃ HÓA CEASAR

Viết chương trình mã hóa và giải mã văn bản với thuật toán mã hóa Caesar.

Chương trình có thể thực hiện các chức năng sau:

Cho phép nhập văn bản vào hệ thống.

Cho phép nhập khóa bảo vệ văn bản.

Cho phép ghi File và mở File.

Hướng dẫn mã dịch chuyển Caesar:

-Ta lần lượt đánh chỉ số cho các chữ cái bắt đầu từ 0.

- Gọi k là 1 số nguyên từ 0 ->25 được gọi là khóa.

-Hàm mã hóa: $E(p,k)=(p+k)\bmod 26$ với p là chỉ số của ký tự cần mã hóa. -Hàm giải mã: $D(c,k)=|c-k|\bmod 26$ với c là chỉ số của ký tự cần giải mã.

Encryption:

1. Map the plaintext characters to numbers : $a = 0, \dots, z = 25$

2. Encrypt the message (sequence of numbers m) using
$$c = a*m + b \bmod 26$$

where a and b are the encryption keys.

3. Map the numbers back to characters to obtain the cipher

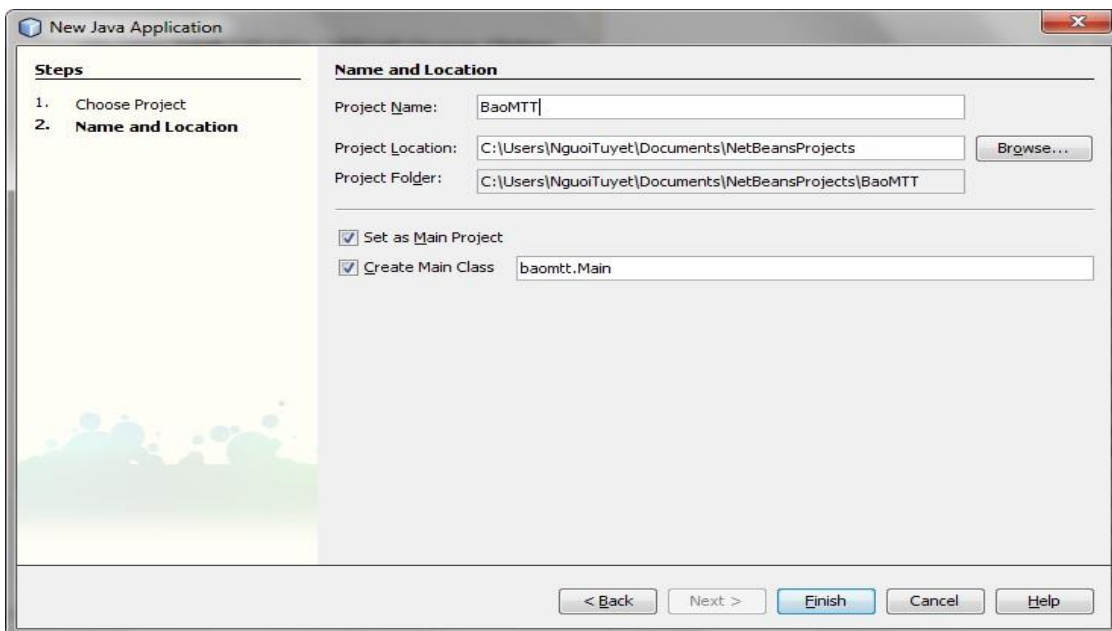
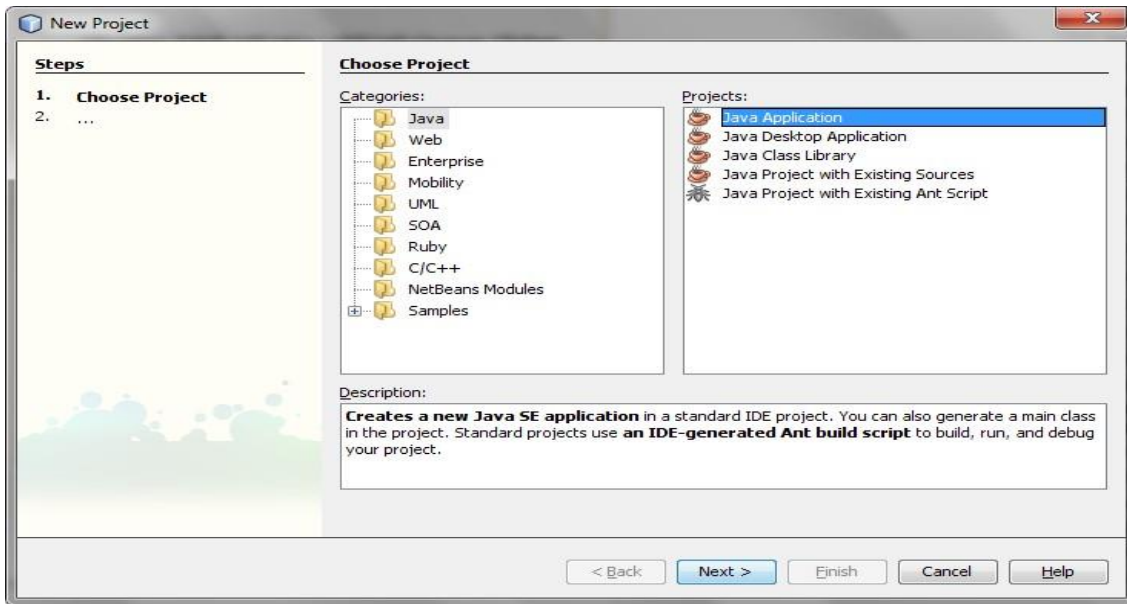
Decryption:

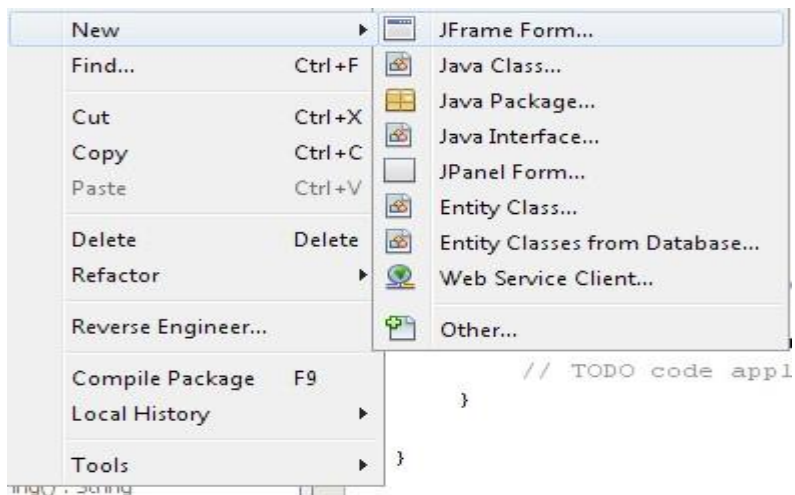
1. Map the cipher characters to numbers : $a = 0, \dots, z = 25$

2. Decrypt the number sequence c using
$$m = a^{-1}*c + a^{-1}*b \bmod 26$$

where a^{-1} is the **multiplicative inverse** of a mod 26

3. Map the numbers back to characters

Bước 1: Tạo project mới: File → New Project**Bước 2: Tạo mới JFrame Form thiết kế:**



Giao diện thiết kế Frame:



Bước 3: Thiết kế Form

A screenshot of a Java Swing window titled "Chương Trình Mã Hóa/ Giải Mã Ceasar Cipher". The window has a light gray background and a thin orange border. It contains the following elements: a label "PlainText:" followed by a large text area; a label "Khóa :" followed by a small text field; a button labeled "v Encrypt v"; a button labeled "Ghi File"; a label "Cipher Text :" followed by a large text area; a button labeled "^ Dencrypt ^"; and a button labeled "Mở File".

Bước 4: Viết hàm xử lý sự kiện

a. Hàm xử lý sự kiện Encrypt

```
private void btnmahoa(java.awt.event.ActionEvent evt) {
    int k=Integer.valueOf(this.txtkhoa.getText());
    String br=this.txtvanban.getText();
    this.txtmahoa.setText(mahoa(br,k));
}
```

b. Hàm xử lý sự kiện Ghi File

```
private void bntGhiFileActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        // TODO add your handling code here:
        BufferedWriter bw = null;
        //Nơi lưu dữ liệu
        String fileName = "D:\\Dulieu.txt";
        //luu van ban
        String s = txtmahoa.getText();
        // Ghi dữ Liệu S vào tạo tin fileName
        bw = new BufferedWriter(new FileWriter(fileName));
        bw.write(s);
        bw.close();
        JOptionPane.showMessageDialog(null, " Đã Ghi File Thành Công !!!");
    } catch (IOException ex) {
        Logger.getLogger(Cesar.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

c. Hàm xử lý sự kiện Dencrypt

```
private void btngiaima(java.awt.event.ActionEvent evt) {
    int k=Integer.valueOf(this.txtkhoa.getText());
    String br=this.txtmahoa.getText();
    this.txtvanban.setText(mahoa(br,-k));
}

char mahoakt(char c,int k)
{
    if(!Character.isLetter(c)) return c;
    return (char) (((Character.toUpperCase(c) - 'A') + k) % 26 + 26) % 26 + 'A');
}

private String mahoa(String br, int k) {
    String kq="";
    int n=br.length();
    for(int i=0;i<n;i++)
        kq+=mahoakt(br.charAt(i),k);
    return kq;
}
```

d. Hàm xử lý sự kiện Mở File

```
private void bntMoFileActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        // TODO add your handling code here:
        // Lớp BufferedReader kế thừa từ lớp Reader thuộc nhóm Input
        //Nhóm này là 1 trong 2 loại chính của Các luồng Ký Tự
        // Có chức năng đọc dữ liệu dạng ký tự
        BufferedReader br = null;
        String fileName = "D:\\Dulieu.txt";
        br = new BufferedReader(new FileReader(fileName));
        //nhận dữ liệu
        StringBuffer sb = new StringBuffer();
        JOptionPane.showMessageDialog(null, " Đã mở File Thành Công !!!");
        //Đọc mỗi lần tối đa 5 ký tự
        char[] ca = new char[5];
        while (br.ready()) {
            int len = br.read(ca);
            sb.append(ca, 0, len);
        }
        br.close();

        //xuat chuoi
        System.out.println("Du Lieu la : " + " " + sb);
        String chuoi = sb.toString();
        // Hiển thị lên Form
        txtvanban.setText(chuoi);
    } catch (IOException ex) {
        Logger.getLogger(Ceasar.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

1.2 THUẬT TOÁN MÃ HÓA VIGENERE

Viết chương trình mã hóa và giải mã văn bản với thuật toán mã hóa Vigenere.

Chương trình có thể thực hiện các chức năng sau:

Cho phép nhập văn bản vào hệ thống.

Cho phép nhập khóa bảo vệ văn bản.

Cho phép mở File và Ghi File.

Hướng Dẫn: Mật mã Vigenere còn gọi là mật mã nhiều bảng mã. Ưu điểm của mã này là việc sử dụng 26 bảng mã khác nhau. Do đó mà không bị phá trong một thời gian dài. Ngoài ra mã này còn hỗ trợ việc sử dụng từ khóa vô cùng tiện lợi.

Thuật toán:

- Khóa K là một bộ gồm nhiều khoá con:
 - $K = (k_1, k_2, \dots, k_m)$

- **Mã hoá:**

- $e_k(x_1, x_2, \dots, x_m) = (x_1 + k_1, x_2 + k_2, \dots, x_m + k_m)$

- **Giải mã:**

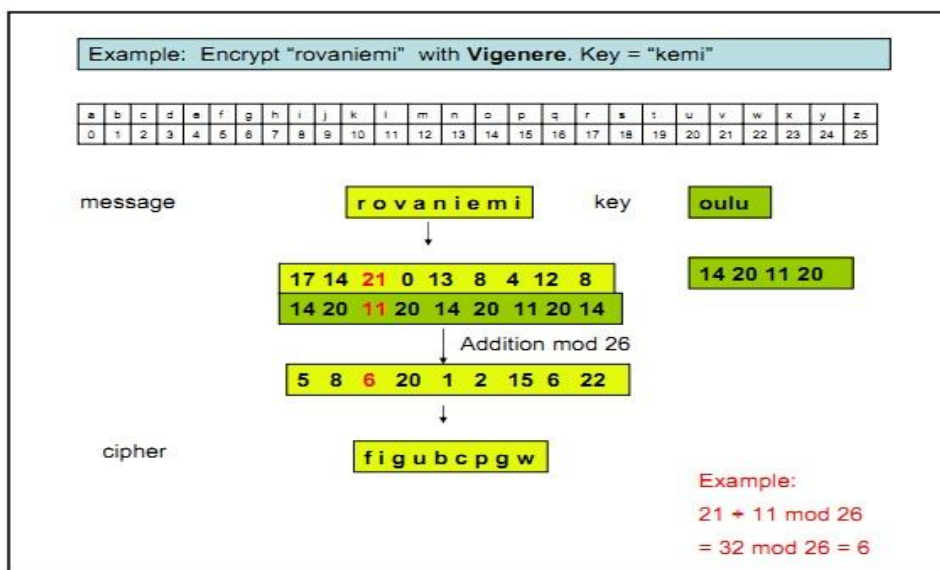
- $d_k(y_1, y_2, \dots, y_m) = (y_1 - k_1, y_2 - k_2, \dots, y_m - k_m)$

- (cộng, trừ theo modulo 26)

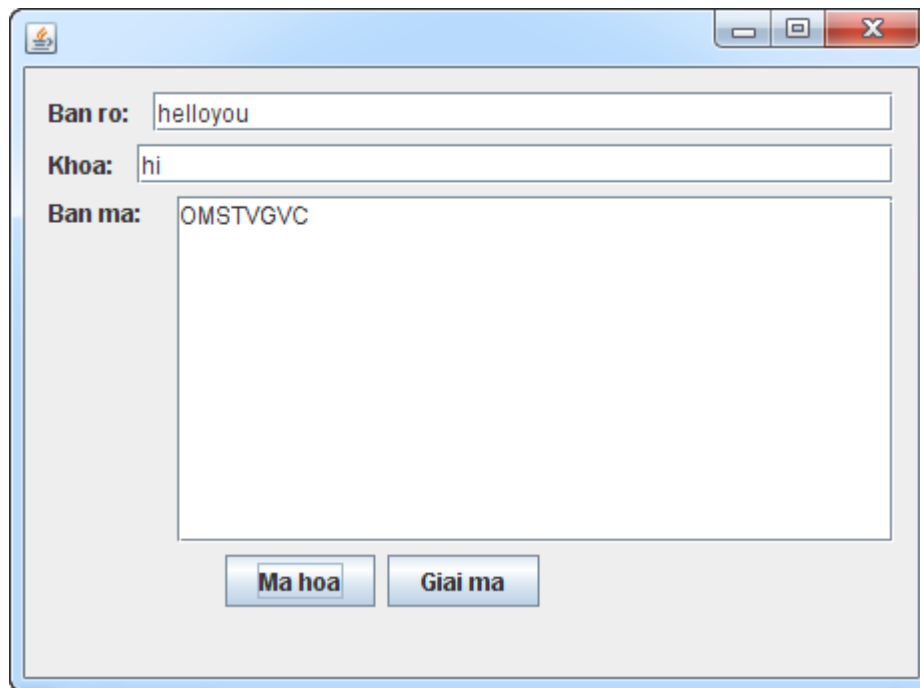
→ "MÃ KHỐI" (block cipher)

Ví dụ:

- $\{A, B, C, \dots, X, Y, Z\} = \mathbb{Z}_{26} = \{0, 1, \dots, 25\}$.
- $K = (2, 8, 15, 7, 4, 17)$ ("CIPHER").
- $p = \text{"thiscryptosy"}$.
- $c = \text{"VPXZGIA\u0304XIVWP\u0304"}$



Bước 1: Thiết Kế Form:



Hàm khởi tạo form:

```
public class Form2 extends javax.swing.JFrame {  
    int vig[][];  
    /** Creates new form Form2 */  
    public Form2() {  
        initComponents();  
        vig=new int[26][26];  
        for(int i=0;i<26;i++)  
            for(int j=0;j<26;j++)  
                vig[i][j]=(i+j)%26;  
    }  
}
```

Bước 2: Viết hàm xử lý sự kiện

a. Hàm xử lý sự kiện Encrypt

```
private void btnMaHoa(java.awt.event.ActionEvent evt) {  
    String banro=this.txtbr.getText();  
    String k=this.txtk.getText();  
    String banma=mahoa(banro,k);  
    this.txtbm.setText(banma);  
}
```

```

private String mahoa (String banro, String key) {
    int n=banro.length();
    String banma="";
    int k=0;
    for (int i=0; i<n; i++)
        if (Character.isLetter(banro.charAt(i)))
        {
            banma+=mahoa(banro.charAt(i), key.charAt(k));
            k++;
            k=k%key.length();
        } else
            banma+=banro.charAt(i);
    return banma;
}

char mahoa (char x, char k)
{
    int xn=Character.toUpperCase(x)-'A';
    int kn=Character.toUpperCase(k)-'A';
    int yn=vig[kn][xn];
    return (char) (yn+'A');
}

```

b. Hàm xử lý sự kiện Dencypt

```

private void btnGiaiMa (java.awt.event.ActionEvent evt) {
    String banma=this.txtbm.getText();
    String k=this.txtk.getText();
    String kt1="";
    int kn=k.length();
    for (int i=0; i<kn; i++)
        kt1+=(char) (((26-(Character.toUpperCase(k.charAt(i))-'A'))%26)+'A');
    this.txtk.setText(kt1);
    String banro=mahoa(banma, kt1);
    this.txtbr.setText(banro);
}

```

Bài Tập mở rộng: Yêu cầu viết phần mềm mã hóa và giải mã với 2 thuật toán trên bao gồm:

- Menu mã hóa: Thuật toán Ceasar, Thuật Toán Vigenere
- Menu giải mã: Thuật toán Ceasar, Thuật Toán Vigenere

Các chức năng mã hóa và giải mã đều phải có mục mã hóa và giải mã theo File(File có thể là .txt, .dat,...)

1.3 THUẬT TOÁN MÃ HÓA RAIL FENCE

Viết chương trình mã hóa và giải mã văn bản với thuật toán mã hóa Rail Fence.

Chương trình có thể thực hiện các chức năng sau:

Cho phép nhập văn bản vào hệ thống.

Cho phép nhập khóa bảo vệ văn bản.

Cho phép mở File và Ghi File.

Hướng Dẫn : Mã Rail Fence còn được gọi là mã zig zag là một hình thức của mã chuyển vị:

- Thông điệp được viết lần lượt từ trái qua phải trên các cột (rail) của một hàng đào tưởng tượng theo đường chéo từ trên xuống dưới.
- Theo đường chéo từ dưới lên khi đạt tới cột thấp nhất.
- Và khi đạt tới cột cao nhất, lại viết theo đường chéo từ trên xuống. Cứ lặp đi lặp lại như thế nào cho đến khi viết hết toàn bộ nội dung của thông điệp.
- Ví dụ: mã hóa chuỗi HUTECH TECHNOLOGY với khóa là 2.

Plaintext:	HUTECHTECHNOLOGY
RailFence:	H T C T C N L G U E H E H O O Y
Ciphertext:	HTCTCNLGUEHEHOXY

Bước 1: Thiết Kế Form:

Thuật Toán Mã Hóa Rail Fence Cipher

Plaintext:

Key:

Ciphertext:

Bước 2: Viết hàm xử lý sự kiện

a. Hàm xử lý sự kiện Encrypt

```
private void mahoabtn(java.awt.event.ActionEvent evt) {  
    int k=Integer.valueOf(this.txtkhoa.getText());  
    String s=this.txtbtr.getText();  
    int n=s.length();  
    int sd,sc;  
    sd=k;  
    sc=n/sd+1;  
    char hr[][]=new char[sd][sc];  
    int c,d;  
    c=0;d=0;  
    int sodu=n%sd;  
    for(int i=0;i<n;i++)  
    {
```



```

        hr[d][c]=s.charAt(i);
        d++;
        if(d==k)
        {
            c++; d=0;
        }
    }
    String kq="";
    int sokytu=sc;
    for(int i=0;i<sd;i++)
    {
        if(i>=sodu) sokytu=sc-1;
        for(int j=0;j<sokytu;j++)
            kq=kq+hr[i][j];
    }
    this.txtbm.setText(kq);
}

```

b. Hàm xử lý sự kiện Dencrypt:

```

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    int k=Integer.valueOf(this.txtkhoa.getText());
    String s=this.txtbm.getText();
    int n=s.length();
    int sd,sc;
    sd=k;
    sc=n/sd+1;
    char hr[][]=new char[sd][sc];
    for(int i=0;i<sd;i++)
    {
        if(i>=sodu) sokytu=sc-1;
        for(int j=0;j<sokytu;j++)
        {
            hr[i][j]=s.charAt(t);
            t++;
        }
    }
    int c,d;
    c=0;d=0;
}

```

```

for(int i=0;i<n;i++)
{
    kq+=hr[d][c];
    d++;
    if(d==k)
    {
        c++; d=0;
    }
}
this.txtbr.setText(kq);
}

```

Bước 3: Kiểm Tra



1.4 THUẬT TOÁN MÃ HÓA PLAYFAIL

Viết chương trình mã hóa và giải mã văn bản với thuật toán mã hóa PPlayFail.

Chương trình có thể thực hiện các chức năng sau:

Cho phép nhập văn bản vào hệ thống.

Cho phép nhập khóa bảo vệ văn bản.

Cho phép mở File và Ghi File.

Hướng dẫn:

✚ **Phương pháp** là lập ma trận 5x5 dựa trên từ khóa cho trước và các ký tự trên bảng chữ cái :

- Trước hết viết các chữ của từ khoá vào các hàng của ma trận bắt từ hàng thứ nhất.
- Nếu ma trận còn trống, viết các chữ khác trên bảng chữ cái chưa được sử dụng vào các ô còn lại. Có thể viết theo một trình tự qui ước trước, chẳng hạn từ đầu bảng chữ cái cho đến cuối.
- Vì có 26 chữ cái tiếng Anh, nên thiếu một ô. Thông thường ta dồn hai chữ nào đó vào một ô chung, chẳng hạn I và J.
- Giả sử sử dụng từ khoá MORNACHY. Lập ma trận khoá Playfair tương ứng như sau:

M	O	N	A	R
C	H	Y	B	D
E	F	G	I,J	K
L	P	Q	S	T
U	V	W	X	Z

✚ **Quy tắc mã hóa và giải mã**

- Chia bản rõ thành từng cặp chữ. Nếu một cặp nào đó có hai chữ như nhau, thì ta chèn thêm một chữ lọc chẳng hạn X. Ví dụ, trước khi mã “balloon” biến đổi thành “ba lx lo on”.
- Nếu cả hai chữ trong cặp đều rơi vào cùng một hàng, thì mã mỗi chữ bằng chữ ở phía bên phải nó trong cùng hàng của ma trận khóa (cuộn vòng quanh từ cuối về đầu), chẳng hạn “ar” biến đổi thành “RM” .
- Nếu cả hai chữ trong cặp đều rơi vào cùng một cột, thì mã mỗi chữ bằng chữ ở phía bên dưới nó trong cùng cột của ma trận khóa (cuộn vòng quanh từ cuối về đầu), chẳng hạn “mu” biến đổi thành “CM”.
- Trong các trường hợp khác, mỗi chữ trong cặp được mã bởi chữ cùng hàng với nó và cùng cột với chữ cùng cặp với nó trong ma trận khóa. Chẳng hạn, “hs” mã thành “BP”, và “ea” mã thành “IM” hoặc “JM” .

Bước 1: Thiết Kế Form:

The screenshot shows a Java Swing window with a light blue title bar and standard Windows-style window controls (minimize, maximize, close). The window contains a form with three text input fields. The first field is labeled 'Ban ro:' and contains the text 'HELLO'. The second field is labeled 'Khoa:' and contains the text 'MONARCHY'. The third field is labeled 'Ban ma:' and contains the text 'CFSUPM'. Below these fields are two buttons: 'Ma hoa' and 'Giai ma'.

Khai báo bảng khóa playfair ứng với khóa MONARCH khóa form:

```
public class Form3 extends javax.swing.JFrame {
    char pf[][]={{'M','O','N','A','R'},
                 {'C','H','Y','B','D'},
                 {'E','F','G','I','K'},
                 {'L','P','Q','S','T'},
                 {'U','V','W','X','Z'}};
};
```

Bước 2: Viết hàm xử lý sự kiện

a. Hàm xử lý sự kiện nút mã hóa

```
private void btnmahoa(java.awt.event.ActionEvent evt) {
    String banro=this.txtbr.getText();
    banro=banro.toUpperCase();
    banro=banro.replace('J', 'I');
    String banma=mahoa(banro);
    this.txtbm.setText(banma);
}
```

```
private String mahoa(String banro) {
    int n=banro.length();
    int i=0;
    String banma="";
    char a,b;
    while (i<n)
    {
        if(i==n-1)
        {
            a=banro.charAt(i);
            b='X';
            i++;
        }else{
            a=banro.charAt(i);
            b=banro.charAt(i+1);

            if(a==b)
            {
                b='X';
                i++;
            }else
                i+=2;
        }
        //thay the trong ban playfair
        banma+=thayThe(a,b);
    }
    return banma;
}
```

```

String thayThe(char a, char b)
{
    String vta=timViTri(a);
    String vtb=timViTri(b);
    char x,y;
    if(vta.charAt(0)==vtb.charAt(0)) //cung dong
    {
        x=pf[vta.charAt(0)-'0'][(vta.charAt(1)-'0')+1%5];
        y=pf[vtb.charAt(0)-'0'][(vtb.charAt(1)-'0')+1%5];
        return x+""+y;
    }
    if(vta.charAt(1)==vtb.charAt(1)) //cung dong
    {
        x=pf[(vta.charAt(0)-'0')+1%5][vta.charAt(1)-'0'];
        y=pf[(vtb.charAt(0)-'0')+1%5][vtb.charAt(1)-'0'];
        return x+""+y;
    }
    x=pf[vta.charAt(0)-'0'][vtb.charAt(1)-'0'];
    y=pf[vtb.charAt(0)-'0'][vta.charAt(1)-'0'];
    return x+""+y;
}

private String timViTri(char a) {
    for(int i=0;i<5;i++)
        for(int j=0;j<5;j++)
            if(pf[i][j]==a)
                return i+""+j;
    return "";
}

```

b. Hàm xử lý sự kiện nút giải mã

```

private void btngiaima(java.awt.event.ActionEvent evt) {
    String banma=this.txtbm.getText();
    String banro=giaima(banma);
    int n=banro.length();
    String br="";
    for(int i=0;i<n-2;i+=2)
        if(banro.charAt(i)==banro.charAt(i+2))
            br+=banro.charAt(i);
        else
            br+=banro.charAt(i)+ ""+banro.charAt(i+1);
    if(banro.charAt(n-1)=='X' )
        br+=banro.charAt(n-2);
    else
        br+=banro.charAt(n-2); br+=banro.charAt(n-1);
    this.txtbr.setText(br);
}

```

```

private String giaima(String banma) {
    int n=banma.length();
    String banro="";
    char a,b;
    for(int i=0;i<n;i+=2)
    {
        a=banma.charAt(i);
        b=banma.charAt(i+1);
        //thay the trong ban playfair
        banro+=thayTheNguoc(a,b);
    }
    return banro;
}

String thayTheNguoc(char a, char b)
{
    String vta=timViTri(a);
    String vtb=timViTri(b);
    char x,y;
    if(vta.charAt(0)==vtb.charAt(0)) //cung dong
    {
        x=pf[vta.charAt(0)-'0'][(vta.charAt(1)-'0')-1+5]%5;
        y=pf[vtb.charAt(0)-'0'][(vtb.charAt(1)-'0')-1+5]%5;
        return x+""+y;
    }
    if(vta.charAt(1)==vtb.charAt(1)) //cung dong
    {
        x= pf[ (vta.charAt(0)-'0')-1+5]%5 [(vta.charAt(1)-'0')];
        y=pf[ (vtb.charAt(0)-'0')-1+5]%5 [(vtb.charAt(1)-'0')];
        return x+""+y;
    }
    x=pf[ (vta.charAt(0)-'0')][(vtb.charAt(1)-'0')];
    y=pf[ (vtb.charAt(0)-'0')][(vta.charAt(1)-'0')];
    return x+""+y;
}

```

1.5 THUẬT TOÁN MÃ HÓA TRANSPOSITION CIPHER

Viết chương trình mã hóa và giải mã văn bản với thuật toán mã hóa Transposition cipher.

Chương trình có thể thực hiện các chức năng sau:

Cho phép nhập văn bản vào hệ thống.

Cho phép nhập khóa bảo vệ văn bản.

Cho phép mở File và Ghi File.

Hướng dẫn: Hệ mã hóa đối chỗ (Transposition Cipher)

Là hệ mã hóa trong đó các kí tự của bản gốc được giữ nguyên, nhưng vị trí bị thay đổi.

Đảo ngược toàn bộ plaintext: nghĩa là bản gốc được viết theo thứ tự ngược lại từ sau ra trước.

Ví dụ Plaintext: SECURE EMAIL

Bản mã: LIAMEERUCES

Mã hóa theo mẫu hình học: bản gốc được sắp xếp lại theo một mẫu hình học nào đó, thường là một mảng hoặc ma trận hai chiều.

Ví dụ: bản gốc ban đầu là BAO MAT

Ví dụ mã hoá theo mẫu hình học.

Cột 1	Cột 2	Cột 3
B	A	O
M	A	T

Nếu lấy các cột theo thứ tự 2, 3, 1. Bản mã sẽ là AAOTBM

Đổi chỗ cột: đổi chỗ các kí tự trong plaintext thành dạng hình chữ nhật theo cột.

Ví dụ : Bản gốc BAO MAT THU DIEN TU

Bản gốc được chuyển thành ma trận 3x5 như sau:

Bảng ví dụ mã hóa bằng phương pháp đổi chỗ cột

Cột 1	Cột 2	Cột 3	Cột 4	Cột 5
-------	-------	-------	-------	-------

B	M	T	D	N
A	A	H	I	T
O	T	U	E	U

Vì có 5 cột nên chúng có thể được sắp lại theo $5! = 120$ cách khác nhau

Nếu ta chuyển vị các cột theo thứ tự 3, 5, 2, 4, 1 rồi lấy các kí tự theo hàng ngang ta sẽ thu được bản mã: TNMDBHTAIAUUTEO.

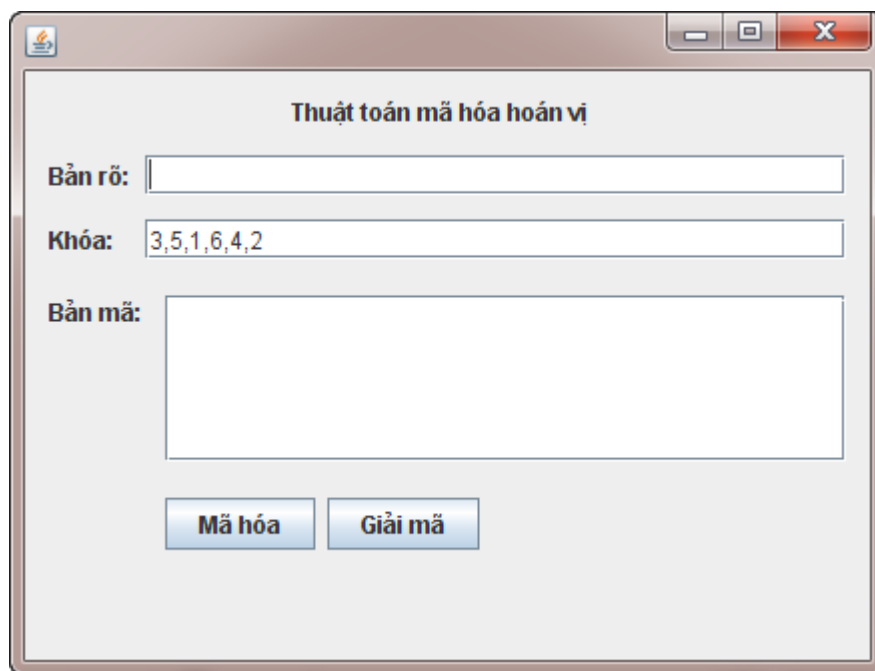
Hoán vị các kí tự của bản gốc theo chu kỳ cố định d: Nếu hàm f là hoán vị của một khối gồm d kí tự thì khóa mã hóa được biểu diễn bởi $K(d, f)$

Ví dụ: với $d = 5$, f hoán vị của dãy 12345 thành 35142 **Bảng Hoán vị các kí tự của bản gốc theo chu kỳ cố định d**

Vị trí ban đầu	Vị trí hoán vị	Nội dung mã hóa	Mã hóa
1	3	G	O
2	5	R	P
3	1	O	G
4	4	U	U
5	2	P	R

Theo bảng trên bản gốc ban đầu được mã hóa thành OPGUP.

Bước 1: Thiết Kế Form :



Thuật toán mã hóa hoán vị

Bản rõ:

Khóa:

Bản mã:

Bước 2: Viết hàm xử lý sự kiện

a. Hàm xử lý sự kiện Encrypt

```
private void mahoabtn(java.awt.event.ActionEvent evt) {  
    String k=this.txtkhoa.getText();  
    String ks[]=new String[6];  
    ks=k.split(",");  
    //System.out.println(ks.length);  
    int key[]=new int[6];  
    for(int i=0;i<6;i++)  
        key[i]=Integer.valueOf(ks[i])-1;  
    String sa=this.txtbr.getText();  
    String kq="";  
    int na=sa.length();  
    int d=0;  
    int c;  
    String s="";  
    int thieu=6-na%6;  
    for(int i=0;i<thieu;i++) sa=sa+" ";  
    while (d<na)  
    {  
        c=d+6;  
        s=sa.substring(d, c);  
        for(int i=0;i<6;i++)  
            kq=kq+s.charAt(key[i]);  
        d=d+6;  
    }  
    this.txtbm.setText(kq);  
}
```

b. Hàm xử lý sự kiện Dencrypt

```

private void btngiaima(java.awt.event.ActionEvent evt) {
    String k=this.txtkhoa.getText();
    String ks[]=new String[6];
    ks=k.split(",");
    int key[]=new int[6];
    for(int i=0;i<6;i++)
        key[i]=Integer.valueOf(ks[i])-1;
    int keyt1[]=new int[6];
    for(int i=0;i<6;i++)
        keyt1[key[i]]=i;
    String sa=this.txtbm.getText();
    String kq="";
    int na=sa.length();
    int d=0;
    int c;
    String s="";
    while (d<na)
    {
        c=d+6;
        s=sa.substring(d, c);
        for(int i=0;i<6;i++)
            kq=kq+s.charAt(keyt1[i]);
        d=d+6;
    }
    this.txtbr.setText(kq);
}

```

Bài Tập nâng cao: Viết phần mềm mã hóa văn bản với các thuật toán mã hóa trên.

Chương trình có thể thực hiện các chức năng sau:

Cho phép nhập văn bản vào hệ thống.

Cho phép nhập khóa bảo vệ văn bản.

Cho phép mở File và Ghi File.

Cho phép bên gửi mã hóa dữ liệu và bên nhận mã hóa dữ liệu với khóa K.

BÀI 2. MÃ HÓA ĐỐI XỨNG HIỆN ĐẠI

2.1 THUẬT TOÁN MÃ HÓA DES

Viết chương trình mã hóa và giải mã văn bản với thuật toán mã hóa DES.

Chương trình có thể thực hiện các chức năng sau:

Cho phép nhập văn bản vào hệ thống.

Cho phép nhập khóa bảo vệ văn bản.

Cho phép ghi File và mở File.

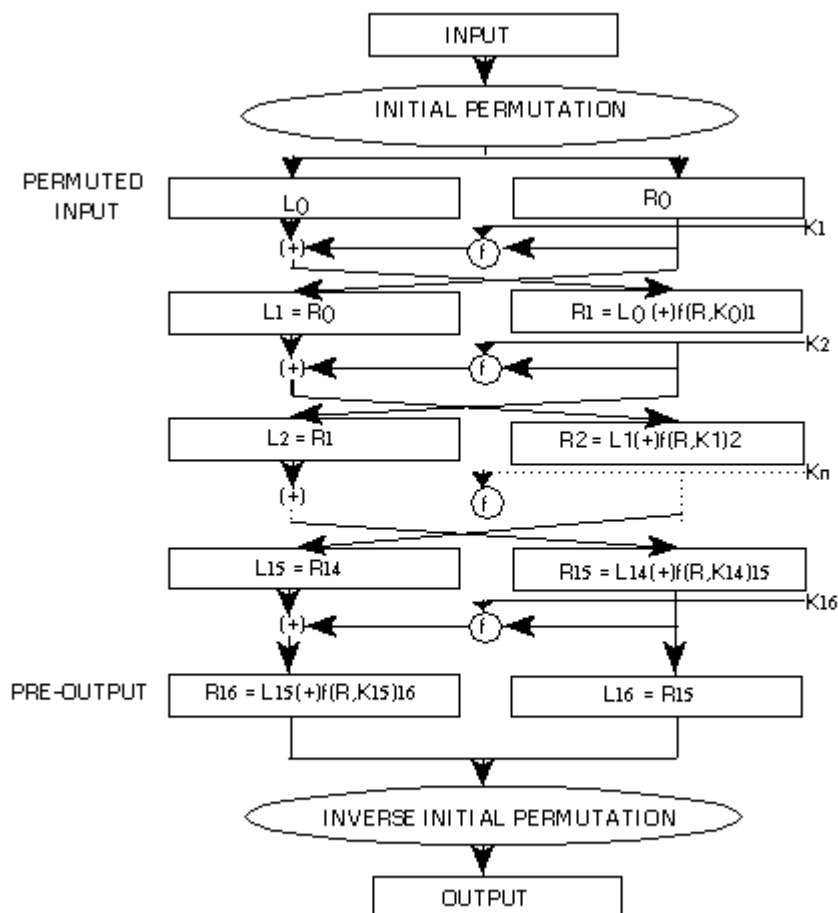
2.1.1 Hướng dẫn mã hóa DES:

DES Là một hệ mật mã được sử dụng rộng rãi nhất trên thế giới. DES được IBM phát triển vào những năm 1970 và được xem như cải biên của hệ mật mã LUCIPHER. DES được chấp nhận bởi National Bureau of Standards, ngày nay gọi là NIST (National Institute of Standards and Technology). DES trở thành chuẩn mã hóa dữ liệu chính thức của chính phủ Hoa Kỳ vào năm 1977.

Mô tả thuật toán:

DES là thuật toán mã hóa khối (block cipher), mỗi khối dữ liệu có độ dài 64 bit. Một block bản gốc sau khi mã hóa tạo ra một block bản mã. Quá trình mã hóa và giải mã đều dùng chung một khóa.

Khóa có độ dài là 56 bit, cộng thêm 8 bit chẵn lẻ được sử dụng để kiểm soát lỗi. Các bit chẵn lẻ nằm ở các vị trí 8, 16, 24...64. tức là cứ 8 bit thì có một bit kiểm soát lỗi .

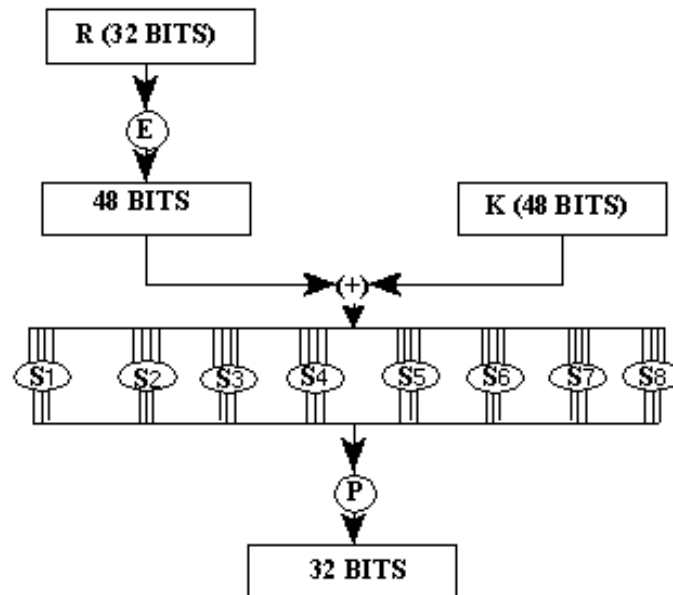


Hình 2.1 Sơ đồ hoạt động của DES

Theo sơ đồ hoạt động của DES như trên ta có thể thấy:

DES thực hiện trên từng block bản gốc. Sau khi thực hiện hoán vị khởi đầu (Initial Permutation – IP) khối dữ liệu được chia làm hai nửa trái và phải, mỗi nửa 32 bit. Quá trình được lặp lại qua 16 vòng, mỗi vòng là một hàm f . Sau 16 vòng lặp, hai nửa trái và phải được kết hợp lại và thực hiện hoán vị cuối cùng (hoán vị ngược – Inverse Initial Permutation) để kết thúc thuật toán.

Mỗi vòng của DES được thực hiện theo các bước sau:



Hình 2.2 Một vòng hoạt động của DES

Bước 1: Sử dụng hoán vị khởi đầu để thay đổi thứ tự các bit.

Bảng P3.B10: Bảng hoán vị khởi đầu: (hoán vị bit 1 thành bit 58, bit 2 thành bit 50....)

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

Bước 2: Bản gốc được chia làm hai nửa trái và phải, mỗi nửa 32 bit.

Bước 3: Ban đầu khóa 64 bit được bỏ đi 8 bit kiểm soát lỗi. Sự loại bỏ được thực hiện theo bảng sau:

Bảng 1: Bảng loại bỏ 8 bit kiểm soát lỗi

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

Sau đó khóa được chia làm hai nửa, mỗi nửa 28 bit.

Bước 4: Các nửa của khóa lần lượt được dịch trái (số bit dịch là 1 hay 2 tùy theo vòng thực hiện).

Bảng 2: Bảng dịch:

Vòng	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Số bit dịch	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Các nửa được ghép lại với nhau, hoán vị và chọn ra 48 bit bằng cách đổi chỗ các bit theo bảng hoán vị nén - compression permutation (hay còn gọi là hoán vị lựa chọn-permuted choice):

Bảng 3: Bảng hoán vị nén: (bit ở vị trí 14 của khóa dịch được chuyển tới vị trí 1 của đầu ra, bit ở vị trí 17 của khóa dịch được chuyển tới vị trí 2 của đầu ra,..., bit thứ 18 bị loại bỏ...)

14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

Bước 5: 32 bit của bản gốc bên phải được mở rộng thành 48 bit để XOR với 48 bit khóa. Khối bit này lại thực hiện hoán vị một lần nữa, thay đổi thứ tự các bit bằng cách lặp lại một số bit (hoán vị mở rộng - Expansion Permutation).

Bảng 4: Bảng hoán vị mở rộng – hộp E (bit ở vị trí thứ 32 của khối dữ liệu vào được chuyển tới vị trí thứ nhất trong khối dữ liệu ra, bit ở vị trí thứ 4 của khối dữ liệu vào được chuyển tới vị trí thứ 5 và 7 trong khối dữ liệu ra,...)

32	1	2	3	4	5	4	5	6	7	8	9
8	9	10	11	12	12	12	13	14	15	16	17
16	17	18	19	20	21	20	21	22	23	24	25
24	25	26	27	28	29	28	29	30	31	32	1

Bước 6: kết quả của bước 3 và bước 5 được XOR với nhau.

Bước 7: Kết quả của bước 6 được chuyển thành 32 bit bằng cách sử dụng hàm thay thế và lựa chọn.

Sự thay thế được thực hiện bởi 8 hộp thay thế (substitution boxes, S-boxes). Khối 48 bit được chia thành 8 khối 6 bit. Mỗi khối được thực hiện trên một hộp S riêng biệt (separate S-box): khối 1 được thực hiện trên hộp S1, khối 2 được thực hiện trên hộp S2...

Mỗi hộp S là một bảng gồm 4 hàng và 16 cột. Mỗi phần tử của hộp là một số 4 bit. Với sáu bit vào hộp S sẽ xác định được số hàng và số cột để tìm ra kết quả.

Cách thức xác định kết quả: nhận vào 6 bit lần lượt là b_1, b_2, b_3, b_4, b_5 , và b_6 . Bit b_1 và b_6 được kết hợp lại thành một số 2 bit tương ứng với số hàng trong bảng (có giá trị từ 0 đến 3). Bốn bit ở giữa được kết hợp lại thành một số 4 bit tương ứng với số cột trong bảng (nhận giá trị từ 0 đến 15).

Ví dụ : Dùng hộp S thứ 6. Nếu dữ liệu nhận vào là 110010. Bit đầu tiên kết hợp với bit cuối tạo thành 10 (khi đổi sang số thập phân có giá trị bằng 2 tương ứng với hàng thứ 2). Bốn bit giữa kết hợp lại thành 1001 (khi đổi sang số thập phân có giá trị bằng 9 tương ứng với cột thứ 9) \Rightarrow Giá trị cần tìm hàng 2 cột 9 là 0. Như vậy giá trị 0000 được thay thế cho 110010.

Dùng hộp S thứ nhất. Nếu dữ liệu nhận vào là 011011. Bit đầu tiên kết hợp với bit cuối tạo thành 01 (khi đổi sang số thập phân có giá trị bằng 1 tương ứng với hàng 1). Bốn bit giữa kết hợp lại thành 1101 (khi đổi sang số thập phân có giá trị bằng 13 tương ứng với cột thứ 13) \Rightarrow Giá trị cần tìm hàng 1 cột 13 là 5. Như vậy giá trị 0101 được thay thế cho 011011.

Bảng 5: Bảng hộp S:

Hộp S thứ nhất.

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Hộp S thứ hai.

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

Hộp S thứ ba.

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

Hộp S thứ tư.

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Hộp S thứ năm.

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

Hộp S thứ sáu.

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

Hộp S thứ bảy.

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
---	----	---	----	----	---	---	----	---	----	---	---	---	----	---	---

13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

Hộp S thứ tám.

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Kết quả của sự thay thế là 8 khối 4 bit được sinh ra, chúng được kết hợp lại thành một khối 32 bit. Khối này được chuyển tới bước tiếp theo: hộp hoán vị P (P-box permutation). Hoán vị ở bước này ánh xạ mỗi bit dữ liệu vào tới một vị trí trong khối dữ liệu ra, không có bit nào bị bỏ qua cũng như được sử dụng hai lần. nó còn được gọi là hoán vị trực tiếp (straight permutation).

Bảng 6: Bảng hộp hoán vị P cho biết vị trí của mỗi bit cần chuyển (bit 1 chuyển tới bit 16, bit 2 chuyển tới bit 7...)

16	7	20	21	29	12	28	17	1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25

Bước 8: Kết quả của bước 7 được XOR với nửa trái 32 bit được tạo ra ở bước 2.

Bước 9: kết quả tạo ra ở bước 8 trở thành nửa phải mới, nửa phải cũ (tạo ở bước 2) trở thành nửa trái mới.

Sau khi thực hiện hết 16 vòng lặp hoán vị cuối cùng được thực hiện để kết thúc thuật toán.

Hoán vị cuối cùng là nghịch đảo của hoán vị khởi đầu.

Bảng 7: Bảng hoán vị cuối

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

Các chế độ hoạt động của DES

Có bốn chế độ làm việc đã được phát triển cho DES:

- Chế độ sách mã điện tử (ECB).
- Chế độ phản hồi mã (CFB).
- Chế độ liên kết khối mã (CBC – Cipher Block Chaining).
- Chế độ phản hồi đầu ra (OFB).

2.1.2 Hướng dẫn thực hành

Bước 1: Thiết Kế Form :

The image shows a graphical user interface for a DES encryption/decryption tool. The window is titled "DES". It contains the following elements:

- Input Key :** A text input field for entering the encryption key.
- Buttons:** Three buttons labeled "Mã Hóa" (Encrypt), "Hiển Thị" (Show), and "Ghi File" (Save File) are positioned below the key input.
- Plaintext :** A large text area for entering the plaintext, with a vertical scrollbar on the right.
- CipherText :** A large text area for displaying the resulting ciphertext, also with a vertical scrollbar on the right.
- Buttons:** Two buttons labeled "Giải Mã" (Decrypt) and "All Show" are located at the bottom of the form.

Bước 2: Viết hàm xử lý sự kiện

B2.1 Hàm doCopy

```

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();
}

```

B2.2 Hàm mã Hóa và giải mã

```

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.DECRYPT_MODE, is, os);
}

```

B2.3 Hàm thực hiện

```

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"); // DES/ECB/PKCS5Padding for SunJCE

    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);
    }
}

```

B2.4 Viết chức năng Mã Hóa

```

private void bntMaHoaActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        String key=txtkhoa.getText() ; // needs to be at least 8 characters for DES

        FileInputStream fis = new FileInputStream("D:\\Des.txt");
        FileOutputStream fos = new FileOutputStream("D:\\EnDes.txt");
        encrypt(key, fis, fos);
        JOptionPane.showMessageDialog(null, " Đã mã hóa văn bản");
        //FileInputStream fis2 = new FileInputStream("D:\\encrypted.txt");
        //FileOutputStream fos2 = new FileOutputStream("D:\\decrypted.txt");
        //decrypt(key, fis2, fos2);
    } catch (Throwable e) {
        e.printStackTrace();
    }
}

```

B2.5 Viết Chức năng Giải Mã

```

private void bntGiaiMaActionPerformed(java.awt.event.ActionEvent evt) {
    FileInputStream fis2 = null;
    try {
        String key = txtkhoa.getText();
        fis2 = new FileInputStream("D:\\EnDes.txt");
        FileOutputStream fos2 = new FileOutputStream("D:\\DeDes.txt");
        decrypt(key, fis2, fos2);
        BufferedReader br = null;
    }
}

```

```

        br = new BufferedReader(new FileReader(fileName));
        StringBuffer sb = new StringBuffer();
        JOptionPane.showMessageDialog(null, " Đã Giải Mã");
        char[] ca = new char[5];
        while (br.ready()) {
            int len = br.read(ca);
            sb.append(ca, 0, len);
        }
        br.close();
        //xuất chuỗi
        System.out.println("Dữ Liệu là : " + " " + sb);
        String chuỗi = sb.toString();

        txtmahoa.setText(chuoi);
    } catch (Throwable ex) {
        Logger.getLogger(DESCS.class.getName()).log(Level.SEVERE, null, ex);
    } finally {
        try {
            fis2.close();
        } catch (IOException ex) {
            Logger.getLogger(DESCS.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
}

```

B2.6 Viết chức năng Ghi FILE

```

private void bntGhiFileActionPerformed(java.awt.event.ActionEvent evt) {
    try {

        BufferedWriter bw = null;
        //ghi văn bản đã mã hóa
        String fileName = "D:\\Des.txt";
        //lưu văn bản
        String s = txtvanban.getText();
        bw = new // văn bản sau khi mã hóa
        BufferedWriter(new FileWriter(fileName));
        // ghi văn bản
        bw.write(s);
        bw.close();
        JOptionPane.showMessageDialog(null, " Đã ghi file");
        txtmahoa.setText(s);
    } catch (IOException ex) {
        Logger.getLogger(DESCS.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

B2.7 Viết chức năng Mở FILE

```

private void bntMoFileActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        BufferedReader br = null;

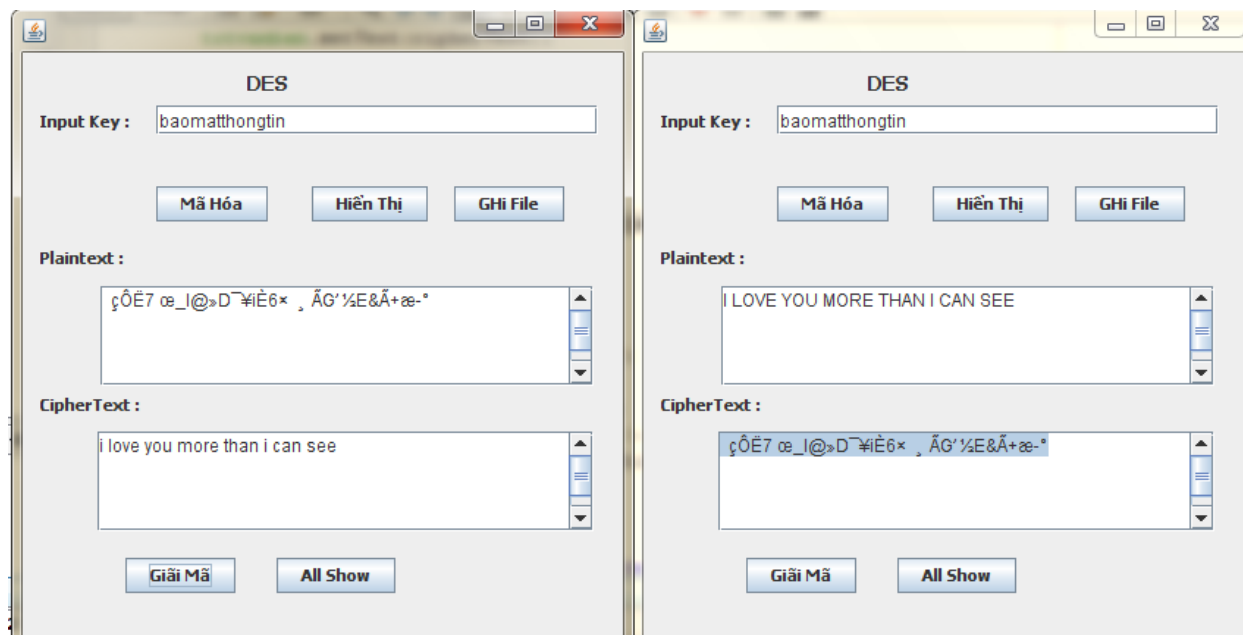
        String fileName = "D:\\\\EnDes.txt"; //GEN-
        br = new BufferedReader(new FileReader(fileName));
        StringBuffer sb = new StringBuffer();

        JOptionPane.showMessageDialog(null, " Đã mở file");
        char[] ca = new char[5];
        while (br.ready()) {
            int len = br.read(ca);
            sb.append(ca, 0, len);
        }
        br.close();
        //xuất chuỗi
        System.out.println("Du Lieu la : " + " " + sb);
        String chuoi = sb.toString();

        txtvanban.setText(chuoi);
    } catch (IOException ex) {
        Logger.getLogger(DESCS.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

Bước 3: Kiểm Tra



2.2 THUẬT TOÁN MÃ HÓA 3DES

Viết chương trình mã hóa và giải mã văn bản với thuật toán mã hóa 3DES.

Chương trình có thể thực hiện các chức năng sau:

Cho phép nhập văn bản vào hệ thống.

Cho phép nhập khóa bảo vệ văn bản.

Cho phép ghi File và mở File.

2.2.1 Hướng dẫn thuật toán TRIPLEDES:

TripleDES một biến thể an toàn hơn của DES còn được gọi là DESede hay 3DES. TripleDES có tính bảo mật cao hơn DES do sử dụng 3 vòng DES với các khóa khác nhau. Vòng đầu tiên và vòng thứ ba là vòng mã hóa, vòng thứ hai là vòng giải mã. DESede có thể dùng hai hoặc ba khóa có độ dài 56, 112 hoặc 168. nếu dùng hai khóa thì khóa đầu tiên được dùng cho vòng thứ nhất và vòng thứ ba, khóa thứ hai dùng cho vòng thứ hai.

Mã hóa với ba khóa 56 bit (168 bit).

Bảng B1: 3DES Mã hóa với ba khóa 56 bit

NGƯỜI GỬI	NGƯỜI NHẬN
Bước 1: mã hóa plaintext bằng khóa nhất	Bước 1: giải mã bản mã với khóa thứ ba
Bước 2: mã hóa văn bản được tạo ra ở bước 1 bằng khóa thứ hai	Bước 2: giải mã văn bản được tạo ra ở bước 1 bằng khóa thứ hai
Bước 3: mã hóa văn bản được tạo ra ở bước 2 bằng khóa thứ ba, tạo ra bản mã gửi cho người nhận.	Bước 3: giải mã văn bản được tạo ra ở bước 2 bằng khóa thứ nhất, tạo ra bản gốc do người gửi gửi.

Mã hóa với hai khóa 56 bit (112 bit)

Bảng B2: 3DES Mã hóa với hai khóa 56 bit

NGƯỜI GỬI	NGƯỜI NHẬN
Bước 1: mã hóa plaintext bằng khóa nhất	Bước 1: giải mã bản mã với khóa thứ nhất
Bước 2: giải mã văn bản được tạo ra ở bước 1 bằng khóa thứ hai	Bước 2: mã hóa văn bản được tạo ra bước 1 bằng khóa thứ hai
Bước 3: mã hóa văn bản được tạo ra ở bước 2 bằng khóa thứ nhất, tạo ra bản mã gửi cho người nhận.	Bước 3: giải mã văn bản được tạo ra bước 2 bằng khóa thứ nhất, tạo ra bản gốc do người gửi gửi.

Mã hóa với một khóa 56 bit

Bảng B3: 3DES Mã hóa với một khóa 56 bit

NGƯỜI GỬI	NGƯỜI NHẬN
Bước 1: mã hóa plaintext	Bước 1: giải mã bản mã nhận được từ người gửi.
Bước 2: giải mã văn bản được tạo ra ở bước 1	
Bước 3: mã hóa văn bản được tạo ra ở bước, tạo ra bản mã gửi cho người nhận.	

Mặc dù 3DES có tính bảo mật cao hơn DES, nhưng thực tế ít được sử dụng vì để tạo ra được bản mã phải chạy ba lần DES, nên tốc độ chậm, chiếm nhiều tài nguyên.

2.2.2 Hướng dẫn thực hành

Bước 1: Thiết Kế Form:



Bước 2: Viết hàm xử lý sự kiện

B2.1: Khai báo các biến sau

```
private static final String UNICODE_FORMAT = "UTF8";
public static final String DESEDE_ENCRYPTION_SCHEME = "DESede";
private KeySpec myKeySpec;
private SecretKeyFactory mySecretKeyFactory;
private Cipher cipher;
byte[] keyAsBytes;
private String myEncryptionKey;
private String myEncryptionScheme;
SecretKey key;
```

B2.2: Viết phương thức mã hóa encrypt

```

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;
}

```

B2.3: Viết phương thức giải mã decrypt

```

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("chuoi plaintext : " + a);
        // decryptedText= bytes2String(plainText);
        decryptedText=a;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return decryptedText;
}

```

B2.4 Viết hàm xử lý sự kiện mã hóa

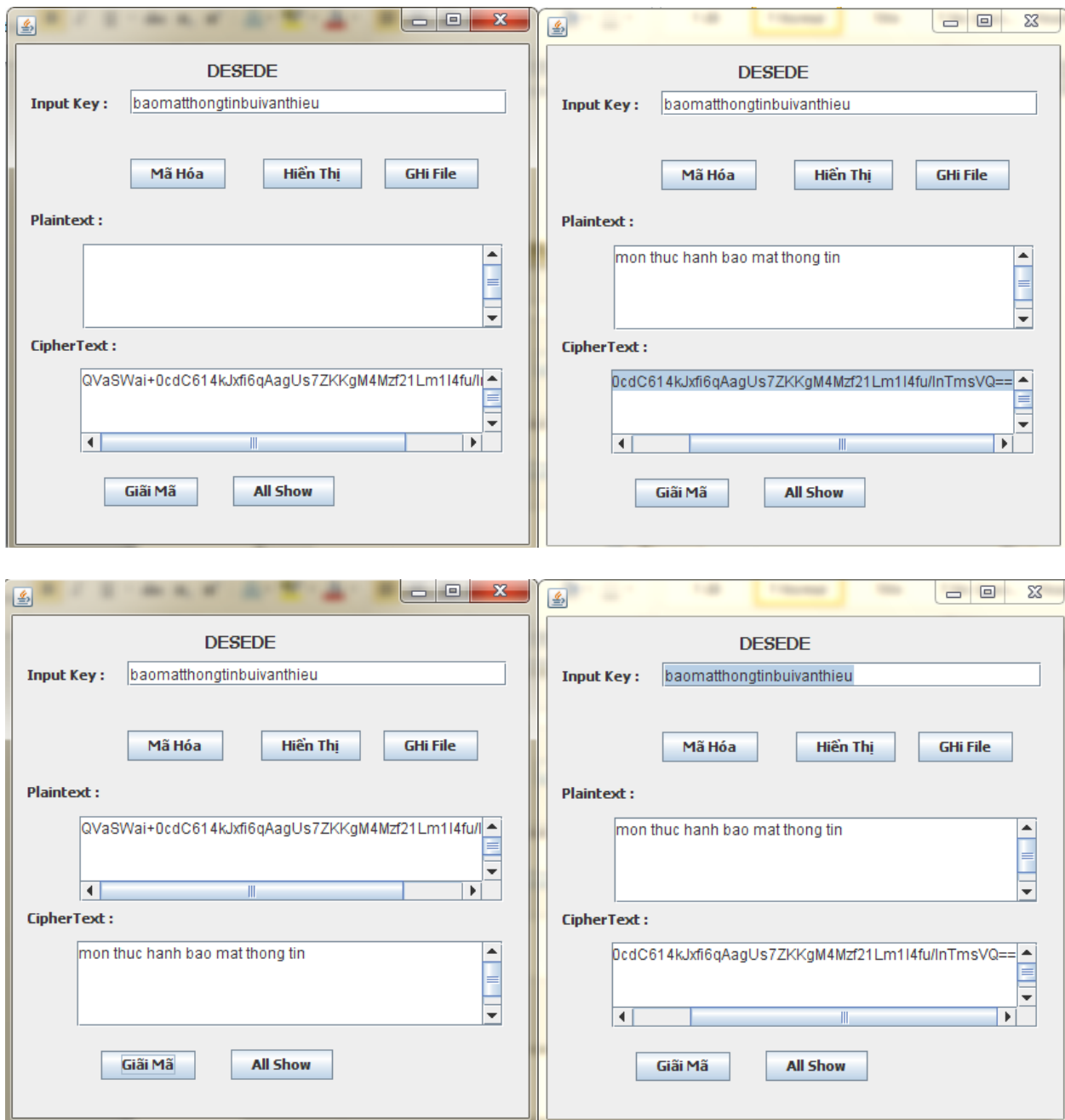
```

private void bntMaHoaActionPerformed(java.awt.event.ActionEvent evt) {
    try{
        // quot;Sanjaal.com&quot
        myEncryptionKey =txtkhoa.getText();
        myEncryptionScheme = DESEDE_ENCRYPTION_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(" khoa ma hoa k : " + " " + key);
        // sử dụng lớp DESEDE_EN
        String plainText=txtvanban.getText();
        // gọi phương thức mã hóa
        String encrypted=encrypt(plainText);
        System.out.println("Encrypted Value : " + encrypted);
        txtmahoa.setText(encrypted);

    } catch( Exception ex){}
}

```

Bước 3: Kiểm Tra



2.3 THUẬT TOÁN MÃ HÓA AES

Viết chương trình mã hóa và giải mã văn bản với thuật toán mã hóa AES.

Chương trình có thể thực hiện các chức năng sau:

Cho phép nhập văn bản vào hệ thống.

Cho phép nhập khóa bảo vệ văn bản.

Cho phép ghi File và mở File.

2.3.1 Hướng dẫn thuật toán AES

AES chỉ làm việc với khối dữ liệu 128 bit và khóa có độ dài 128, 192 hoặc 256 bit. Các khóa con sử dụng trong các chu trình được tạo ra bởi quá trình tạo khóa con Rijndael. Rijndael có thể làm việc với dữ liệu và khóa có độ dài bất kỳ là bội số của 32 bit nằm trong khoảng từ 128 tới 256 bit.

Hầu hết các phép toán trong thuật toán AES đều thực hiện trong một trường hữu hạn.

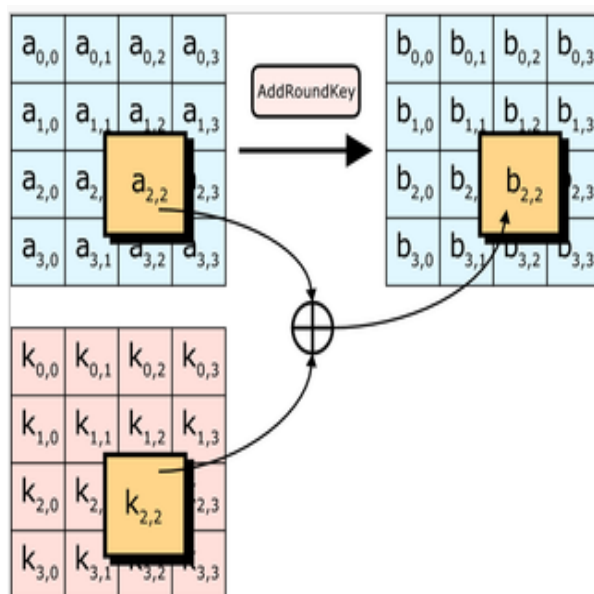
AES làm việc với từng khối dữ liệu 4×4 byte (tiếng Anh: *state*, khối trong Rijndael có thể có thêm cột). Quá trình mã hóa bao gồm 4 bước:

- AddRoundKey — mỗi byte của khối được kết hợp với khóa con, các khóa con này được tạo ra từ quá trình tạo khóa con Rijndael.
- SubBytes — đây là phép thế (phi tuyến) trong đó mỗi byte sẽ được thế bằng một byte khác theo bảng tra (Rijndael S-box).
- ShiftRows — đổi chỗ, các hàng trong khối được dịch vòng.
- MixColumns — quá trình trộn làm việc theo các cột trong khối theo một phép biến đổi tuyến tính.

Tại chu trình cuối thì bước MixColumns được thay thế bằng bước AddRoundKey

a. Bước AddRoundKey:

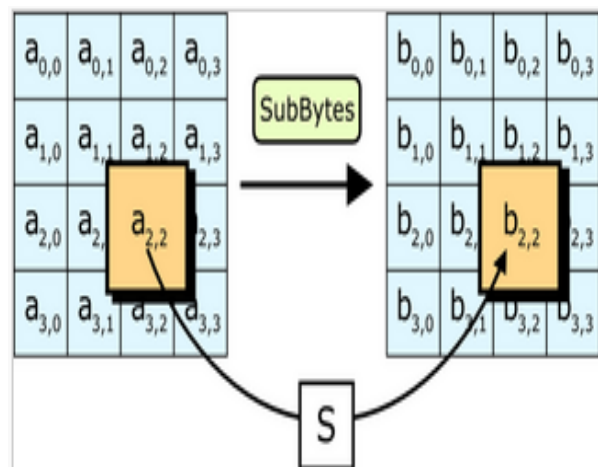
Tại bước này, khóa con được kết hợp với các khối. Khóa con trong mỗi chu trình được tạo ra từ khóa chính với quá trình tạo khóa con Rijndael; mỗi khóa con có độ dài giống như các khối. Quá trình kết hợp được thực hiện bằng cách XOR từng bit của khóa con với khối dữ liệu.



Trong bước AddRoundKey, mỗi byte được kết hợp với một byte trong khóa con của chu trình sử dụng phép toán XOR (\oplus).

b. Bước SubBytes

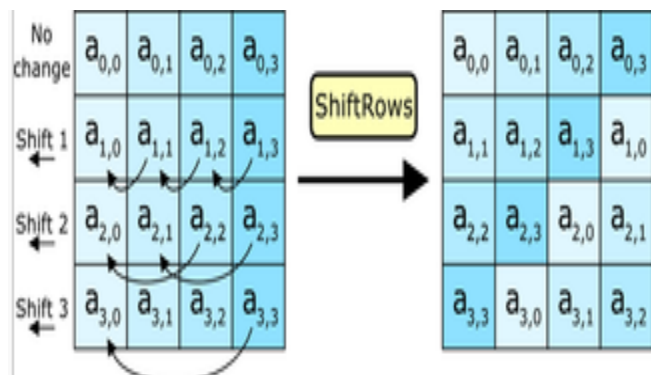
Các byte được thể thông qua bảng tra S-box. Đây chính là quá trình phi tuyến của thuật toán. Hộp S-box này được tạo ra từ một phép nghịch đảo trong trường hữu hạn GF (2^8) có tính chất phi tuyến. Để chống lại các tấn công dựa trên các đặc tính đại số, hộp S-box này được tạo nên bằng cách kết hợp phép nghịch đảo với một phép biến đổi affine khả nghịch. Hộp S-box này cũng được chọn để tránh các điểm bất động (fixed point).



Trong bước SubBytes, mỗi byte được thay thế bằng một byte theo bảng tra, S , $b_{ij} = S(a_{ij})$.

c. Bước ShiftRows

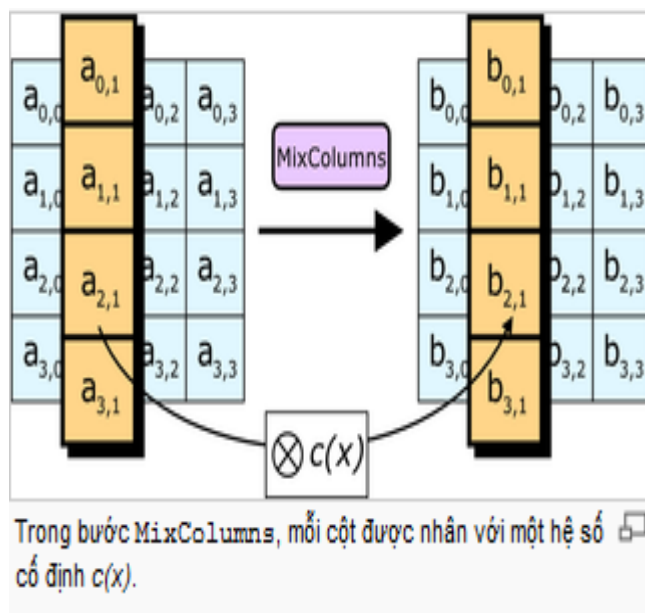
Các hàng được dịch vòng một số vị trí nhất định. Đối với AES, hàng đầu được giữ nguyên. Mỗi byte của hàng thứ 2 được dịch trái một vị trí. Tương tự, các hàng thứ 3 và 4 được dịch 2 và 3 vị trí. Do vậy, mỗi cột khối đầu ra của bước này sẽ bao gồm các byte ở đủ 4 cột khối đầu vào. Đối với Rijndael với độ dài khối khác nhau thì số vị trí dịch chuyển cũng khác nhau.



Trong bước ShiftRows, các byte trong mỗi hàng được dịch vòng trái. Số vị trí dịch chuyển tùy thuộc từng hàng.

d. Bước MixColumns

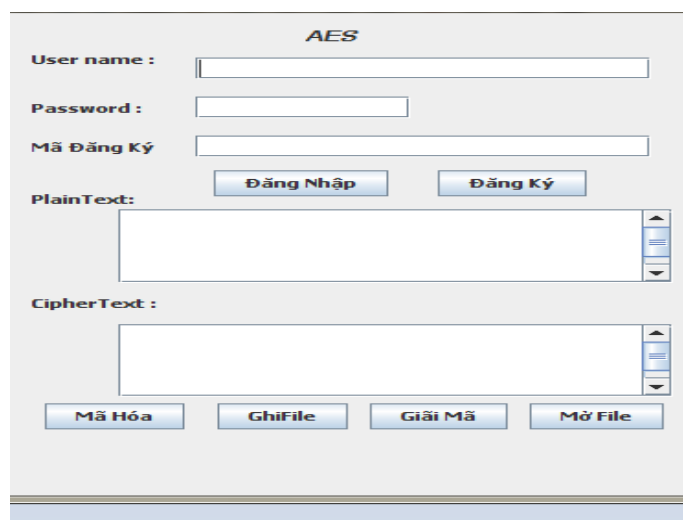
Bốn byte trong từng cột được kết hợp lại theo một phép biến đổi tuyến tính khả nghịch. Mỗi khối 4 byte đầu vào sẽ cho một khối 4 byte ở đầu ra với tính chất là mỗi byte ở đầu vào đều ảnh hưởng tới cả 4 byte đầu ra. Cùng với bước ShiftRows, MixColumns đã tạo ra tính chất khuyếch tán cho thuật toán. Mỗi cột được xem như một đa thức trong trường hữu hạn và được nhân với đa thức $c(x) = 3x^3 + x^2 + x + 2 \pmod{x^4 + 1}$. Vì thế, bước này có thể được xem là



Tóm lại: Thuật toán AES có khối dữ liệu 128 bit , độ dài khóa 128,192, 256 bit , cấu trúc mạng thay thế-hoán vị , số chu trình 10,12,14 tùy theo độ dài khóa.

2.3.2 Hướng dẫn thực hành

Bước 1: Thiết Kế Form :



Bước 2: Viết hàm xử lý sự kiện

B2.1: Viết hàm xử lý sự kiện đăng nhập


```

private void bntĐăngNhapActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        user = txtuser.getText();
        pass = txtpass.getText();
        khoa = user + pass;
        BufferedReader br = null;
        String fileName = "D:\\AES.txt"; //GEN-
        br = new BufferedReader(new FileReader(fileName));
        StringBuffer sb = new StringBuffer();
        char[] ca = new char[5];
        while (br.ready()) {
            int len = br.read(ca);
            sb.append(ca, 0, len);
        }
        br.close();
        //xuat chuoi
        System.out.println("Khoa la : " + " " + sb);
        String chuoi = sb.toString();
        Boolean k=khoa.equals(chuoi);
        if(k==true) JOptionPane.showMessageDialog(null, " Đăng Nhập Thành Công");
        else
            JOptionPane.showMessageDialog(null, " Đăng Nhập Thất bại");
        txtkhoa.setText(chuoi.getBytes().toString());
        KeyGenerator keyGen = KeyGenerator.getInstance("AES");
        keyGen.init(128);
        secretKey = keyGen.generateKey();
    } catch (NoSuchAlgorithmException ex) {
        // Logger.getLogger(FrAES.class.getName()).log(Level.SEVERE, null, ex);
    } catch (Exception ex) {}
    // Logger.getLogger(FrAES.class.getName()).log(Level.SEVERE, null, ex);
}

```

B2.2: Viết hàm xử lý sự kiện đăng ký

```

private void bntĐăngKýActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        // TODO add your handling code here:
        user = txtuser.getText();
        pass = txtpass.getText();
        khoa = user + pass;
        BufferedWriter bw = null;
        //ghi van ban da ma hoa
        String fileName = "D:\\AES.txt";
        //luu van ban
        String s = txtvanban.getText();
        bw = new // van ban sau khi ma hoa
        BufferedWriter(new FileWriter(fileName));
        // ghi van ban
        bw.write(khoa);
        bw.close();
        JOptionPane.showMessageDialog(null, "Bạn Đã Đăng Ký Thành Công .Vui lòng Đăng nhập lại !!!");
        txtkhoa.setText(khoa.getBytes().toString());
    } catch (IOException ex) {
        Logger.getLogger(FrAES.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

B2.3: Viết hàm xử lý sự kiện mã hóa

```

private void bntMahoaActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        System.out.println(" Sinh khoa: " + secretKey);
        //thuật toán AES
        Cipher aesCipher = Cipher.getInstance("AES");
        // Sinh khoa
        aesCipher.init(Cipher.ENCRYPT_MODE, secretKey);

        /**/ Mã hóa văn bản
        String strData = txtvanban.getText();
        //convert văn bản sang kiểu byte
        byte[] byteDataToEncrypt = strData.getBytes();
        // Gọi phương thức doFinal để mã hóa
        byteCipherText = aesCipher.doFinal(byteDataToEncrypt);
        String strCipherText = new BASE64Encoder().encode(byteCipherText);
        System.out.println("Cipher Text generated using AES is " + strCipherText);
        txtmahoa.setText(strCipherText);
    } catch (Exception ex) {
        System.out.println(" Lỗi mã hóa: " + ex);
    }
}

```

B2.3 Viết hàm xử lý sự kiện giải mã

```

private void bntGiaiMaActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        // TODO add your handling code here:
        // txtvanban.setText(txtmahoa.getText());
        String cipherText=txtmahoa.getText();
        txtvanban.setText(cipherText);
        Cipher aesCipher = Cipher.getInstance("AES");
        aesCipher.init(Cipher.DECRYPT_MODE, secretKey, aesCipher.getParameters());
        //String giaima =txtmahoa.getText();
        // byte[] giaima=cipherText.getBytes();
        //byteCipherText
        byte[] byteDecryptedText = aesCipher.doFinal(byteCipherText);
        String strDecryptedText = new String(byteDecryptedText);
        System.out.println(" Decrypted Text message is " + strDecryptedText);
        txtmahoa.setText(strDecryptedText);

    } catch (Exception ex) {
        System.out.println(" Lỗi giải Mã: " + ex);
    }
}

```

B2.4 Viết hàm xử lý sự kiện ghi File

```
private void bntGhiFileActionPerformed(java.awt.event.ActionEvent evt) {
    try {

        BufferedWriter bw = null;
        //ghi van ban da ma hoa
        String fileName = "D:\\GhiAES.txt";
        //luu van ban
        String s = txtmahoa.getText();
        bw = new // van ban sau khi ma hoa
        BufferedWriter(new FileWriter(fileName));
        // ghi van ban
        bw.write(s);
        bw.close();
        JOptionPane.showMessageDialog(null, " Đã ghi file D:\\GhiAES.txt");
    } catch (IOException ex) {
        Logger.getLogger(FrAES.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

B2.5 Viết hàm xử lý sự kiện mở File

```
private void bntMoFileActionPerformed(java.awt.event.ActionEvent evt) {
    try {

        BufferedReader br = null;

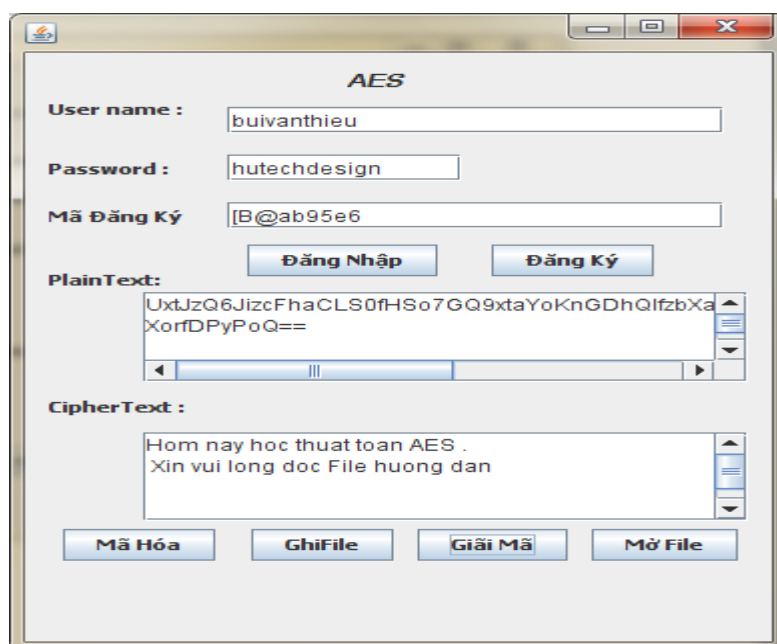
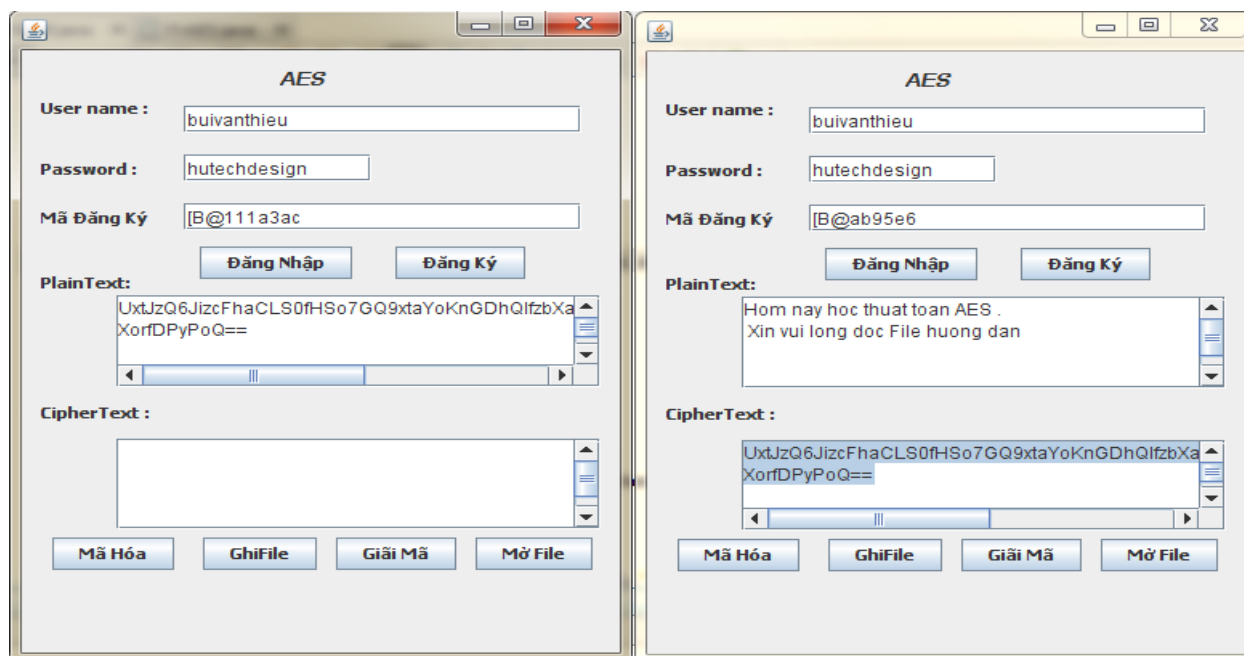
        String fileName = "D:\\GhiAES.txt"; //GEN-
        br = new BufferedReader(new FileReader(fileName));
        StringBuffer sb = new StringBuffer();

        JOptionPane.showMessageDialog(null, " Đã mở file");
        char[] ca = new char[5];
        while (br.ready()) {
            int len = br.read(ca);
            sb.append(ca, 0, len);
        }
        br.close();
        //xuất chuỗi
        System.out.println("Du Lieu la : " + " " + sb);
        String chuoi = sb.toString();

        txtvanban.setText(chuoi);
        // -----
        bntGiaiMa.enable(true);

    } catch (IOException ex) {
        Logger.getLogger(FrAES.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Bước 3: Kiểm Tra



Bài Tập mở rộng: Viết phần mềm mã hóa văn bản với các thuật toán mã hóa trên.

Chương trình có thể thực hiện các chức năng sau:

Cho phép nhập văn bản vào hệ thống.

Cho phép nhập khóa bảo vệ văn bản.

Cho phép mở File và Ghi File.

Cho phép bên gửi mã hóa dữ liệu và bên nhận mã hóa dữ liệu với khóa K. (sinh viên nghiên cứu viết thêm chức năng..)

BÀI 3. MÃ HÓA KHÓA CÔNG KHAI

RSA

3.1 GIỚI THIỆU THUẬT TOÁN RSA

RSA là một thuật toán mật mã hoá khoá công khai. Đây là thuật toán đầu tiên phù hợp với việc tạo ra chữ ký điện tử đồng thời với việc mã hoá. Nó đánh dấu một sự tiến bộ vượt bậc của lĩnh vực mật mã học trong việc sử dụng khoá công cộng

Giả sử Alice và Bob cần trao đổi thông tin bí mật thông qua một kênh không an toàn (ví dụ như Internet). Với thuật toán RSA, Alice đầu tiên cần tạo ra cho mình cặp khóa gồm khóa công khai và khóa bí mật theo các bước sau

1. Chọn 2 [số nguyên tố](#) lớn p và q với $p \neq q$, lựa chọn ngẫu nhiên và độc lập.
2. Tính: $n = p \cdot q$
3. Tính: giá trị hàm số Euler.

$$\phi(n) = (p - 1)(q - 1)$$

4. Chọn một số tự nhiên e sao cho

$$1 < e < \phi(n)$$

và là số nguyên tố cùng nhau với $\phi(n)$

5. Tính: d sao cho.

$$de \equiv 1 \pmod{\phi(n)}$$

Khóa công khai bao gồm:

- n , môđun, và
- e , số mũ công khai (cũng gọi là *số mũ mã hóa*).
- Khóa bí mật bao gồm:
- n , môđun, xuất hiện cả trong khóa công khai và khóa bí mật, và

- o d , số mũ bí mật (cũng gọi là *số mũ giải mã*).

Ví dụ: Giả sử Bob muốn gửi đoạn thông tin M cho Alice. Đầu tiên Bob chuyển M thành một số $m < n$ theo một hàm có thể đảo ngược (từ m có thể xác định lại M) được thỏa thuận trước.

Lúc này Bob có m và biết n cũng như e do Alice gửi. Bob sẽ tính c là bản mã hóa của m theo công thức: $c = m^e \mod n$

Cuối cùng Bob gửi c cho Alice. Alice nhận c từ Bob và biết khóa bí mật d . Alice có thể tìm được m từ c theo công thức sau:

$$m \equiv c^d \mod n$$

Biết m , Alice tìm lại M theo phương pháp đã thỏa thuận trước. Quá trình giải mã hoạt động vì ta có $c^d \equiv (m^e)^d \equiv m^{ed} \pmod{n}$

. Do $ed \equiv 1 \pmod{p-1}$ và $ed \equiv 1 \pmod{q-1}$, (theo [Định lý Fermat nhỏ](#)) nên: $m^{ed} \equiv m \pmod{p}$

$$\text{Và } m^{ed} \equiv m \pmod{q}$$

Do p và q là hai số nguyên tố cùng nhau, ta có: $m^{ed} = m \pmod{pq}$

hay: $c^d = m \pmod{n}$

Ví dụ 2:

Ta chọn hai số nguyên tố p và q , với $p = 5$ và $q = 7$

Tính $n = p \cdot q = 5 \cdot 7 = 35$.

$Z = (p-1) \cdot (q-1) = (5-1)(7-1) = 24$

Tiếp đến chọn e thỏa $1 < e < n$

→ chọn $e = 5$.

Tìm d , sao cho $e \cdot d - 1$ chia hết cho z (24)

→ chọn $d = 29$.

Do đó: Public key $= (n, e) = (35, 5)$

Private key = (n,d) =(35,29)

Mã hoá chuỗi sau:secure

Ta có bảng sau

<u>Nội dung bị mã hoá</u>	$M = c^d \bmod n$	<u>Dữ liệu gốc</u>
24	19	S
10	5	E
33	3	C
21	21	u
23	18	R
10	5	e

3.2 HƯỚNG DẪN THỰC HÀNH

Lớp RSA cài đặt thuật toán mã hóa bất đối xứng RSA như sau:

```
import java.math.BigInteger ;
import java.util.Random ;
import java.io.* ;
public class RSA
{
    /**
     * Bit length of each prime number.
     */
    int primeSize ;

    /**
     * Two distinct large prime numbers p and q.
     */
    BigInteger p, q ;

    /**
     * Modulus N.
     */
    BigInteger N ;
```



```
/**
 * r = ( p - 1 ) * ( q - 1 )
 */
BigInteger r ;

/**
 * Public exponent E and Private exponent D
 */
BigInteger E, D ;
Thêm Random
Random random = new Random();
public RSA() {

}

/**
 * Constructor.
 *
 * @param primeSize      Bit length of each prime number.
 */

public RSA( int primeSize )
{
    this.primeSize = primeSize ;

    // Generate two distinct large prime numbers p and q.
    generatePrimeNumbers() ;

    // Generate Public and Private Keys.
    generatePublicPrivateKeys() ;
}

/**
 * Generate two distinct large prime numbers p and q.
```

```
*/  
  
public void generatePrimeNumbers()  
{  
    p = BigInteger.probablePrime(primeSize /2, random);  
  
    do  
    {  
        q = BigInteger.probablePrime(primeSize /2, random);  
    }  
    while( q.compareTo( p ) == 0 ) ;  
}  
  
/**  
 * Generate Public and Private Keys.  
 */  
public void generatePublicPrivateKeys()  
{  
    // N = p * q  
    N = p.multiply( q ) ;  
  
    // r = ( p - 1 ) * ( q - 1 )  
    r = p.subtract( BigInteger.valueOf( 1 ) ) ;  
    r = r.multiply( q.subtract( BigInteger.valueOf( 1 ) ) ) ;  
  
    // Choose E, coprime to and less than r  
    do  
    {  
        E = new BigInteger( 2 * primeSize, new Random() ) ;  
    }  
    while( ( E.compareTo( r ) != -1 ) ||  
           ( E.gcd( r ).compareTo( BigInteger.valueOf( 1 ) ) != 0 ) ) ;
```

```
// Compute D, the inverse of E mod r
D = E.modInverse( r ) ;
}

/**
 * Encrypts the plaintext (Using Public Key).
 *
 * @param message String containing the plaintext message to
 * be encrypted.
 * @return The ciphertext as a BigInteger array.
 */
public BigInteger[] encrypt( String message )
{
    int i ;
    byte[] temp = new byte[1] ;

    byte[] digits = message.getBytes() ;

    BigInteger[] bigdigits = new BigInteger[digits.length] ;

    for( i = 0 ; i < bigdigits.length ; i++ )
    {
        temp[0] = digits[i] ;
        bigdigits[i] = new BigInteger( temp ) ;
    }

    BigInteger[] encrypted = new BigInteger[bigdigits.length] ;

    for( i = 0 ; i < bigdigits.length ; i++ )
        encrypted[i] = bigdigits[i].modPow( E, N ) ;

    return( encrypted ) ;
}
```

```

    }

    public BigInteger[] encrypt( String message, BigInteger userD, BigInteger
userN)
    {
        int i ;
        byte[] temp = new byte[1] ;

        byte[] digits = message.getBytes() ;

        BigInteger[] bigdigits = new BigInteger[digits.length] ;

        for( i = 0 ; i < bigdigits.length ; i++ )
        {
            temp[0] = digits[i] ;
            bigdigits[i] = new BigInteger( temp ) ;
        }

        BigInteger[] encrypted = new BigInteger[bigdigits.length] ;

        for( i = 0 ; i < bigdigits.length ; i++ )
            encrypted[i] = bigdigits[i].modPow( userD, userN ) ;

        return( encrypted ) ;
    }
/**
 * Decrypts the ciphertext (Using Private Key).
 *
 * @param encrypted      BigInteger array containing
 * the ciphertext to be decrypted.
 * @return The decrypted plaintext.
 */
    public String decrypt( BigInteger[] encrypted, BigInteger D, BigInteger N
)

```

```
{  
    int i ;  
  
    BigInteger[] decrypted = new BigInteger[encrypted.length] ;  
  
    for( i = 0 ; i < decrypted.length ; i++ )  
        decrypted[i] = encrypted[i].modPow( D, N ) ;  
  
    char[] charArray = new char[decrypted.length] ;  
  
    for( i = 0 ; i < charArray.length ; i++ )  
        charArray[i] = (char) ( decrypted[i].intValue() ) ;  
  
    return( new String( charArray ) ) ;  
}  
  
/**  
 * Get prime number p.  
 *  
 * @return Prime number p.  
 */  
public BigInteger getp()  
{  
    return( p ) ;  
}  
  
/**  
 * Get prime number q.  
 *  
 * @return Prime number q.  
 */  
public BigInteger getq()
```

```
{  
    return( q ) ;  
}  
  
/**  
 * Get r.  
 *  
 * @return r.  
 */  
public BigInteger getr()  
{  
    return( r ) ;  
}  
  
/**  
 * Get modulus N.  
 *  
 * @return Modulus N.  
 */  
public BigInteger getN()  
{  
    return( N ) ;  
}  
  
/**  
 * Get Public exponent E.  
 *  
 * @return Public exponent E.  
 */  
public BigInteger getE()  
{  
    return( E ) ;  
}
```

```
/**
 * Get Private exponent D.
 *
 * @return Private exponent D.
 */
public BigInteger getD()
{
    return( D ) ;
}

/**
 * RSA Main program for Unit Testing.
 */
public static void main( String[] args ) throws IOException
{
    /*if( args.length != 1 )
    {
        System.out.println( "Syntax: java RSA PrimeSize" ) ;
        System.out.println( "e.g. java RSA 8" ) ;
        System.out.println( "e.g. java RSA 512" ) ;

        System.exit( -1 ) ;
    }

    // Get bit length of each prime number
    int primeSize = Integer.parseInt( args[0] ) ;*/
    int primeSize =8;

    // Generate Public and Private Keys

    RSA rsa = new RSA( primeSize ) ;
```

```
        System.out.println( "Key Size: [" + primeSize + "]" );
        System.out.println( "" ) ;

        System.out.println( "Generated prime numbers p and q" );
        System.out.println(      "p:      ["      +      rsa.getp().toString(      16
).toUpperCase() + "]" );
        System.out.println(      "q:      ["      +      rsa.getq().toString(      16
).toUpperCase() + "]" );
        System.out.println( "" ) ;

        System.out.println( "The public key is the pair (N, E) which will
be published." ) ;
        System.out.println(      "N:      ["      +      rsa.getN().toString(      16
).toUpperCase() + "]" ) ;
        System.out.println(      "E:      ["      +      rsa.getE().toString(      16
).toUpperCase() + "]" ) ;
        System.out.println( "" ) ;

        System.out.println( "The private key is the pair (N, D) which will
be kept private." ) ;
        System.out.println(      "N:      ["      +      rsa.getN().toString(      16
).toUpperCase() + "]" ) ;
        System.out.println(      "D:      ["      +      rsa.getD().toString(      16
).toUpperCase() + "]" ) ;
        System.out.println( "" ) ;

        // Get message (plaintext) from user
        System.out.println( "Please enter message (plaintext):" ) ;
        String plaintext = ( new BufferedReader( new InputStreamReader(
System.in ) ) ).readLine() ;
        System.out.println( "" ) ;

        // Encrypt Message
        BigInteger[] ciphertext = rsa.encrypt( plaintext ) ;
```



```
System.out.print( "Ciphertext: [" ) ;
for( int i = 0 ; i < ciphertext.length ; i++ )
{
    System.out.print( ciphertext[i].toString( 16 ).toUpperCase()
) ;

    if( i != ciphertext.length - 1 )
        System.out.print( " " ) ;
}
System.out.println( "]" ) ;
System.out.println( "" ) ;

RSA rsa1 = new RSA(8);

String    recoveredPlaintext    =    rsa1.decrypt(    ciphertext
,rsa.getD(),rsa.getN()) ;

System.out.println( "Recovered plaintext: [" + recoveredPlaintext
+ "]" ) ;
}
}
```

Bài tập hướng dẫn: Mã hóa và giải mã văn bản sử dụng thuật toán RSA.

```

public class PwdEncryption{

    public static void main(String args[]){
        Scanner in = new Scanner(System.in);
        String nhash;
        BigInteger[] ciphertext = null;
        BigInteger n = null;
        BigInteger d = null;
        String password="";
        System.out.println("Enter text to be encrypted:");
        password=in.nextLine();

        System.out.println("Password (Input) : "+ password);
        //8FD1 5FB3 5057 5FB3 65E63AB879713ABB

        RSA rsa = new RSA( 8 ) ;
        n=rsa.getN();
        d=rsa.getD();
        ciphertext = rsa.encrypt(password) ;

        StringBuffer bf = new StringBuffer();
        for( int i = 0 ; i < ciphertext.length ; i++ )
        {
            bf.append( ciphertext[i].toString( 16 ).toUpperCase() ) ;

            if( i != ciphertext.length - 1 )
                System.out.print( " " ) ;
        }

        String message=bf.toString();
        System.out.println();
        System.out.println("Encrypted Message : "+message);
        3D1D2E

        String dhash = rsa.decrypt( ciphertext ,d,n) ;
        System.out.println();
        System.out.println("Decrypted Message : "+dhash);
    }
}

```

Kết quả chương trình:

```
Enter text to be encrypted:
Hutech technology
Password (Input) : Hutech technology

Encrypted Message : 3874721B80BE3AE0396F7FF7FCD80BE3AE0396F7FF4FE4365978F4365986B9DA9

Decrypted Message : Hutech technology
BUILD SUCCESSFUL (total time: 23 seconds)
```

Bài Tập NC : Viết chương trình mã hóa và giải mã sử dụng thuật toán RSA. Yêu cầu chương trình:

Thiết kế form chứa thông tin sau: Họ và tên đầy đủ, địa chỉ địa chỉ email, số điện thoại, nhập password có ký tự đặc biệt.

Viết hàm xử lý mã hóa và giải mã.

3.3 BÀI TẬP NÂNG CAO

Viết chương trình mã hóa và giải mã sử dụng thuật toán RSA:

Viết hàm tạo khóa Private key và Public key.

Viết hàm mã hóa sử dụng Publickey để mã hóa văn bản.

Viết hàm giải mã sử dụng private key để giải mã văn bản.

Hướng dẫn:

- Hàm tạo khóa

```
public class Skey_RSA{
    public static void main(String args[]) throws Exception{
        // KeyPairGenerator: giúp tạo ra các cặp key
        KeyPairGenerator kpg=KeyPairGenerator.getInstance("RSA");
        kpg.initialize(1024);
        // public/private keypairs được dùng để khởi tạo phase của
        // quá trình đăng ký key
        KeyPair kp=kpg.genKeyPair();
        PublicKey pbkey=kp.getPublic();
        PrivateKey prkey=kp.getPrivate();
        //Ghi file publickey
        FileOutputStream f1=new FileOutputStream("D:\\Skey_RSA_pub.dat");
        ObjectOutputStream b1=new ObjectOutputStream(f1);
        b1.writeObject(pbkey);
        // ghi file private key
        FileOutputStream f2=new FileOutputStream("D:\\Skey_RSA_priv.dat");
        ObjectOutputStream b2=new ObjectOutputStream(f2);
        b2.writeObject(prkey);

    }
}
```

- Hàm mã hóa

```

public class Enc_RSA{
    public static void main(String args[]) throws Exception{
        // chuỗi cần mã hóa
        String s="Hello World!";
        //doc file public key
        FileInputStream f=new FileInputStream("D:\\Skey_RSA_pub.dat");
        ObjectInputStream b=new ObjectInputStream(f);
        //Sử dụng hàm readObject của ObjectInputStream
        //để đọc dữ liệu từ tập tin nhị phân lên.
        //Thứ tự đọc cần đảm bảo đúng với thứ tự ghi
        RSAPublicKey pbk=(RSAPublicKey)b.readObject( );
        BigInteger e=pbk.getPublicExponent();
        BigInteger n=pbk.getModulus();
        System.out.println("e= "+e);
        System.out.println("n= "+n);
        byte ptext[]=s.getBytes("UTF8");
        BigInteger m=new BigInteger(ptext);
        BigInteger c=m.modPow(e,n);
        System.out.println("c= "+c);
        String cs=c.toString( );
        BufferedWriter out=
            new BufferedWriter(new OutputStreamWriter(
                new FileOutputStream("D:\\Enc_RSA.dat")));
        out.write(cs,0,cs.length( ));
        out.close( );

    }
}

```

- Hàm giải mã

```

public class Dec_RSA{
    public static void main(String args[]) throws Exception{
        //doc van ban da ma hoa
        BufferedReader in=
            new BufferedReader(new InputStreamReader
                (new FileInputStream("D:\\Enc_RSA.dat")));
        String ctext=in.readLine();
        // chuyển sang kiểu biginteger
        BigInteger c=new BigInteger(ctext);
        //doc khóa private key
        FileInputStream f=new FileInputStream("D:\\Skey_RSA_priv.dat");
        //Sử dụng hàm readObject của ObjectInputStream
        //để đọc dữ liệu từ tập tin nhị phân lên.
        //Thứ tự đọc cần đảm bảo đúng với thứ tự ghi
        ObjectInputStream b=new ObjectInputStream(f);
        RSAPrivateKey prk=(RSAPrivateKey)b.readObject( );
        BigInteger d=prk.getPrivateExponent();
        BigInteger n=prk.getModulus();
        System.out.println("d= "+d);
        System.out.println("n= "+n);
        BigInteger m=c.modPow(d,n);
        System.out.println("m= "+m);
        byte[] mt=m.toByteArray();
        System.out.println("PlainText is ");
        for(int i=0;i<mt.length;i++){
            System.out.print((char) mt[i]);
        }
    }
}

```

Bài tập mở rộng: Viết chương trình như trên nhưng thiết kế trên Jfame Form và viết hàm xử lý.

BÀI 4. QUẢN LÝ KHÓA DÙNG MÃ KHÓA CÔNG KHAI

Viết chương trình trao đổi dữ liệu theo mô hình Client-Server theo yêu cầu sau:

1.1 Sử dụng thuật toán Diffie-Hellman trao đổi quá cho nhau và sinh ra cặp khóa chung .

1.2 Dùng khóa chung mã hóa và giải mã văn bản (văn bản được lưu ở dạng .doc, .dat,...)

4.1 HƯỚNG DẪN THUẬT TOÁN TRAO ĐỔI KHÓA DIFFIE-HELLMAN

Diffie-Hellman là một thuật toán dùng để trao đổi khóa chứ không dùng để bảo vệ tính bí mật của dữ liệu. Tuy nhiên, Diffie-Hellman lại có ích trong giai đoạn trao đổi khóa bí mật của các thuật toán mật mã đối xứng.

Thuật toán trao đổi khóa Diffie-Hellman dựa trên phép logarit rời rạc. Cho trước một số g và $x = g^k$, để tìm k ta thực hiện phép logarit : $k = \log_g(x)$. Tuy nhiên, nếu cho trước g , n và $(g^k \bmod n)$, thì quá trình xác định k được thực hiện theo cách khác với cách ở trên và được gọi là logarit rời rạc.

Thuật toán Diffie-Hellman được mô tả như sau:

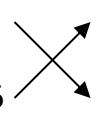
Gọi n là một số nguyên tố lớn và g là một cơ số sinh (generator, số nguyên nhỏ) thỏa điều kiện: với mọi $x \in \{1, 2, \dots, n-1\}$, ta luôn tìm được số y sao cho $x = g^y \bmod n$.

Giá trị n và g được phổ biến công khai giữa các thực thể trao đổi khóa. Sau đó user A tạo ra một số riêng $X_a < n$, tính giá trị $Y_a = (g^{X_a} \bmod n)$ và gửi cho B. Tương tự, user B cũng tạo ra một số riêng $X_b < n$ tính giá trị $Y_b = (g^{X_b} \bmod n)$ và gửi lại cho A. X_a và X_b tương đương khóa private, Y_a và Y_b tương đương khóa public.

User B xác định được khóa bí mật dùng cho phiên làm việc bằng cách tính giá trị $(g^{x_a} \bmod n)^{x_b} = (g^{x_a x_b} \bmod n)$. Bằng cách tương tự, user A cũng xác định được cùng khóa bí mật này bằng cách tính giá trị $(g^{x_b} \bmod n)^{x_a} = (g^{x_a x_b} \bmod n)$.

Giả sử trong quá trình trao đổi các giá trị, phía tấn công bắt được $(g^{x_a} \bmod n)$ và $(g^{x_b} \bmod n)$, họ rất khó xác định được X_a và X_b vì độ phức tạp của phép toán logarit rời rạc là rất cao.

Ví dụ: với $n=31$ và $g=3$

A: $x=8 \Rightarrow y=3^8 \bmod 31 = 20$  $k = 16^8 \bmod 31 = 4$

B: $x=6 \Rightarrow y=3^6 \bmod 31 = 16$ $k = 20^6 \bmod 31 = 4$

Khóa bí mật không được tạo trước và chuyển từ A sang B hoặc ngược lại, khóa bí mật chỉ được tạo ra sau khi A và B trao đổi với nhau Y_a và Y_b . Vì vậy khóa bí mật này thường được gọi là khóa phiên.

4.2 HƯỚNG DẪN THỰC HÀNH

4.2.1 Tạo Class CryptoUtil viết phương thức sau

```
public static final String toHexString(byte[] block)
{
    StringBuffer buf = new StringBuffer();
    int len = block.length;

    for (int i = 0; i < len; i++)
    {
        byte2hex(block[i], buf);
        if (i < len-1)
        {
            buf.append(":");
        }
    }
    return buf.toString();
}
```

Hàm này dùng để chuyển từ kiểu hex sang kiểu String.


```

public static final void byte2hex(byte b, StringBuffer buf)
{
    char[] hexChars = { '0', '1', '2', '3',
                          '4', '5', '6', '7',
                          '8', '9', 'A', 'B',
                          'C', 'D', 'E', 'F' };

    int high = (b & 0xf0) >> 4;
    int low = (b & 0x0f);
    buf.append(hexChars[high]);
    buf.append(hexChars[low]);
}

```

4.2.2 Bên Alice

Thiết kế form Alice

Viết hàm xử lý sự kiện Tạo Khóa A

Các biến cần khai báo bên Alice

```

KeyAgreement aliceKeyAgree;
PublicKey bobPubKey;
SecretKey aliceDesKey;
Cipher aliceCipher;

```

```

private void bntkhoaAActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try{
        AlgorithmParameterGenerator paramGen = AlgorithmParameterGenerator.getInstance("DH");
        paramGen.init(512);
        AlgorithmParameters params = paramGen.generateParameters();

        DHParameterSpec dhSkipParamSpec =
            (DHParameterSpec) params.getParameterSpec(DHParameterSpec.class);

        System.out.println("Generating a DH KeyPair...");
        KeyPairGenerator aliceKpairGen = KeyPairGenerator.getInstance("DH");
        aliceKpairGen.initialize(dhSkipParamSpec);
        KeyPair aliceKpair = aliceKpairGen.generateKeyPair();

        System.out.println("Initializing the KeyAgreement Engine with DH private key");
        aliceKeyAgree = KeyAgreement.getInstance("DH");
        aliceKeyAgree.init(aliceKpair.getPrivate());

        byte[] alicePubKeyEnc = aliceKpair.getPublic().getEncoded();
        FileOutputStream fos=new FileOutputStream("D:/A.pub");
        fos.write(alicePubKeyEnc);
        fos.close();
        txtkhoaA.setText(alicePubKeyEnc.toString());

    }catch(Exception ex){}
}

```

Viết hàm xử lý sự kiện Hiện Thị KB

```

private void bntkhoaBActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try{
        FileInputStream fis=new FileInputStream("D:/B.pub");
        byte[] bkeyP=new byte[fis.available()];
        fis.read(bkeyP);
        fis.close();
        txtkhoaB.setText(bkeyP.toString());
    }
    catch(Exception ex){
    }
}

```

Viết hàm xử lý sự kiện Khóa Chung

```
private void bntkhoaABActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    try{  
        FileInputStream fis=new FileInputStream("D:/B.pub");  
        byte[] bobPubKeyEnc=new byte[fis.available()];  
        fis.read(bobPubKeyEnc);  
        fis.close();  
  
        KeyFactory aliceKeyFac = KeyFactory.getInstance("DH");  
        X509EncodedKeySpec x509KeySpec = new X509EncodedKeySpec(bobPubKeyEnc);  
        bobPubKey = aliceKeyFac.generatePublic(x509KeySpec);  
        System.out.println("Executing PHASE1 of key agreement...");  
        aliceKeyAgree.doPhase(bobPubKey, true);  
        byte[] aliceSharedSecret = aliceKeyAgree.generateSecret();  
  
        System.out.println("khoa chung: secret (DEBUG ONLY):" + CryptoUtil.toHexString(aliceSharedSecret));  
        txtkhoachung.setText(CryptoUtil.toHexString(aliceSharedSecret));  
  
    }  
    catch(Exception ex){}
```

Viết hàm xử lý sự kiện Mã Hóa KAB

```
private void bntmahoakabActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    //ma hoa khoa chung bang thuat toan DES
    try{
        aliceKeyAgree.doPhase(bobPubKey, true);
        aliceDesKey = aliceKeyAgree.generateSecret("DES");
        txtmahoakab.setText(aliceDesKey.toString());
        // khoa chung A-B
        BufferedWriter bw = null;
        //ghi van ban da ma hoa
        String fileName = "D:\\KhoaA.txt";
        //luu van ban
        bw = new // van ban sau khi ma hoa
        BufferedWriter(new FileWriter(fileName));
        // ghi van ban
        bw.write(aliceDesKey.toString());
        bw.close();

    }catch(Exception ex){}
}
```

Viết hàm xử lý sự kiện Mã Hóa/ Giải mã

```
private void bntmahoagiainaActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    DESCS des= new DESCS();
    des.setVisible(true);
}
```

4.2.3 Bên Bob

Thiết kế form Bob

Bob

Khóa BOB :

Khóa Alice :

Khóa KAB:

Mã Hóa KAB:

Các biến cần khai báo cho bob

```
KeyAgreement bobKeyAgree;  
PublicKey alicePubKey ;  
SecretKey bobDesKey;  
Cipher bobCipher;
```

Viết hàm xử lý sự kiện Hiển Thị Khóa KA(Alice)

```
private void bntkhoaAActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    try{  
        FileInputStream fis=new FileInputStream("D:/A.pub");  
        byte[] akeyP=new byte[fis.available()];  
        fis.read(akeyP);  
        fis.close();  
        txtkhoaA.setText(akeyP.toString());  
    }  
    catch(Exception ex){  
    }  
}
```

Viết hàm xử lý sự kiện Tạo khóa KB (bob)

```
private void bntkhoaBActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    try{  
        boolean read=false;  
        //doi cho toi khi co tap tin alice.pub  
        while(!read){  
            try{  
                FileInputStream fis=new FileInputStream("D:/A.pub");  
                fis.close();  
                read=true;  
            }catch(Exception ex){}  
        }  
    }  
}
```



```

FileInputStream fis=new FileInputStream("D:/A.pub");
byte[] alicePubKeyEnc=new byte[fis.available()];
fis.read(alicePubKeyEnc);
fis.close();
KeyFactory bobKeyFac = KeyFactory.getInstance("DH");
X509EncodedKeySpec x509KeySpec = new X509EncodedKeySpec(alicePubKeyEnc);
alicePubKey = bobKeyFac.generatePublic(x509KeySpec);
DHParameterSpec dhParamSpec = ((DHPublicKey) alicePubKey).getParams();
System.out.println("Generate DH keypair ...");
KeyPairGenerator bobKpairGen = KeyPairGenerator.getInstance("DH");
bobKpairGen.initialize(dhParamSpec);
KeyPair bobKpair = bobKpairGen.generateKeyPair();
System.out.println("Initializing KeyAgreement engine...");
bobKeyAgree = KeyAgreement.getInstance("DH");
bobKeyAgree.init(bobKpair.getPrivate());
byte[] bobPubKeyEnc = bobKpair.getPublic().getEncoded();
FileOutputStream fos=new FileOutputStream("D:/B.pub");
fos.write(bobPubKeyEnc);
fos.close();
txtkhoab.setText(bobPubKeyEnc.toString());
} catch (Exception ex) {}
}

```

Viết hàm xử lý sự kiện Khóa Chung và sự kiện Mã Hóa KAB

```

private void bntkhoabActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try{
        bobKeyAgree.doPhase(alicePubKey, true);
        byte[] bobSharedSecret = bobKeyAgree.generateSecret();
        System.out.println("Khoa chung :Shared secret (DEBUG ONLY): " + CryptoUtil.toHexString(bobSharedSecret));
        txtkhoachung.setText(CryptoUtil.toHexString(bobSharedSecret));
    }
    catch (Exception ex){}
}

private void bntmahokabActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try{
        bobKeyAgree.doPhase(alicePubKey, true);
        bobDesKey = bobKeyAgree.generateSecret("DES");
        txtmahokab.setText(bobDesKey.toString());
        // khoa chung A-B
        BufferedWriter bw = null;
        //ghi van ban da ma hoa
        String fileName = "D:\\KhoaB.txt";
        //luu van ban
        bw = new // van ban sau khi ma hoa
        BufferedWriter(new FileWriter(fileName));
        // ghi van ban
        bw.write(bobDesKey.toString());
        bw.close();
    } catch (Exception ex){}
}

```

Viết hàm xử lý sự kiện Mã Hóa/ Giải mã

```
private void bntmahoaActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    DESCS des= new DESCS();
    des.setVisible(true);
}
```

4.2.4 Viết Frame DESCS

Thiết kế form sau

Viết chức năng Mã Hóa

```
private void bntMaHoaActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        String key=txtkhoa.getText() ; // needs to be at least 8 characters for DES

        FileInputStream fis = new FileInputStream("D:\\Des.txt");
        FileOutputStream fos = new FileOutputStream("D:\\EnDes.txt");
        encrypt(key, fis, fos);
        JOptionPane.showMessageDialog(null, " Đã mã hóa văn bản");
        //FileInputStream fis2 = new FileInputStream("D:\\encrypted.txt");
        //FileOutputStream fos2 = new FileOutputStream("D:\\decrypted.txt");
        //decrypt(key, fis2, fos2);
    } catch (Throwable e) {
        e.printStackTrace();
    }
}
```

Viết chức năng mở khóa A

```

private void bntMoKhoaAActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        BufferedReader br = null;

        String fileName = "D:\\KhoaA.txt"; //GEN-
        br = new BufferedReader(new FileReader(fileName));
        StringBuffer sb = new StringBuffer();

        JOptionPane.showMessageDialog(null, " Đã mở file");
        char[] ca = new char[5];
        while (br.ready()) {
            int len = br.read(ca);
            sb.append(ca, 0, len);
        }
        br.close();
        //xuất chuỗi
        System.out.println("Du Lieu la : " + " " + sb);
        String chuoi = sb.toString();

        txtkhoa.setText(chuoi);
    } catch (IOException ex) {
        Logger.getLogger(DESCS.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

Viết chức năng mở khóa B

```

private void bntMoKhoaBActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        BufferedReader br = null;

        String fileName = "D:\\KhoaB.txt"; //GEN-
        br = new BufferedReader(new FileReader(fileName));
        StringBuffer sb = new StringBuffer();

        JOptionPane.showMessageDialog(null, " Đã mở file");
        char[] ca = new char[5];
        while (br.ready()) {
            int len = br.read(ca);
            sb.append(ca, 0, len);
        }
        br.close();
        //xuất chuỗi
        System.out.println("Du Lieu la : " + " " + sb);
        String chuoi = sb.toString();

        txtkhoa.setText(chuoi);
    } catch (IOException ex) {
        Logger.getLogger(DESCS.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

Viết chức năng ghi File


```

private void bntGhiFileActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        BufferedWriter bw = null;
        //ghi van ban da ma hoa
        String fileName = "D:\\Des.txt";
        //luu van ban
        String s = txtvanban.getText();
        bw = new // van ban sau khi ma hoa
        BufferedWriter(new FileWriter(fileName));
        // ghi van ban
        bw.write(s);
        bw.close();
        JOptionPane.showMessageDialog(null, "Đã ghi file");
        txtmahoa.setText(s);
    } catch (IOException ex) {
        Logger.getLogger(DESCS.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

Viết chức năng giải mã

```

private void bntGiaiMaActionPerformed(java.awt.event.ActionEvent evt) {
    FileInputStream fis2 = null;
    try {
        String key = txtkhoa.getText();
        fis2 = new FileInputStream("D:\\EnDes.txt");
        FileOutputStream fos2 = new FileOutputStream("D:\\DeDes.txt");
        decrypt(key, fis2, fos2);
        BufferedReader br = null;
        String fileName = "D:\\DeDes.txt"; //GEN-
        br = new BufferedReader(new FileReader(fileName));
        StringBuffer sb = new StringBuffer();
        JOptionPane.showMessageDialog(null, "Đã Giải Mã");
        char[] ca = new char[5];
        while (br.ready()) {
            int len = br.read(ca);
            sb.append(ca, 0, len);
        }
        br.close();
        //xuat chuoi
        System.out.println("Du Lieu la : " + " " + sb);
        String chuoi = sb.toString();
        txtmahoa.setText(chuoi);
    } catch (Throwable ex) {
    }
}

```

Viết chức năng hiển thị

```

private void bnthienthitatcaActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        BufferedReader br = null;
        String fileName = "D:\\DeDes.txt"; //GEN-
        br = new BufferedReader(new FileReader(fileName));
        StringBuffer sb = new StringBuffer();
        JOptionPane.showMessageDialog(null, " Đã mở file");
        char[] ca = new char[5];
        while (br.ready()) {
            int len = br.read(ca);
            sb.append(ca, 0, len);
        }
        br.close();
        String ff = "D:\\EnDes.txt";
        br = new BufferedReader(new FileReader(ff));
        StringBuffer sb1 = new StringBuffer();
        char[] ca1 = new char[5];
        while (br.ready()) {
            int len = br.read(ca1);
            sb1.append(ca1, 0, len);
        }
        //xuat chuoil
        System.out.println("Du Lieu la : " + " " + sb);
        String chuoil = sb.toString();
        String chuoil1 = sb1.toString();
        txtvanban.setText(chuoil);
        txtmahoa.setText(chuoil1);
    } catch (IOException ex) {
    }
}

```

4.2.5 Kết Quả

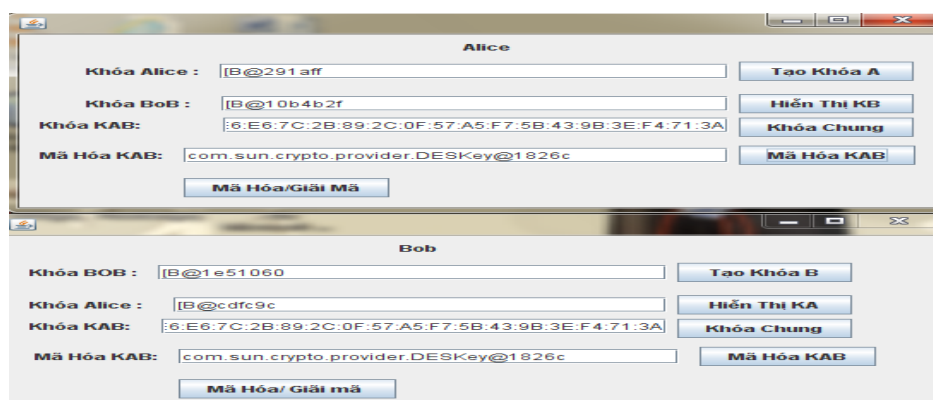
B1: Run Frame Alice: check button tạo khóa A.

B2 : Run Frame Bob: check button tạo khóa B.

B3: Frame Alice: Check button hiển thị KB; Frame Bob: Check button hiển thị KA.

B4: Frame Alice và Frame Bob lần lượt check button tạo khóa Chung.

B5: Frame Alice và Frame Bob lần lượt check button mã hóa KAB



B5. Frame Alice: check button mã hóa và giải mã , button cho phép hiển thị form mã hóa và giải mã văn bản với thuật toán DES với khóa chung của Alice và Bob.

B6: Nhập nội dung văn bản cần mã hóa và ghi xuống File, trước khi nhập cần check button Mở Khóa A (khóa Alice) để dùng khóa Alice mã hóa văn bản.

B7: Frame Bob check button mã hóa và giải mã , button cho phép hiển thị form mã hóa và giải mã văn bản với thuật toán DES với khóa chung của Alice và Bob. Check button Mở Khóa B , dung khóa của Bob để giải mã văn bản do Alice mã hóa.

The image shows two windows from a cryptographic application. The top window, titled "Bob", is for key management. It contains four text input fields: "Khóa BOB:" with value "B@1e9cb75", "Khóa Alice:" with value "B@4413ee", "Khóa KAB:" with a long hexadecimal string, and "Mã Hóa KAB:" with value "com.sun.crypto.provider.DESKey@18722". To the right of each field is a button: "Tạo Khóa B", "Hiện Thi KA", "Khóa Chung", and "Mã Hóa KAB". At the bottom is a "Mã Hóa/ Giải mã" button. The bottom window is for encryption/decryption. It has an "Input Key:" field with the same value as the top window. Below it are buttons: "Mã Hóa", "Mở Khóa A", "Mở Khóa B", and "Ghi File". There are two text areas: "Plaintext:" containing "lop bao mat thong tin toi thu 6" and "CipherText:" containing a string of non-printable characters. At the bottom are "Giải Mã" and "All Show" buttons. The taskbar at the bottom shows "TCDaiViet" and the time "17/09/2011 7:32 AM".

Bob

Khóa BOB : B@1e9cb75 **Tạo Khóa B**

Khóa Alice : B@4413ee **Hiện Thi KA**

Khóa KAB: A:93:3C:5D:91:53:23:8E:CD:73:29:9A:32:AD:C7:5F:05 **Khóa Chung**

Mã Hóa KAB: com.sun.crypto.provider.DESKey@18722 **Mã Hóa KAB**

Mã Hóa/ Giải mã

Input Key : com.sun.crypto.provider.DESKey@18722

Mã Hóa **Mở Khóa A** **Mở Khóa B** **Ghi File**

Plaintext :

lop bao mat thong tin toi thu 6

CipherText :

m>x N'sÓŒÉ" % h:‡BY Ç@Y*#> „„

Giải Mã **All Show**

TCDaiViet 17/09/2011 7:32 AM

BÀI 5. HÀM BẮM

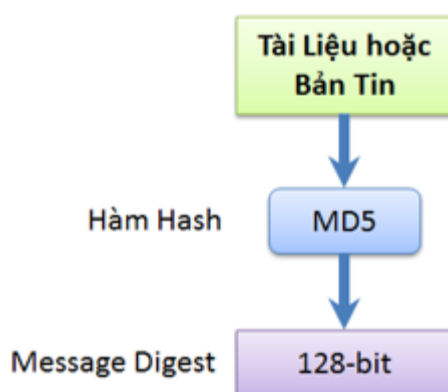
5.1 HÀM BẮM MD5

Hiện thực thuật toán hàm băm MD5 với yêu cầu sau:

- 1.1 Cho phép người dùng nhập username và password.
- 1.2 Dùng thuật toán MD5 băm username , password và lưu vào File
- 1.3 Dùng username và password đăng nhập , chứng thực với File đã ghi username, password.

5.1.1 Giới thiệu thuật toán MD5

MD5 (Message - Digest - algorithm 5) giải thuật tiêu hóa tập tin là một chuẩn Internet (RFC 1321). Có khả năng băm mã hóa tập tin bất kỳ thành chuỗi HEX 32 ký tự, tương đương 128-bit (*mỗi ký tự hex 4-bit x 32 ký tự = 128 bit*).



Hoặc có thể định nghĩa theo cách khác. MD5 là cách căn bản để lấy chùm ký tự (là digest, alphabetic hay gì khác), được gọi là string nhập vào và cho ra là 32 ký tự hexa. (0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f).

MD5 được thiết kế bởi Ronald Rivest vào năm 1991 để thay thế cho hàm băm trước đó, MD4. Vào năm 1996, người ta phát hiện ra một lỗ hổng trong MD5; trong khi vẫn chưa biết nó có phải là lỗi nghiêm trọng hay không, những chuyên gia mã hóa bắt đầu đề nghị sử dụng những giải thuật khác, như SHA-1 (khi đó cũng bị xem là không an

toàn). Trong năm 2004, nhiều lỗ hổng hơn bị khám phá khiến cho việc sử dụng giải thuật này cho mục đích bảo mật đang bị đặt nghi vấn.

ĐẶC ĐIỂM MD5

Việc tính MD đơn giản, có khả năng xác định được file có kích thước nhiều Gb.

Không có khả năng tính ngược, khi tìm ra MD.

Do bản chất ngẫu nhiên của hàm băm và số lượng cực lớn các giá trị hash có thể, nên hầu như không có khả năng hai bản tin phân biệt có cùng giá trị hash.

Giá trị MD phụ thuộc vào bản tin tương ứng.

Một chuỗi chỉ có duy nhất một hash.

Giá trị MD phụ thuộc vào tất cả các bit của bản tin tương ứng.

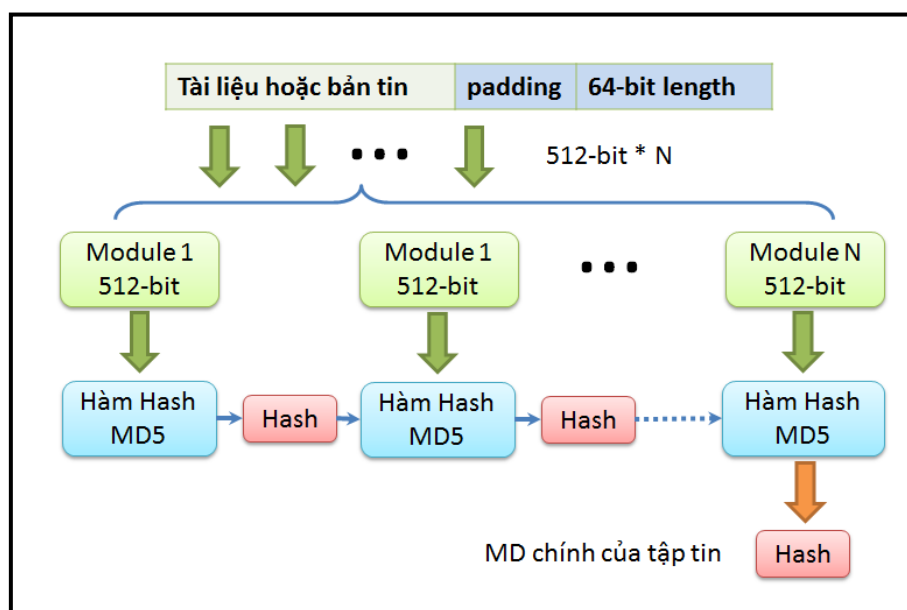
Ví dụ :

love is blue → 03d4ad6e7fee3f54eb46b5ccde58249c

love is Blue → 82b76f8eeb4a91aa640f9a23016c7b1c

5.1.2 Thuật toán MD5

Giải thuật MD5 chính hoạt động trên trạng thái 128-bit, được chia thành 4 từ 32-bit, với ký hiệu A, B, C và D. Chúng được khởi tạo với những hằng số cố định. Giải thuật chính sau đó sẽ xử lý các khối tin 512-bit, mỗi khối xác định một trạng thái. Quá trình xử lý khối tin bao gồm bốn giai đoạn giống nhau, gọi là vòng; mỗi vòng gồm có 16 tác vụ giống nhau dựa trên hàm phi tuyến F, cộng Module, và dịch trái.



Thực hiện qua các bước sau:

Bước 1: Thêm các bit vào chuỗi

Thực hiện nối dài thông điệp. (theo hình vẽ thông điệp là B) để chia nhỏ thành các module 512.



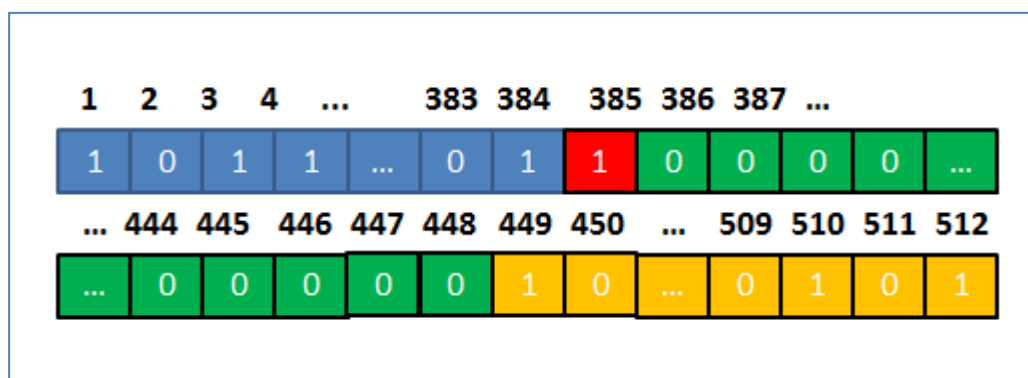
Thêm bit '1' vào cuối thông điệp để đánh dấu.

Thêm vào k bit '0' sao cho $(b \text{ bit} + \text{bit } 1 + k \text{ bit } 0) \bmod 512 = 448$

64 bit tiếp theo sẽ được thêm vào biểu thị chiều dài của chuỗi bit ban đầu.

$$(B \text{ bit} + \text{bit '1'} + k \text{ bit '0'} + 64 \text{ bit chiều dài}) \bmod 512 = 0$$

Ví dụ: Ta có chuỗi 384bit



Quá trình thêm bit

Bước 2: Khởi tạo bộ đệm MD

Một bộ đệm 4 word (A,B,C,D) được dùng để tính mã số thông điệp. Ở đây mỗi A,B,C,D là một thanh ghi 32 bit. Những thanh ghi này được khởi tạo theo những giá trị hex sau (các byte thấp trước) :

<i>word</i>	<i>A</i>	:	<i>01</i>	<i>23</i>	<i>45</i>	<i>67</i>
<i>word</i>	<i>B</i>	:	<i>89</i>	<i>ab</i>	<i>cd</i>	<i>ef</i>
<i>word</i>	<i>C</i>	:	<i>fe</i>	<i>dc</i>	<i>ba</i>	<i>98</i>
<i>word D : 76 54 32 10</i>						

Bước 3: Xử lý thông điệp theo từng khối 16 word

Trước hết ta định nghĩa các hàm phụ, các hàm này nhận đầu vào là 3 word 32 bit và tạo ra một word 32 bit.

$$\begin{aligned}
 F_1(X, Y, Z) &= (X \wedge Y) \vee (\neg X \wedge Z) \\
 F_2(X, Y, Z) &= (X \wedge Z) \vee (Y \wedge \neg Z) \\
 F_3(X, Y, Z) &= X \oplus Y \oplus Z \\
 F_4(X, Y, Z) &= Y \oplus (X \vee \neg Z)
 \end{aligned}$$

Với $\oplus, \wedge, \vee, \neg$ lần lượt là **XOR, AND, OR, NOT**

Đây là quá trình thực hiện xử lý của 4 hàm F ở trên:

Quá trình này sử dụng một bảng có 64 giá trị $T[1 \dots 64]$ được tạo ra từ hàm sin. Gọi $T[i]$ là phần tử thứ i của bảng, thì $T[i]$ là phần nguyên của $4294967296 * |\sin(i)|$, i được tính theo radian.

Thực hiện:

```
/* Xử lý mỗi khối 16 word */
```

```
For (i = 0 to N/16-1) do
```

```
/* Copy block i into X. */
```

```
For j = 0 to 15 do
```

```
Set  $X[j]$  to  $M[i*16+j]$ .
```

```
end /* of loop on j */
```

```
/* Lưu A vào AA, B vào BB, C vào CC, D vào DD . Làm buffer */
```

```
AA = A
```

```
BB = B
```

```
CC = C
```

```
DD = D
```

Quá trình thực hiện qua các vòng

Vòng 1

- **[abcd k s t]** là bước thực hiện của phép toán

$$a = b + ((a + F(b, c, d) + X[k] + T[i]) \lll s)$$

```

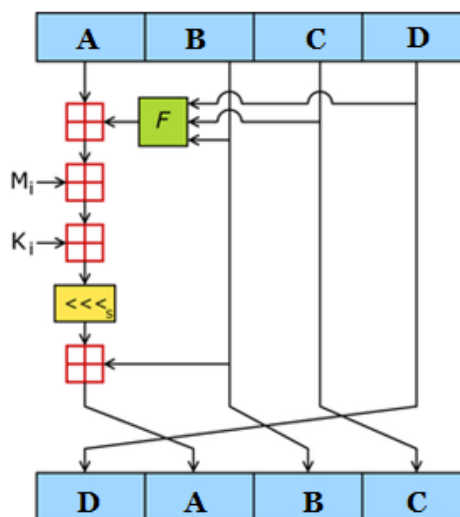
/* Do the following 16 operations. */
[ABCD 0 7 1] [DABC 1 12 2] [CDAB 2 17 3] [BCDA 3 22 4]
[ABCD 4 7 5] [DABC 5 12 6] [CDAB 6 17 7] [BCDA 7 22 8]
[ABCD 8 7 9] [DABC 9 12 10] [CDAB 10 17 11] [BCDA 11 22 12]
[ABCD 12 7 13] [DABC 13 12 14] [CDAB 14 17 15] [BCDA 15 22 16]

```

- Nhận xét: Vòng Một dùng hàm F_1 với giá trị
 - t từ 1..16 và
 - k từ 0..15

Một Thao Tác MD5 - MD5

- Một thao tác MD5—MD5 bao gồm 64 tác vụ thể này, nhóm trong 4 vòng 16 tác vụ. F là một hàm phi tuyến; một hàm được dùng trong mỗi vòng. M_i chỉ ra một khối tin nhập vào 32-bit, và K_i chỉ một hằng số 32-bit, khác nhau cho mỗi tác vụ.
- $\lll s$ chỉ sự xoay bit về bên trái s đơn vị; s thay đổi tùy theo từng tác vụ. \boxplus chỉ cộng thêm với modulo 2^{32} .



Vòng 2

- [abcd k s t] là bước thực hiện của phép toán

$$a = b + ((a + F_2(b, c, d) + X[k] + T[i]) \lll s)$$

```
/* Do the following 16 operations. */
[ABCD 1 5 17] [DABC 6 9 18] [CDAB 11 14 19] [BCDA 0 20 20]
[ABCD 5 5 21] [DABC 10 9 22] [CDAB 15 14 23] [BCDA 4 20 24]
[ABCD 9 5 25] [DABC 14 9 26] [CDAB 3 14 27] [BCDA 8 20 28]
[ABCD 13 5 29] [DABC 2 9 30] [CDAB 7 14 31] [BCDA 12 20 32]
```

- Nhận xét: Vòng Một dùng hàm F_2 với giá trị
 - t từ 17..32 và
 - $k = (1+5t) \bmod 16$

Vòng 3

- [abcd k s t] là bước thực hiện của phép toán

$$a = b + ((a + F_3(b, c, d) + X[k] + T[i]) \lll s)$$

```
/* Do the following 16 operations. */
[ABCD 5 4 33] [DABC 8 11 34] [CDAB 11 16 35] [BCDA 14 23 36]
[ABCD 1 4 37] [DABC 4 11 38] [CDAB 7 16 39] [BCDA 10 23 40]
[ABCD 13 4 41] [DABC 0 11 42] [CDAB 3 16 43] [BCDA 6 23 44]
[ABCD 9 4 45] [DABC 12 11 46] [CDAB 15 16 47] [BCDA 2 23 48]
```

- Nhận xét: Vòng Một dùng hàm F_2 với giá trị
 - t từ 33..48 và
 - $k = (5+3t) \bmod 16$

Vòng 4

- $[abcd\ k\ s\ t]$ là bước thực hiện của phép toán

$$a = b + ((a + F_4(b, c, d) + X[k] + T[i]) \lll s)$$

```
/* Do the following 16 operations. */
[ABCD 0 6 49] [DABC 7 10 50] [CDAB 14 15 51] [BCDA 5 21 52]
[ABCD 12 6 53] [DABC 3 10 54] [CDAB 10 15 55] [BCDA 1 21 56]
[ABCD 8 6 57] [DABC 15 10 58] [CDAB 6 15 59] [BCDA 13 21 60]
[ABCD 4 6 61] [DABC 11 10 62] [CDAB 2 15 63] [BCDA 9 21 64]
```

- Nhận xét: Vòng Một dùng hàm F_2 với giá trị
 - t từ 49..64 và
 - $k = 7t \bmod 16$

/* Sau đó làm các phép cộng sau. (Nghĩa là cộng vào mỗi thanh ghi giá trị của nó trước khi vào vòng lặp) */

A	=	A	+	AA
B	=	B	+	BB
C	=	C	+	CC
D	=	D	+	DD

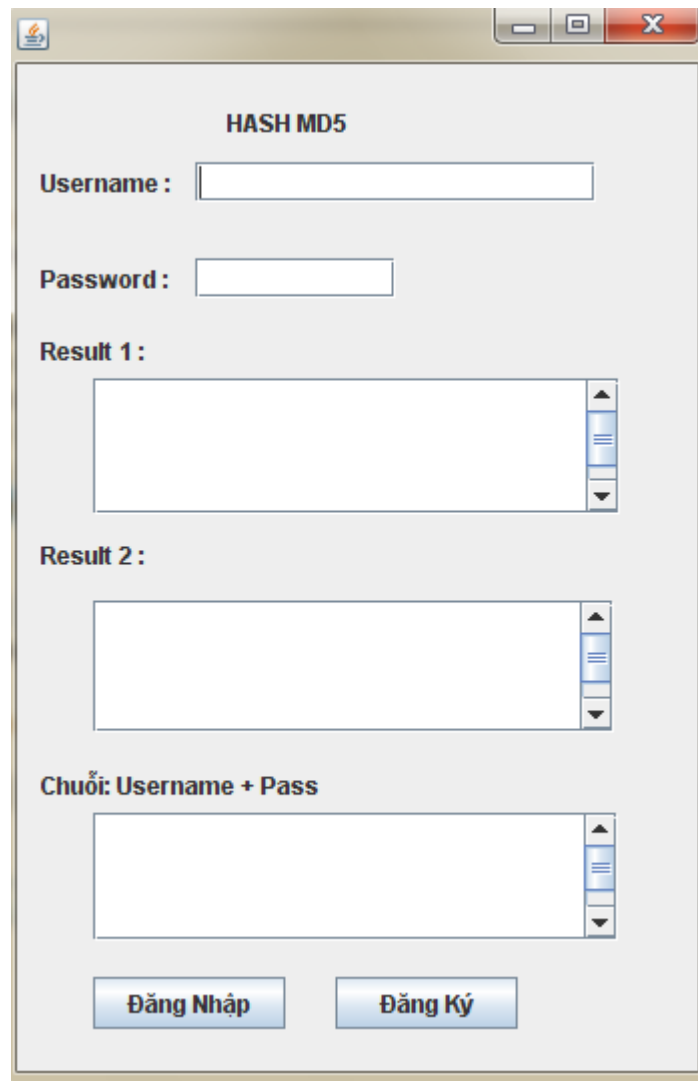
end /* of loop on i */

Bước 4: In ra

Mã số thông điệp được tạo ra là A,B,C,D. Nghĩa là chúng ta bắt đầu từ byte thấp của A, kết thúc với byte cao của D.

5.2 HƯỚNG DẪN THỰC HÀNH

Thiết kế form



HASH MD5

Username :

Password :

Result 1 :

Result 2 :

Chuỗi: Username + Pass

Thư Viện cần sử dụng

```
import java.io.BufferedReader;  
import java.io.BufferedWriter;  
import java.io.FileReader;  
import java.io.FileWriter;  
import java.security.MessageDigest;  
import javax.swing.JOptionPane;
```

Viết hàm xử lý sự kiện

Xử lý sự kiện “Đăng Ký”

```

private void bntDangKyActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        // TODO add your handling code here:
        String user = txtuser.getText();
        String pass = txtpass.getText();
        String bam = "";
        bam = user + pass;
        MessageDigest md = MessageDigest.getInstance("MD5");
        md.update(bam.getBytes());
        byte[] byteData = md.digest();
        //convert the byte to hex format method 1
        StringBuffer sb = new StringBuffer();
        for (int i = 0; i < byteData.length; i++) {
            sb.append(Integer.toString((byteData[i] & 0xff) + 0x100, 16).substring(1));
        }
        System.out.println("Digest(in hex format):: " + sb.toString());
        txtbam1.setText(sb.toString());
        //convert the byte to hex format method 2
        StringBuffer hexString = new StringBuffer();
        for (int i = 0; i < byteData.length; i++) {
            String hex = Integer.toHexString(0xff & byteData[i]);
            if (hex.length() == 1) {
                hexString.append('0');
            }
            hexString.append(hex);
        }
        System.out.println("Digest(in hex format):: " + hexString.toString());
        txtbam2.setText(hexString.toString());
        txtgoc.setText(bam.toString());

        //Viết chức năng ghi File
        BufferedWriter bw = null;
        //ghi van ban da ma hoa
        String fileName = "D:\\BamMD5.txt";
        //luu van ban
        bw = new BufferedWriter(new FileWriter(fileName));
        // ghi van ban
        bw.write(hexString.toString());
        bw.close();
        JOptionPane.showMessageDialog(null, "Bạn Đã Đăng Ký Thành Công .Vui lòng Đăng nhập lại !!!");

    } catch (Exception ex) {
        System.out.println(" Loi bam username và password : " + ex);
    }
}

```

Xử lý sự kiện đăng nhập

```

private void bntDangNhapActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String user = txtuser.getText();
    String pass = txtpass.getText();
    String bam = "";
    bam = user + pass;
    //mở File đã lưu username và password
    BufferedReader br = null;
    String fileName = "D:\\BamMD5.txt"; //GEN-
    try{
        br = new BufferedReader( new FileReader(fileName));
        StringBuffer sb = new StringBuffer();
        char[] ca = new char[5];
        while (br.ready()) {
            int len = br.read(ca);
            sb.append(ca, 0, len);
        }
        br.close();
        //hiển thị File đã lưu
        System.out.println("chung thuc :" + " " + sb);
        String chuoi = sb.toString();

        // Thực hiện băm username và pass cho người dùng đăng nhập
        MessageDigest md = MessageDigest.getInstance("MD5");
        md.update(bam.getBytes());
        byte[] byteData = md.digest();
        StringBuffer hexString = new StringBuffer();
        for (int i = 0; i < byteData.length; i++) {
            String hex = Integer.toHexString(0xff & byteData[i]);
            if (hex.length() == 1) {
                hexString.append('0');
            }
            hexString.append(hex);
        }
        System.out.println("Bam username và password :" + " " + hexString.toString());
        // Thực hiện so sánh username và password
        Boolean k=hexString.toString().equals(chuoi);
        if(k==true)
        {
            JOptionPane.showMessageDialog(null, " Đăng Nhập Thành Công");
            // hiển thị username và password bị băm
            txtbam1.setText(hexString.toString());
        }
    } catch (IOException ex) {
        ex.printStackTrace();
    }
}

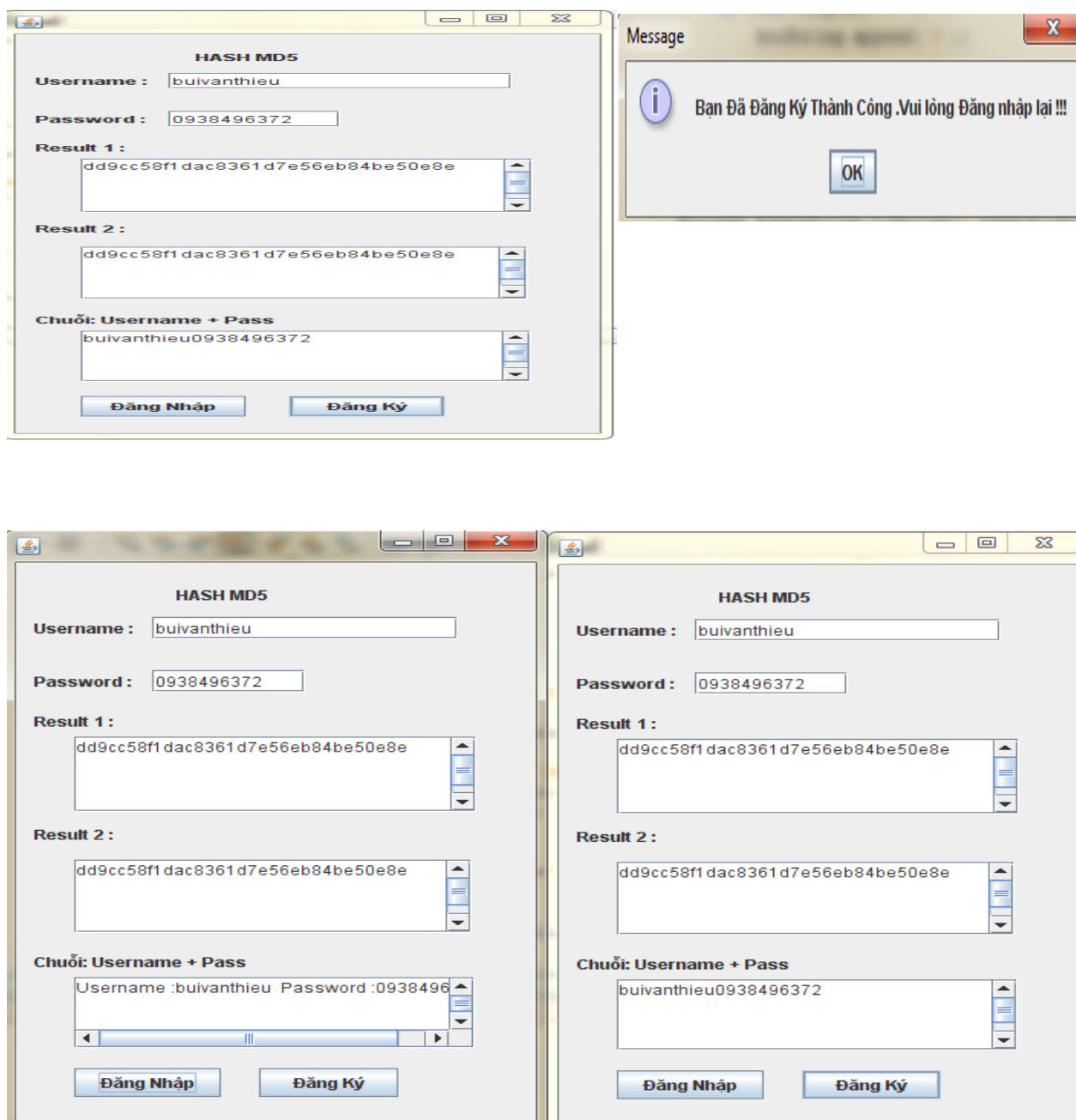
```

```

//hiển thị mã băm lưu ở File
txtbam2.setText(chuoi);
txtgoc.setText("Username :"+user+" " + " Password :"+ pass );
}
else
    JOptionPane.showMessageDialog(null, " Đăng Nhập Thất bại");
} catch (Exception ex) {
    System.out.println(" Lỗi Đăng Nhập :"+ ex);
}
}

```

Kiểm Tra Kết Quả



5.3 THUẬT TOÁN SHA

Hiện thực thuật toán hàm băm SHA với yêu cầu sau:

Nhập chuỗi và sử dụng thuật toán SHA băm chuỗi theo 2 cách.

Hướng dẫn Thuật Toán

Thiết kế Frame:

The image shows a Java Swing window titled "Chương Trình Hiện Thực Hàm Băm SHA". The window has a light gray background and a thin orange border. It contains the following elements:

- A text input field labeled "Nhập Chuỗi" (Enter String) with a bright green background.
- Two text output fields labeled "Hash SHA C1" and "Hash SHA C2", each with a white background and a vertical scrollbar on the right.
- Two buttons at the bottom: "BămSHA" (Hash SHA) and "Thoát" (Exit).

```

private void bntbamSHAActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        String chuoi = "";
        chuoi = txtchuoi.getText();
        MessageDigest md = MessageDigest.getInstance("SHA-256");
        md.update(chuoi.getBytes());
        byte byteData[] = md.digest();
        StringBuffer sb = new StringBuffer();
        for (int i = 0; i < byteData.length; i++) {
            sb.append(Integer.toString((byteData[i] & 0xff) + 0x100, 16).substring(1));
        }

        System.out.println("Hex format1 : " + sb.toString());
        txtsha1.setText(sb.toString());

        //convert the byte to hex format method 2
        StringBuffer hexString = new StringBuffer();
        for (int i=0;i<byteData.length;i++) {
            String hex=Integer.toHexString(0xff & byteData[i]);
            if(hex.length()==1) hexString.append('0');
            hexString.append(hex);
        }
        System.out.println("Hex format2 : " + hexString.toString());
        txtsha2.setText(hexString.toString());
    } catch (NoSuchAlgorithmException ex) {
        Logger.getLogger(FrameSHA.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

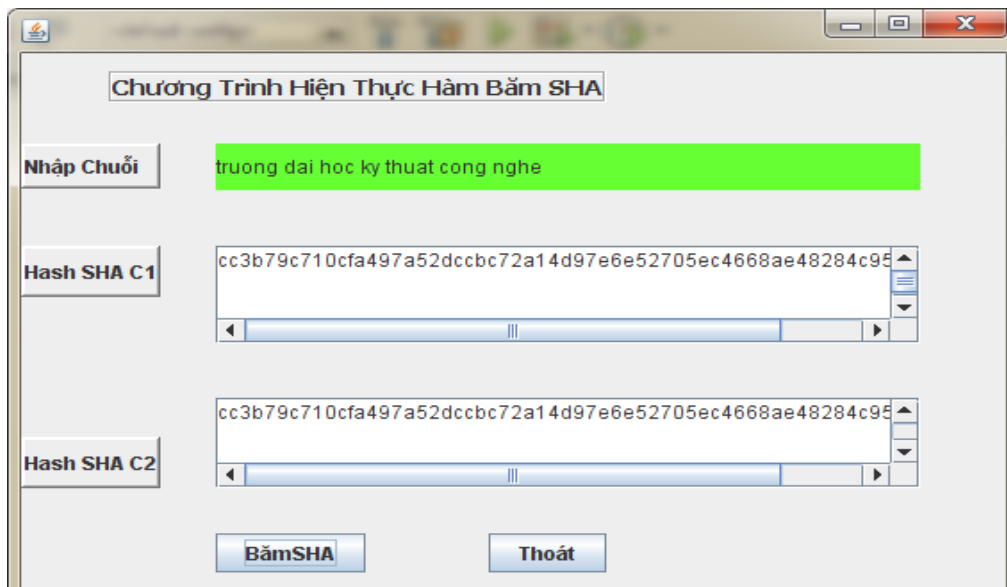
Xử lý sự kiện thoát Form

```

private void bntthoatActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    System.exit(WIDTH);
}

```

Kết Quả



Bài tập mở rộng: Hiện thực thuật toán hàm băm SHA với yêu cầu sau:

- 3.1 Cho phép người dùng nhập username và password.
- 3.2 Dùng thuật toán SHA băm username , password và lưu vào File
- 3.3 Dùng username và password đăng nhập , chứng thực với File đã ghi username, password. (sử dụng hướng dẫn bài 1 và bài 2...)