Instituto Tecnológico de Durango

Ing. En Sistemas Computacionales
Criptografía
Encriptado: "DES"

N° de Control: 12041156
Nombre: Mauricio Alejandro Martínez Pacheco
GRUPO: 7YZ

Victoria de Durango, Dgo.

23 DE FEBRERO DE 2015

Introducción

En la clase de criptografía se nos encargó implementar el algoritmo de cifrado de DES en el lenguaje de programación Java.

En sí es un algoritmo sencillo, pero muy laborioso de implementar, por lo que realizarlo, solo implicó trabajo mecánico y no tanto intelectual, ya que el algoritmo ya estaba descrito por Stallings.

Desarrollo

Comenzando a desarrollar el algoritmo, me topo que solamente se encriptan 64 bits de texto plano y 64 de la llave, no más, ni menos. Por lo que primeramente me puse a desarrollar una manera de permitir entradas menores a 64 y mayores, esto en el texto plano. En la llave solo permito menos de 64 bits, ya que en el texto plano, más de 64 bits solo significa ejecutar el DES más veces, pero en la llave no es posible.

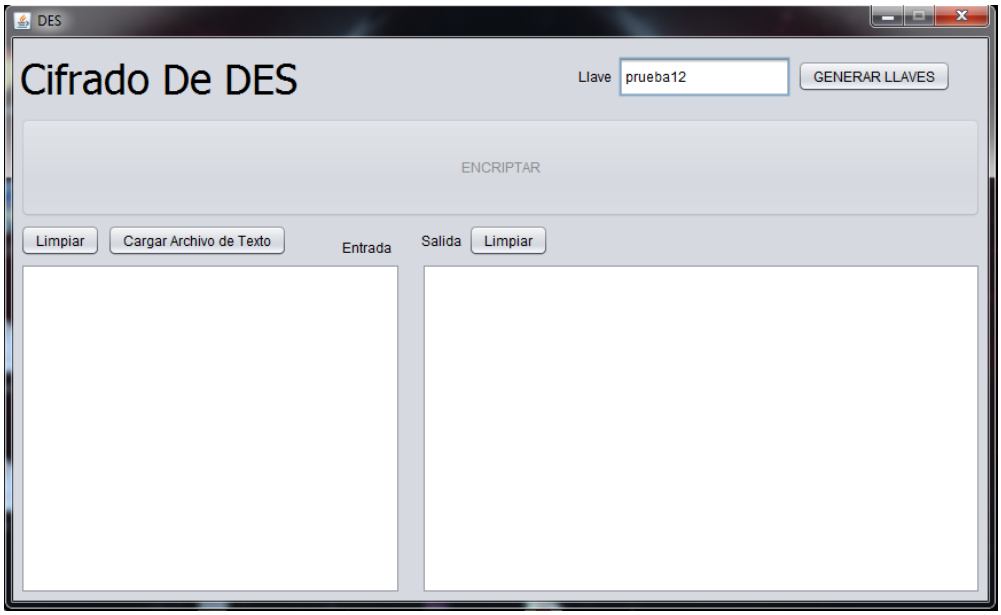
Para esto utilizo un objeto llamado Vector, éste se comporta como un arreglo dinámico. En este objeto guardo arreglos de tipo booleano, los cuales son los paquetes de 64 bits que pasaran por DES.

```
Vector entradabits=new Vector();
```

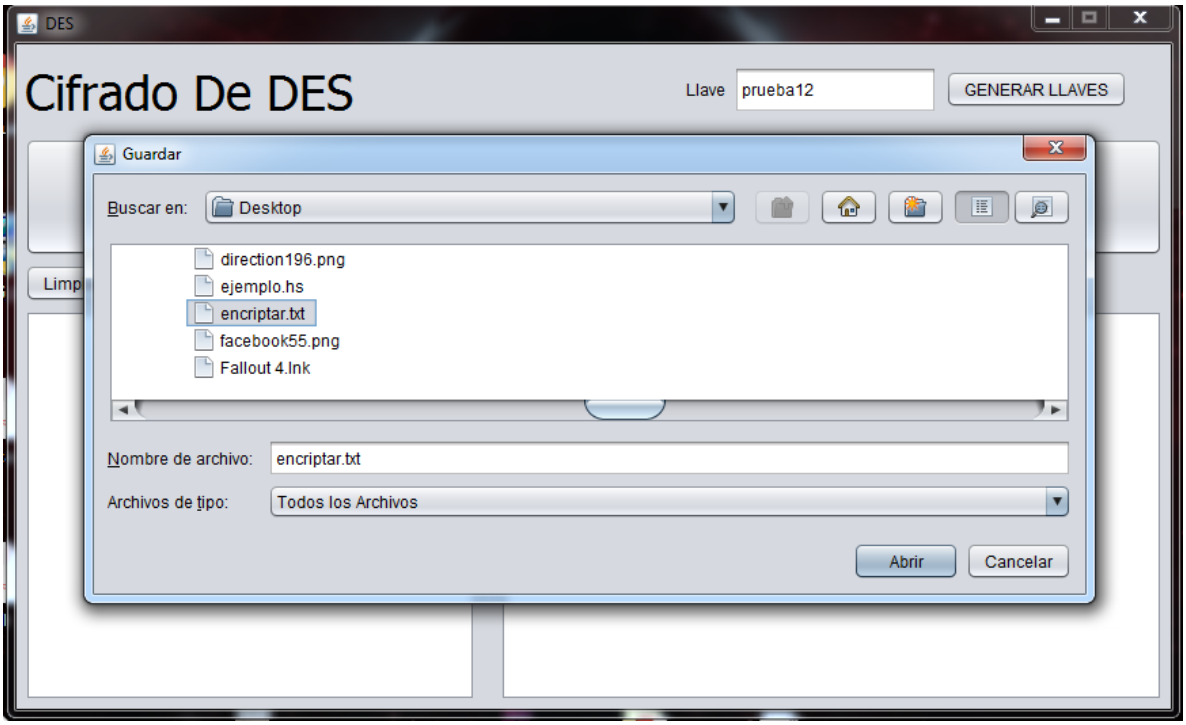
Este objeto tiene un método llamado .size() que nos retorna el número de elementos que contiene, esto nos va a servir para darle el límite al ciclo que ejecutará n veces el algoritmo de DES además de ir concatenando el encriptado en el jTextArea de salida.

La generación de llaves la manejo en un método aparte, esto con el objetivo de hacer una sola generación de las 16 llaves, para ahorrar tiempo de ejecución y por si deseamos encriptar varios textos planos con la misma llave.

Una vez que se generan las llaves, el botón de ”ENCRIPTAR” se habilita y ya podemos realizar el proceso.



La entrada la tengo en una caja de texto, la cual permite la escritura directa de caracteres o cargar un archivo de texto .txt.



Al darle al botón de encriptar se toma el contenido del jTextArea de entrada, lo convierte a un String y después mediante un ciclo se va convirtiendo carácter por carácter a un String que contiene los 8 bits en binario por ejemplo: el carácter “p” es el 112 ASCII, esto equivale a “01110000”.

Ahora se toma esa cadena y se evalúa dentro de un ciclo si su longitud es menor a 64 o mayor e igual. Si es menor de 64 se calculan cuantos ceros le falta al paquete para ser de 64 bits y se agregan a la izquierda de la cadena. Si es mayor o igual a 64 se va rellenando un arreglo booleano de 64 posiciones con true si el carácter de la cadena es '1' y false si no es, después se inserta ese arreglo booleano en el objeto Vector y se le extrae a la cadena principal esos 64 bits y si queda una cadena nula se termina el ciclo, pero si aún quedan bits se repite el proceso de evaluar su longitud.

Ya teniendo el objeto Vector con los paquetes de 64 bits, se procede a llamar al método encriptar() de la clase DES. Que el algoritmo esté en otra clase nos facilita la llamada de n veces al algoritmo.

En esta clase llamada DES, el constructor va a recibir como parámetro la llave, esto para la generación única de las 16 llaves, en este constructor se inicializan todas las matrices del algoritmo y al final se manda llamar al método obtencionLlaves() en el cual se realiza el algoritmo de generación de llaves.

La generación de llaves comienza con la permutación de opción uno, la cual nos reducirá la llave de 64 a una de 56 bits, además esta misma matriz nos parte los 56 en dos mitades de 28, que decidí almacenarlas en dos arreglos booleanos de 28 bits.

Después en un ciclo de 16 vueltas se realizan los corrimientos a la izquierda de bits dependiendo del Schedule, el Schedule lo metí dentro de otro arreglo, así se sabe cuál se va a usar en la vuelta n. Después de correr los bits, se respaldan las dos mitades para la siguiente llave y se realiza la permutación de opción 2. Decidí utilizar una matriz de 16 renglones por 48 columnas que está declarada como un atributo de la clase, esto quiere decir que puede ser utilizada por todos los métodos de esa clase, así es como se puede generar las llaves una sola vez y encriptar tantos textos planos queramos con esa llave. Entonces ya realizada la permutación se almacenan las dos mitades en un renglón de la matriz de 16 llaves y así se repite el proceso.

Dentro de la clase esta el método encriptar(), el cual es en donde está todo el algoritmo de las redes de feistel, etc. Este método recibe como parámetro el arreglo booleano que contiene el paquete de 64 bits que se va a encriptar. Primeramente en el método se realiza la permutación inicial, aquí es donde implemente estos for anidados para realizarlo más eficientemente, me parece muy limpia esta implementación, porque otros compañeros tenían muchas más líneas de código.

```
int cont=0;
      for(int a=0;a<8;a++){
         for(int b=0;b<8;b++){
            permutado[cont]=paquete[IP[a][b]-1];
            System.out.print(permutado[cont]+",");
            cont++;
         }
      }
```

Después hago la primera separación de las mitades, en donde cada una será de 32 bits. Aquí empiezan las 16 rondas, que simplemente están en un for de 16 vueltas. Al principio de la ronda, hago un backup de la mitad derecha en una variable auxiliar, para que al final de la ronda pase a ser la mitad izquierda, esto porque si se asigna la mitad derecha a la izquierda desde el inicio se pierde la mitad izquierda para la operación XOR.

A continuación es la permutación de expansión, que esta igual en un for como la inicial, solo que claro, se manda llamar la matriz de expansión.

En seguida viene el XOR con la llave de la ronda

```
for (int a = 0; a < 48; a++) {
        xor[a] = expansion[a]  != llaves[i][a];
}
```

La tabla de verdad del XOR es que si son iguales es false y si son diferentes es true, entonces así se puede expresar sin la necesidad de if, ya que la expresión retorna booleano. El índice i es el de la ronda actual.

Después siguen las S-Boxes. Aquí declare las S-Boxes como un arreglo tridimensional, en donde se puede ver como un vector de matrices, esto para que el proceso de S-Box sea solo un ciclo de 8 vueltas.

Para preparar la entrada de 8 paquetes de 6 bits, dividí el arreglo principal en una matriz de 8 renglones por 6 columnas.

Ahora tomo la posición 0 y la 5 que son el primero y el ultimo, y representan la fila de la sbox, también los de las posiciones 1 a la 4 que representan las columnas. Con un método los convierto a decimal. Así ya tengo la S-Box que se va a usar (con el contador del ciclo de 8 vueltas), la fila y la columna.

```
Sboxes[a][binarioADecimal(dosbits)][binarioADecimal(cuatrobits)]
```

Esto regresará un numero entero del 0 al 15, hice un método que convierte este entero a un arreglo de booleanos de 4 posiciones en binario.

```
temp[]=Binario(Sboxes[a][binarioADecimal(dosbits)][binarioADecimal(cuatrobits)]);
 for(int b=0;b<4;b++){
        sboxeado[cont]=temp[b];
         cont++;
}
```

Después sigue la permutación p que es como las otras permutaciones en un ciclo for.

Luego el resultado se le aplica un XOR con la mitad izquierda y esto queda en la mitad derecha, el backup que hicimos desde el inicio de la ronda se asigna como la nueva izquierda y comienza la ronda de nuevo.

Al finalizar las 16 rondas, se hace el intercambio de mitades, la izquierda pasa a ser la derecha y la derecha la izquierda. Y por ultimo, se realiza la permutación inicial inversa.

El resultado lo convierto a un String de 64 caracteres “booleanos”. Y este String es el valor que regresa el método, esto para pasarlo a la interfaz gráfica en donde hay un método para convertir ese String a ASCII.

Este método, separa el String en 8 substrings de 8 caracteres, ya que un ASCII es de 8 bits. Después se realiza la conversión de esos 8 bits a entero, y ese entero ya es el carácter, ya que java expresa los caracteres como valores enteros. Y voy concatenando dichos caracteres en un String y ya terminado se muestra en el jTextField de salida.

Conclusión

El algoritmo es muy sencillo, pero muy laborioso. Además muy complicado de estar comprobando que los valores que van dando son los correctos. Para la comprobación utilicé el ejemplo del libro convirtiendo los valores hexadecimales del libro a binario

Plaintext:	02468aceeca86420	0000111100010101011100011100100101000111110110011110100001011001
Key:	0f1571c947d9e859	0000001001000110100010101100111011101100101010000110010000100000
Ciphertext:	da02ce3a89ecac3b	1101101000000010110011100011101010001001111011001010110000111011

```
import java.awt.HeadlessException;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.Vector;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author NTHINGs
 */
public class GUI extends javax.swing.JFrame {

    DES des;
    public GUI() {
        initComponents();
        int limite = 8;
        KeyListener keyListener = new KeyListener() {
        public void keyPressed(KeyEvent keyEvent) {
        }

        public void keyReleased(KeyEvent keyEvent) {
        }

        public void keyTyped(KeyEvent keyEvent) {
            if (llave.getText().length() == limite) {
                keyEvent.consume();
              }

        }
    };

        llave.addKeyListener(keyListener);
    }

    private String Binario(int Decimal) {
        String resultado="";
        int temp=Decimal;
        //Se aplica el algoritmo para convertir a binario
        while(temp != 0){
            if(temp %2 == 0){
```

```
            resultado="0"+resultado;
        }else{
            resultado="1"+resultado;
        }
        temp = temp/2;
    }

    //Se le agregan ceros a la izquierda, para completar 8 bits por caracter
    int cerosalaizquierda=8-resultado.length();
    for(int x=0;x<cerosalaizquierda;x++){
        resultado="0"+resultado;
    }
    return resultado;
}

private String ASCII(String binario){

    String caracter="";
    String caracteres[]=new String[8];
    String ascii="";
    int valor=0;


    //Separamos la cadena binario en 8 subcadenas de 8 bits
    int cont=8;
    for(int a=0;a<8;a++){
        caracteres[a]=binario.substring(cont-8, cont);
        System.out.println("Caracter "+(a+1)+" :"+caracteres[a]);
        cont=cont+8;
    }


    //Conversion del string binario a numero entero
    for(int a=0;a<8;a++){
        int longitud=caracteres[a].length();
        int potencia = longitud - 1;
        valor=0;
        for(int b=0;b<longitud;b++){
            if(caracteres[a].charAt(b)=='1'){
             valor+= Math.pow(2, potencia);
            }
            potencia --;
        }

        System.out.println("Caracter "+(a+1)+" :"+valor+" ASCII: "+(char)valor);

        ascii=ascii+(char)valor;
    }

    return ascii;
}
```

```
/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jScrollPane1 = new javax.swing.JScrollPane();
    entrada = new javax.swing.JTextArea();
    jScrollPane2 = new javax.swing.JScrollPane();
    salida = new javax.swing.JTextArea();
    jLabel1 = new javax.swing.JLabel();
    jButton1 = new javax.swing.JButton();
    jButton3 = new javax.swing.JButton();
    encriptarbtn = new javax.swing.JButton();
    jLabel2 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    llave = new javax.swing.JTextField();
    jLabel4 = new javax.swing.JLabel();
    jButton8 = new javax.swing.JButton();
    jButton2 = new javax.swing.JButton();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setTitle("DES");
    setResizable(false);

    entrada.setColumns(20);
    entrada.setRows(5);
    jScrollPane1.setViewportView(entrada);
    entrada.setText("");

    salida.setEditable(false);
    salida.setColumns(20);
    salida.setRows(5);
    salida.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
    jScrollPane2.setViewportView(salida);
    salida.setText("");

    jLabel1.setFont(new java.awt.Font("Tahoma", 0, 36)); // NOI18N
    jLabel1.setText("Cifrado De DES");

    jButton1.setText("Limpiar");
    jButton1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton1ActionPerformed(evt);
        }
    });
```

```
        jButton3.setText("Cargar Archivo de Texto");
        jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton3ActionPerformed(evt);
            }
        });

        encriptarbtn.setText("ENCRIPTAR");
        encriptarbtn.setEnabled(false);
        encriptarbtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                encriptarbtnActionPerformed(evt);
            }
        });

        jLabel2.setText("Entrada");

        jLabel3.setText("Salida");

        llave.setColumns(8);

        jLabel4.setText("Llave");

        jButton8.setText("GENERAR LLAVES");
        jButton8.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton8ActionPerformed(evt);
            }
        });

        jButton2.setText("Limpiar");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()

    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(encriptarbtn, javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,  Short.MAX_VALUE)
                    .addGroup(layout.createSequentialGroup()
```

```
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                        .addGroup(layout.createSequentialGroup()
                            .addComponent(jButton1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                            .addComponent(jButton3)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE,  Short.MAX_VALUE)
                            .addComponent(jLabel2)
                            .addGap(26, 26, 26))
                        .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 332,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 324,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGap(18, 18, 18)))

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(jScrollPane2)
                        .addGroup(layout.createSequentialGroup()
                            .addComponent(jLabel3)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                            .addComponent(jButton2)
                            .addGap(0, 0, Short.MAX_VALUE))
                        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                            .addGap(0, 137, Short.MAX_VALUE)
                            .addComponent(jLabel4)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                            .addComponent(llave, javax.swing.GroupLayout.PREFERRED_SIZE,
150, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                            .addComponent(jButton8)
                            .addGap(26, 26, 26)))))
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
```

```
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addContainerGap()
                    .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,  Short.MAX_VALUE)
                    .addGap(4, 4, 4))
                .addGroup(layout.createSequentialGroup()
                    .addContainerGap(16, Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(llave, javax.swing.GroupLayout.PREFERRED_SIZE,
35, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(jLabel4)
                        .addComponent(jButton8))
                    .addGap(18, 18, 18)))
            .addComponent(encriptarbtn, javax.swing.GroupLayout.DEFAULT_SIZE,  87,
Short.MAX_VALUE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel2, javax.swing.GroupLayout.Alignment.TRAILING)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jButton1)
                    .addComponent(jButton3)
                    .addComponent(jLabel3)
                    .addComponent(jButton2)))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE,
288, Short.MAX_VALUE)
                .addComponent(jScrollPane2))
            .addContainerGap())
        );

        pack();
    }// </editor-fold>

    private void jButton1ActionPerformed(java.awt.event.ActionEvent  evt) {
        // TODO add your handling code here:
        entrada.setText("");
    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent  evt) {
        File archivo = null;
        FileReader fr = null;
        BufferedReader br = null;
```

```
    try {
        //Se carga un archivo de texto al jTextArea
        JFileChooser chooser = new JFileChooser();
        chooser.setDialogTitle("Guardar");
        chooser.setMultiSelectionEnabled(false);
        int sel = chooser.showOpenDialog(null);
        if (sel == JFileChooser.APPROVE_OPTION) {
            archivo = chooser.getSelectedFile();
            fr = new FileReader(archivo);
            br = new BufferedReader(fr);
            // Lectura del fichero
            String linea;
            while ((linea = br.readLine()) != null) {
                //linea por linea se escribe en el jTextArea llamado entrada
                entrada.setText(entrada.getText() + linea + "\n");
            }
        }
    }catch(HeadlessException | IOException e){
        e.printStackTrace();
    }finally{
     // En el finally cerramos el fichero, para asegurarnos
     // que se cierra tanto si todo va bien como si salta
     // una excepcion.
    try{
        if( null != fr ){
          fr.close();
        }
     }catch (Exception e2){
        e2.printStackTrace();
     }
    }
}

private void encriptarbtnActionPerformed(java.awt.event.ActionEvent  evt) {
    // TODO add your handling code here:
    salida.setText("");
    if(!"".equals(entrada.getText())&&!"".equals(llave.getText())){
        Vector entradabits=new Vector();
        boolean[] paquete;
        String binario="";

        //Se convierte el contenido del jTextArea a un arreglo de caracteres
        String entradaarray=entrada.getText().trim().toString();

        //Mediante  el ciclo se convierte caracter a caracter a binario de 8 bits
        for(int i=0;i<entradaarray.length();i++){
            binario=binario+Binario(entradaarray.charAt(i));
        }

        //Ahora obtendremos los paquetes de 64 bits de la cadena ya convertida a binario
        //Una variable auxiliar para almacenar el string de tamaño 64
```

```
        String sub;
        //Este ciclo se repetirá tantas veces como elementos tenga la cadena binario
        while(!"".equals(binario)){
            //Si la longitud de binario es menor a 64 ya no completamos otro paquete
            if(binario.length()<64){
                //Vaciamos lo que tenga el arreglo paquete para almacenar el nuevo
                paquete=new boolean[64];
                //Calculamos cuantos bits le hacen falta al paquete para tener una longitud de
64
                int restante=64-binario.length();
                //Aquí rellenamos esa cantidad restante de bits con ceros
                for(int x=0;x<restante;x++){
                    binario="0"+binario;
                }
                //Ahora sacamos todos los caracteres del paquete y verificamos si es un uno
o cero para llenar el arreglo de booleanos
                System.out.print("Paquete : ");
                for(int x=0;x<64;x++){
                    paquete[x] = binario.charAt(x)=='1';
                    System.out.print(paquete[x]+", ");
                }
                System.out.print("\n");
                //Agregamos el arreglo paquete al objeto Vector para separarlo
                entradabits.add(paquete);
                //Se sale del ciclo ya que se acabaron los paquetes
                break;
            }else{
                //Si la longitud es mayor o igual a 64 entonces aun nos quedan paquetes por
procesar
                //Vaciamos lo que tenga el arreglo paquete para almacenar el nuevo
                paquete=new boolean[64];
                //Extraemos una subcadena de 64 posiciones de la cadena binario para
obtener los 64 bits
                sub=binario.substring(0, 64);
                //Ahora sacamos todos los caracteres del paquete y verificamos si es un uno
o cero para llenar el arreglo de booleanos
                System.out.print("Paquete : ");
                for(int x=0;x<64;x++){

                    paquete[x] = sub.charAt(x)=='1';
                    System.out.print(paquete[x]+", ");
                }
                System.out.print("\n");
                //Agregamos el arreglo paquete al objeto Vector para separarlo
                entradabits.add(paquete);
                //Eliminamos esos 64 bits procesados de la cadena para continuar con los
otros 64
                binario=binario.substring(64,binario.length());
            }
        }
```

```
        //Dependiendo del tamaño del objeto Vector es las veces que se ejecutará DES,
debido que solo puede encriptar 64 bits
        for(int y=0;y<entradabits.size();y++){
            paquete=(boolean [])entradabits.get(y);
            System.out.println("ENTRADABITS");
            for(int x=0;x<64;x++){
                System.out.print(paquete[x]+", ");
            }
            System.out.println("");
            salida.setText(salida.getText()+ASCII(des.encriptar(paquete)));

        }
      }else{
          JOptionPane.showMessageDialog(null, "INGRESA TEXTO EN LA ENTRADA O
CARGA UN ARCHIVO","ERROR",JOptionPane.ERROR_MESSAGE);
      }

   }

   private void jButton8ActionPerformed(java.awt.event.ActionEvent  evt) {
      // TODO add your handling code here:
      if(llave.getText().toString().length()<9){
          if(!"".equals(llave.getText())){
              boolean[] paquete;
              String binario="";

              //Se convierte el contenido del jTextArea a un arreglo de caracteres
              char[] entradaarray=llave.getText().trim().toCharArray();

              //Mediante  el ciclo se convierte caracter a caracter a binario de 8 bits
              for(int i=0;i<entradaarray.length;i++){
                  binario=binario+Binario(entradaarray[i]);
              }

              //Ahora obtendremos los paquetes de 64 bits de la cadena ya convertida a
binario
              //Una variable auxiliar para almacenar el string de tamaño 64
              String sub;
              //Este ciclo se repetirá tantas veces como elementos tenga la cadena binario
                    //Vaciamos lo que tenga el arreglo paquete para almacenar el nuevo
                    paquete=new boolean[64];
                    //Calculamos cuantos bits le hacen falta al paquete para tener una longitud
de 64
                    int restante=64-binario.length();
                    //Aquí rellenamos esa cantidad restante de bits con ceros
                    for(int x=0;x<restante;x++){
                        binario="0"+binario;
                    }
                    //Ahora sacamos todos los caracteres del paquete y verificamos si es un
uno o cero para llenar el arreglo de booleanos
                    for(int x=0;x<64;x++){
```

```
                    paquete[x] = binario.charAt(x)=='1';
                }

            des=new DES(paquete);
            encriptarbtn.setEnabled(true);
          }else{
            JOptionPane.showMessageDialog(null, "INGRESA UNA
LLAVE","ERROR",JOptionPane.ERROR_MESSAGE);

          }
       }else{
          JOptionPane.showMessageDialog(null, "EL TAMAÑO MAXIMO DE LA LLAVE ES
DE 8 CARACTERES","ERROR",JOptionPane.ERROR_MESSAGE);
       }
   }

   private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
       // TODO add your handling code here:
       salida.setText("");
   }

   /**
    * @param args the command line arguments
    */
   public static void main(String args[]) {
       /* Set the Nimbus look and feel */
       //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
       /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and
feel.
        * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
        */
       try {
          for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
             if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
             }
          }
       } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(GUI.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
       } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(GUI.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
       } catch (IllegalAccessException ex) {
```

```
java.util.logging.Logger.getLogger(GUI.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException  ex) {

java.util.logging.Logger.getLogger(GUI.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new  Runnable() {
            public void run() {
                new GUI().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton encriptarbtn;
    private javax.swing.JTextArea  entrada;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton8;
    private javax.swing.JLabel  jLabel1;
    private javax.swing.JLabel  jLabel2;
    private javax.swing.JLabel  jLabel3;
    private javax.swing.JLabel  jLabel4;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JTextField  llave;
    private javax.swing.JTextArea  salida;
    // End of variables declaration
}
```

```
public class DES {
    //MATRICES
    //PERMUTACION  INICIAL
    private int IP[][]=new int[8][8];
    //PERMUTACION  INICIAL  INVERSA
    private int IPR[][]={{40,8,48,16,56,24,64,32},
                         {39,7,47,15,55,23,63,31},
                         {38,6,46,14,54,22,62,30},
                         {37,5,45,13,53,21,61,29},
                         {36,4,44,12,52,20,60,28},
                         {35,3,43,11,51,19,59,27},
                         {34,2,42,10,50,18,58,26},
                         {33,1,41,9,49,17,57,25}};
    //PERMUTACION  DE EXPANSION  E
    private int E[][]=new int[8][6];
    //FUNCION  DE PERMUTACION  P
    private int P[][]=new int[4][8];

    //PERMUTACION  OPCION UNO
    private int APC1[][]=new int[4][7];
    private int BPC1[][]=new int[4][7];

    //PERMUTACION  OPCION DOS
    private int PC2[][]=new int[6][8];

    //PROGRAMACION  DE ROTACIONES A LA IZQUIERDA
    private int SCH[]=new int[16];

    //MATRIZ  DE 16 LLAVES
    boolean llaves[][]=new boolean[16][48];
    boolean llavesdes[][]=new boolean[16][48];

    //S-BOXES
    private int Sboxes[][][]=
                    {{{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}},


                     {{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}},


                     {{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}},
```

```
{{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}},


{{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}},


{{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}},


{{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}},


{{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}}};

   DES(boolean[] llave){
      //PERMUTACION INICIAL
      IP[0][0]=58; IP[0][1]=50; IP[0][2]=42; IP[0][3]=34; IP[0][4]=26; IP[0][5]=18; IP[0][6]=10;
IP[0][7]=2;
      IP[1][0]=60; IP[1][1]=52; IP[1][2]=44; IP[1][3]=36; IP[1][4]=28; IP[1][5]=20; IP[1][6]=12;
IP[1][7]=4;
      IP[2][0]=62; IP[2][1]=54; IP[2][2]=46; IP[2][3]=38; IP[2][4]=30; IP[2][5]=22; IP[2][6]=14;
IP[2][7]=6;
      IP[3][0]=64; IP[3][1]=56; IP[3][2]=48; IP[3][3]=40; IP[3][4]=32; IP[3][5]=24; IP[3][6]=16;
IP[3][7]=8;
      IP[4][0]=57; IP[4][1]=49; IP[4][2]=41; IP[4][3]=33; IP[4][4]=25; IP[4][5]=17; IP[4][6]=9;
IP[4][7]=1;
      IP[5][0]=59; IP[5][1]=51; IP[5][2]=43; IP[5][3]=35; IP[5][4]=27; IP[5][5]=19; IP[5][6]=11;
IP[5][7]=3;
      IP[6][0]=61; IP[6][1]=53; IP[6][2]=45; IP[6][3]=37; IP[6][4]=29; IP[6][5]=21; IP[6][6]=13;
IP[6][7]=5;
      IP[7][0]=63; IP[7][1]=55; IP[7][2]=47; IP[7][3]=39; IP[7][4]=31; IP[7][5]=23; IP[7][6]=15;
IP[7][7]=7;
```

```
        //PERMUTACION DE EXPANSION E
        E[0][0]=32; E[0][1]=1;  E[0][2]=2;  E[0][3]=3;  E[0][4]=4;  E[0][5]=5;
        E[1][0]=4;  E[1][1]=5;  E[1][2]=6;  E[1][3]=7;  E[1][4]=8;  E[1][5]=9;
        E[2][0]=8;  E[2][1]=9;  E[2][2]=10; E[2][3]=11; E[2][4]=12; E[2][5]=13;
        E[3][0]=12; E[3][1]=13; E[3][2]=14; E[3][3]=15; E[3][4]=16; E[3][5]=17;
        E[4][0]=16; E[4][1]=17; E[4][2]=18; E[4][3]=19; E[4][4]=20; E[4][5]=21;
        E[5][0]=20; E[5][1]=21; E[5][2]=22; E[5][3]=23; E[5][4]=24; E[5][5]=25;
        E[6][0]=24; E[6][1]=25; E[6][2]=26; E[6][3]=27; E[6][4]=28; E[6][5]=29;
        E[7][0]=28; E[7][1]=29; E[7][2]=30; E[7][3]=31; E[7][4]=32; E[7][5]=1;
        System.out.println("MATRIZ  E: ");
        for(int a=0;a<8;a++){
            for(int b=0;b<6;b++){
                System.out.print((E[a][b]-1)+", ");
            }
        }

        //FUNCION DE PERMUTACION  P
        P[0][0]=16; P[0][1]=7;  P[0][2]=20;  P[0][3]=21;  P[0][4]=29; P[0][5]=12;  P[0][6]=28;
P[0][7]=17;
        P[1][0]=1;  P[1][1]=15; P[1][2]=23;  P[1][3]=26;  P[1][4]=5;  P[1][5]=18;  P[1][6]=31;
P[1][7]=10;
        P[2][0]=2;  P[2][1]=8;  P[2][2]=24;  P[2][3]=14;  P[2][4]=32; P[2][5]=27;  P[2][6]=3;
P[2][7]=9;
        P[3][0]=19; P[3][1]=13; P[3][2]=30;  P[3][3]=6;   P[3][4]=22; P[3][5]=11;  P[3][6]=4;
P[3][7]=25;

        //PERMUTACION OPCION UNO
        APC1[0][0]=57; APC1[0][1]=49; APC1[0][2]=41;  APC1[0][3]=33;  APC1[0][4]=25;
APC1[0][5]=17;  APC1[0][6]=9;
        APC1[1][0]=1;  APC1[1][1]=58; APC1[1][2]=50;  APC1[1][3]=42;  APC1[1][4]=34;
APC1[1][5]=26;  APC1[1][6]=18;
        APC1[2][0]=10; APC1[2][1]=2;  APC1[2][2]=59;  APC1[2][3]=51;  APC1[2][4]=43;
APC1[2][5]=35;  APC1[2][6]=27;
        APC1[3][0]=19; APC1[3][1]=11; APC1[3][2]=3;   APC1[3][3]=60;  APC1[3][4]=52;
APC1[3][5]=44;  APC1[3][6]=36;

        BPC1[0][0]=63; BPC1[0][1]=55; BPC1[0][2]=47;  BPC1[0][3]=39;  BPC1[0][4]=31;
BPC1[0][5]=23;  BPC1[0][6]=15;
        BPC1[1][0]=7;  BPC1[1][1]=62; BPC1[1][2]=54;  BPC1[1][3]=46;  BPC1[1][4]=38;
BPC1[1][5]=30;  BPC1[1][6]=22;
        BPC1[2][0]=14; BPC1[2][1]=6;  BPC1[2][2]=61;  BPC1[2][3]=53;  BPC1[2][4]=45;
BPC1[2][5]=37;  BPC1[2][6]=29;
        BPC1[3][0]=21; BPC1[3][1]=13; BPC1[3][2]=5;   BPC1[3][3]=28;  BPC1[3][4]=20;
BPC1[3][5]=12;  BPC1[3][6]=4;

        //PERMUTACION OPCION DOS
        PC2[0][0]=14;  PC2[0][1]=17; PC2[0][2]=11;  PC2[0][3]=24;  PC2[0][4]=1;
PC2[0][5]=5;   PC2[0][6]=3;   PC2[0][7]=28;
        PC2[1][0]=15;  PC2[1][1]=6;  PC2[1][2]=21;  PC2[1][3]=10;  PC2[1][4]=23;
PC2[1][5]=19;  PC2[1][6]=12;  PC2[1][7]=4;
```

```
        PC2[2][0]=26;  PC2[2][1]=8;  PC2[2][2]=16;  PC2[2][3]=7;   PC2[2][4]=27;
PC2[2][5]=20;  PC2[2][6]=13;  PC2[2][7]=2;
        PC2[3][0]=41;  PC2[3][1]=52; PC2[3][2]=31;  PC2[3][3]=37;  PC2[3][4]=47;
PC2[3][5]=55;  PC2[3][6]=30;  PC2[3][7]=40;
        PC2[4][0]=51;  PC2[4][1]=45; PC2[4][2]=33;  PC2[4][3]=48;  PC2[4][4]=44;
PC2[4][5]=49;  PC2[4][6]=39;  PC2[4][7]=56;
        PC2[5][0]=34;  PC2[5][1]=53; PC2[5][2]=46;  PC2[5][3]=42;  PC2[5][4]=50;
PC2[5][5]=36;  PC2[5][6]=29;  PC2[5][7]=32;

        //PROGRAMACION DE ROTACIONES A LA IZQUIERDA
        SCH[0]=1;
        SCH[1]=1;
        SCH[2]=2;
        SCH[3]=2;
        SCH[4]=2;
        SCH[5]=2;
        SCH[6]=2;
        SCH[7]=2;
        SCH[8]=1;
        SCH[9]=2;
        SCH[10]=2;
        SCH[11]=2;
        SCH[12]=2;
        SCH[13]=2;
        SCH[14]=2;
        SCH[15]=1;


        //LLAVES
        llaves=obtencionllaves(llave);

           for(int b=0;b<48;b++){
              llavesdes[0][b]=llaves[15][b];
              llavesdes[1][b]=llaves[14][b];
              llavesdes[2][b]=llaves[13][b];
              llavesdes[3][b]=llaves[12][b];
              llavesdes[4][b]=llaves[11][b];
              llavesdes[5][b]=llaves[10][b];
              llavesdes[6][b]=llaves[9][b];
              llavesdes[7][b]=llaves[8][b];
              llavesdes[8][b]=llaves[7][b];
              llavesdes[9][b]=llaves[6][b];
              llavesdes[10][b]=llaves[5][b];
              llavesdes[11][b]=llaves[4][b];
              llavesdes[12][b]=llaves[3][b];
              llavesdes[13][b]=llaves[2][b];
              llavesdes[14][b]=llaves[1][b];
              llavesdes[15][b]=llaves[0][b];
           }

        for(int a=0;a<16;a++){
```

```
        System.out.print("\nLlave "+(a+1)+": ");

       for(int b=0;b<48;b++){
          System.out.print(llaves[a][b]+", ");
       }
    }

    for(int a=0;a<16;a++){
        System.out.print("\nLlave inversa"+(a+1)+": ");

       for(int b=0;b<48;b++){
          System.out.print(llavesdes[a][b]+", ");
       }
    }
}

//Este metodo rota el arreglo segun el schedule de rotaciones
private boolean[] rotarIzq(boolean [] arreglo, int posiciones){
    for (int i = 0; i < posiciones; i++) {

        boolean aux = arreglo[0];

        for (int j = 0; j < arreglo.length-1; j++) {
            arreglo[j] = arreglo[j + 1];
        }

        arreglo[arreglo.length-1] = aux;
    }

    return arreglo;
}

private boolean[][] obtencionllaves(boolean[]llave){
    boolean llavesgen[][]=new boolean[16][48];
    boolean llaveizq[]=new boolean[28];
    boolean llaveder[]=new boolean[28];
    int cont=0;
    //PERMUTACION OPCION 1
    System.out.print("\nLlave: ");
    for(int x=0;x<64;x++){
        System.out.print(llave[x]+", ");
    }
    for (int a = 0; a < 4; a++) {
        for (int b = 0; b < 7; b++) {
            llaveizq[cont]=llave[APC1[a][b]-1];
            llaveder[cont]=llave[BPC1[a][b]-1];
            cont++;
        }
    }
    System.out.print("\nLlave izq: ");
    for(int x=0;x<28;x++){
```

```
    System.out.print(llaveizq[x]+", ");
}
System.out.print("\nLlave der: ");
for(int x=0;x<28;x++){
    System.out.print(llaveder[x]+", ");
}

//GENERACION DE 16 SUBLLAVES
boolean temp[]=new boolean[56];
boolean []llaveizqantes=new boolean[28];
boolean []llavederantes=new boolean[28];
llaveizqantes=llaveizq;
llavederantes=llaveder;
for(int x=0;x<16;x++){

    //CORRIMIENTO A LA IZQ SEGUN EL SCHEDULE
    llaveizq=rotarIzq(llaveizqantes, SCH[x]);

    System.out.println("SCHEDULE: "+SCH[x]);

    llaveder=rotarIzq(llavederantes, SCH[x]);
    llaveizqantes=llaveizq;
    llavederantes=llaveder;

    //JUNTAR LOS DOS ARREGLOS PARA HACER LA PERMUTACION OPCION 2
    for(int b=0;b<28;b++){
        temp[b]=llaveizq[b];
    }
    cont=0;
    for(int b=28;b<56;b++){
        temp[b]=llaveder[cont];
        cont++;
    }

    cont=0;
    //PERMUTACION OPCION 2
    boolean permutado2[]=new boolean[48];
    for (int a = 0; a < 6; a++) {
        for (int b = 0; b < 8; b++) {
            permutado2[cont] = temp[PC2[a][b]-1];
            if(cont<48){
                cont++;
            }
        }
    }

    //GUARDAR LAS SUBLLAVES EN EL ARREGLO DE 16
    for(int y=0;y<48;y++){
        llavesgen[x][y]=permutado2[y];
    }
}
```

```
    return llavesgen;
}

private int binarioADecimal(boolean []bits){
    int decimal=0;
    String bitstring="";
    int longitud=bits.length;
    int potencia = longitud - 1;

    //Convertimos el arreglo booleano a un String
    for(int x=0;x<longitud;x++){
        if(bits[x]==true){
            bitstring=bitstring+"1";
        }else{
            bitstring=bitstring+"0";
        }

    }
    //Conversion del string binario a numero entero
    for(int x=0;x<longitud;x++){
       if(bitstring.charAt(x)=='1'){
        decimal+= Math.pow(2, potencia);
       }
       potencia --;
    }
    return decimal;
}

private boolean[] Binario(int Decimal) {
    String resultado="";
    int temp=Decimal;
    //Se aplica el algoritmo para convertir a binario
    while(temp != 0){
        if(temp %2 == 0){
          resultado="0"+resultado;
        }else{
            resultado="1"+resultado;
        }
        temp = temp/2;
    }

    //Se le agregan ceros a la izquierda, para completar 8 bits por caracter
    int cerosalaizquierda=4-resultado.length();
    for(int x=0;x<cerosalaizquierda;x++){
        resultado="0"+resultado;
    }

    boolean[] cuatrobits=new boolean[4];

    for(int x=0;x<resultado.length();x++){
```

```
            cuatrobits[x]=resultado.charAt(x)=='1';
        }
        return cuatrobits;
    }

    public String encriptar(boolean[] paquete){
        String encriptado = "";

        //Se realiza la permutacion inicial
        boolean permutado[]=new boolean[64];
        int cont=0;
        System.out.print("Permutacion inicial : ");
        for(int a=0;a<8;a++){
            for(int b=0;b<8;b++){
                permutado[cont]=paquete[IP[a][b]-1];
                System.out.print(permutado[cont]+", ");
                cont++;
            }
        }
        System.out.print("\nMitad izq : ");

        boolean[] izq=new boolean[32];
        boolean[] der=new boolean[32];
        boolean[] derantes=new boolean[32];
        //Separacion del paquete para mitad izq y der
        for(int x=0;x<32;x++){
            izq[x]=permutado[x];
            System.out.print(izq[x]+", ");
            der[x]=permutado[x+32];
        }
        System.out.print("\nMitad der : ");
        for(int x=0;x<32;x++){
            System.out.print(der[x]+", ");
        }


        boolean expansion[]=new boolean[48];
        boolean xor[]=new boolean[48];
        boolean sboxeado[]=new boolean[32];
        boolean entradasbox[][]=new boolean[8][6];

        cont=0;
        //COMIENZAN LAS RONDAS
        for(int i=0;i<16;i++){
            //Backup del lado derecho para pasar a ser el izquierdo al final de la ronda
            System.out.println("Der. al entrar a ronda: ");
            for(int x=0;x<32;x++){
                derantes[x]=der[x];
                if(der[x]==true){
                    System.out.println('1');
                }else{
```

```
            System.out.println('0');
        }
    }
    //PERMUTACION  DE EXPANSION
    System.out.print("\nPermutacion de expansion : ");
    cont=0;
    char bit='1';
    for (int a = 0; a < 8; a++) {
        for (int b = 0; b < 6; b++) {
            if(der[E[a][b]-1]==false){
                bit='0';
            }else{
                bit='1';
            }
            System.out.println("Bit: "+bit+" Es el de la pos: "+(E[a][b]-1));
            expansion[cont] = der[E[a][b]-1];
            //System.out.print(expansion[cont]+", ");
            cont++;
        }
    }
    //XOR CON LA LLAVE DE
    System.out.print("\nXOR con llave");
    for (int a = 0; a < 48; a++) {
        xor[a] = expansion[a] != llaves[i][a];
        System.out.print(xor[a]+", ");
    }

    //Preparamos la entrada a 8 paquetes de 6 bits
    cont=0;
    System.out.print("\nEntradas a sbox: \n");
    for (int a = 0; a < 8; a++) {
        System.out.print((a+1) +". ");
        for (int b = 0; b < 6; b++) {
            entradasbox[a][b] = xor[cont];
            System.out.print(entradasbox[a][b]+", ");
            cont++;
        }
        System.out.print("\n");
    }

    //Comienza el proceso de s-box
    cont=0;
    boolean cuatrobits[]=new boolean[4];
    boolean dosbits[]=new boolean[2];
    for (int a = 0; a < 8; a++) {
        //System.out.println("ENTRE: "+a);
        //Separación de los 4 bits de enmedio para la columna de la sbox
        //System.out.print("\nSBOX: "+a);
        dosbits[0]=entradasbox[a][0];
        cuatrobits[0]=entradasbox[a][1];
        cuatrobits[1]=entradasbox[a][2];
```

```
            cuatrobits[2]=entradasbox[a][3];
            cuatrobits[3]=entradasbox[a][4];
            dosbits[1]=entradasbox[a][5];
            System.out.println("Renglon: "+binarioADecimal(dosbits));
            System.out.println("Columna: "+binarioADecimal(cuatrobits));
            System.out.println("SBOX: "+a);
            System.out.println("SBOX: ");
            for(int x=0;x<4;x++){
                for(int y=0;y<16;y++){
                    System.out.print(Sboxes[a][x][y]+", ");
                }
            }
            boolean
temp[]=Binario(Sboxes[a][binarioADecimal(dosbits)][binarioADecimal(cuatrobits)]);
            for(int b=0;b<4;b++){
                sboxeado[cont]=temp[b];
                //System.out.println("VALOR SBOX:
"+Sboxes[a][0][binarioADecimal(cuatrobits)]);
                //System.out.print(sboxeado[cont]+", ");
                cont++;
            }

            //System.out.print("\n");
        }

        System.out.println("SBOXEADO: ");
        for(int x=0;x<sboxeado.length;x++){
            System.out.print(sboxeado[x]);
        }
        //Permutacion P

        boolean []permutacionp=new boolean[32];
        cont=0;
        System.out.print("\nPermutacion P: ");
        for(int a=0;a<4;a++){
            for(int b=0;b<8;b++){
                permutacionp[cont]=sboxeado[P[a][b]-1];
                System.out.print(permutacionp[cont]+", ");
                cont++;
            }
        }
        System.out.print("\nIZQ: ");
        for(int x=0;x<32;x++){
            System.out.print(izq[x]+", ");
        }
        //XOR CON LA MITAD IZQUIERDA
        System.out.print("\nXOR CON LA MITAD IZQ: ");
        for(int x=0;x<32;x++){
            der[x]=permutacionp[x] != izq[x];
            System.out.print(der[x]+", ");
        }
```

```
            for(int x=0;x<32;x++){
                izq[x]=derantes[x];
            }
            System.out.print("\n Mitad izq despues de ronda: ");
            for(int x=0;x<32;x++){
                System.out.print(izq[x]+", ");
            }
            System.out.print("\n Mitad der despues de ronda: ");
            for(int x=0;x<32;x++){
                System.out.print(der[x]+", ");
            }

        }

        //PEGAR MITADES

        for(int x=0;x<32;x++){
            paquete[x]=der[x];
            paquete[x+32]=izq[x];
        }

        //PERMUTACION INVERSA
        System.out.print("\nPERMUTACION INVERSA: ");
        boolean [] permutadofinal=new boolean[64];
        cont=0;
        for(int a=0;a<8;a++){
            for(int b=0;b<8;b++){
                permutadofinal[cont]=paquete[IPR[a][b]-1];
                System.out.print(permutadofinal[cont]+", ");
                cont++;
            }
        }

        //Concatenado del arreglo a un string
        for(int x=0;x<64;x++){
            if(permutadofinal[x]==true){
                encriptado=encriptado+"1";
            }else{
                encriptado=encriptado+"0";
            }
        }
        System.out.print("\n Encriptado: "+encriptado);

        return encriptado;
    }

}
```