
# Instituto Tecnológico de Durango

**Ing. En Sistemas Computacionales**
**Criptografía**
**Encriptado: “AES”**

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

**Victoria de Durango, Dgo.**

**28 DE ABRIL DEL 2016**

## Introducción

En la clase de criptografiá se nos encargó realizar la implementación del algoritmo AES para realizar cifrado de datos. El algoritmo es relativamente sencillo, lo complejo es la matemática que tiene detrás, tanto que aún es considerado seguro para ser utilizado.

## Desarrollo

Lo primero que desarrollé fué la interfaz gráfica, en esta ocación no utilicé la herramienta gráfica de Netbeans que genera el código de la interfaz, sino que opté por realizar la interfaz escribiendo yo el código, esto para reducir demasiado el tamaño del mismo porque Netbeans genera mucho código basura.

```
JFrame ventana;
JLabel titulo;
JTextField llavetextfield;
JTextField entradatextfield;
JButton llavebtn;
JButton encriptarbtn;
JTextArea salida;

byte[][]SBOX=new byte[16][16];


palabra []word=new palabra[44];

AES(){
    //Interfaz Gráfica
    ventana=new JFrame();
    JPanel panel=new JPanel();
    panel.setLayout(new BorderLayout());

    titulo=new JLabel("AES",SwingConstants.CENTER);
    titulo.setFont(new Font(titulo.getName(), Font.PLAIN, 40));
    panel.add(titulo, BorderLayout.NORTH);

    JPanel bloque=new JPanel();
    JLabel llavelabel=new JLabel("Llave:");
    llavetextfield=new JTextField(40);
    JLabel entradalabel=new JLabel("Texto A Cifrar:");
    entradatextfield=new JTextField(35);
    bloque.add(llavelabel);
    bloque.add(llavetextfield);
    bloque.add(entradalabel);
    bloque.add(entradatextfield);

    llavebtn=new JButton("GENERAR LLAVES");
    llavebtn.setPreferredSize(new Dimension(400,40));
    bloque.add(llavebtn);
```

```
    encriptarbtn=new JButton("ENCRIPTAR");
    encriptarbtn.setPreferredSize(new Dimension(400,40));
    encriptarbtn.setEnabled(false);
    bloque.add(encriptarbtn);
    panel.add(bloque,BorderLayout.CENTER);

    salida=new JTextArea();
    salida.setRows(10);
    salida.setEditable(false);
    panel.add(salida,BorderLayout.SOUTH);

    ventana.setContentPane(panel);
    ventana.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    ventana.setSize(500, 400);
    ventana.setResizable(false);
    ventana.setVisible(true);

    //Accion al presionar el botón "ENCRIPTAR"
    encriptarbtn.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            encriptar();
        }
    });

    llavebtn.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            encriptarbtn.setEnabled(true);
            generarLlaves();
        }
    });

    final int limite = 16;
    KeyListener keyListener = new KeyListener() {
    public void keyPressed(KeyEvent keyEvent) {
    }

    public void keyReleased(KeyEvent keyEvent) {
    }

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

    }
};
    llavetextfield.addKeyListener(keyListener);
    KeyListener keyListener2 = new KeyListener() {
    public void keyPressed(KeyEvent keyEvent) {
```

```
        }

        public void keyReleased(KeyEvent keyEvent) {
        }

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

        }
    };

        entradatextfield.addKeyListener(keyListener2);

        genSbox();
    }
```

## Generación de Matriz SBOX

Como vemos al final del constructor el siguiente paso fue crear la matriz para las sustituciones. Para esto existe un método en el tipo Short para transformar una cadena, como por ejemplo “FF” en su valor de byte; indicando solamente la base en la que están expresados.

```
public void genSbox(){
                                String    [][]    sboxstring=new    String[][]
{{"63","7c","77","7b","f2","6b","6f","c5","30","01","67","2b","fe","d7","ab
","76"},
                                                  {"ca","82","c9","7d","fa","59
","47","f0","ad","d4","a2","af","9c","a4","72","c0"},
                                                  {"b7","fd","93","26","36","3f
","f7","cc","34","a5","e5","f1","71","d8","31","15"},
                                                  {"04","c7","23","c3","18","96
","05","9a","07","12","80","e2","eb","27","b2","75"},
                                                  {"09","83","2c","1a","1b","6e
","5a","a0","52","3b","d6","b3","29","e3","2f","84"},
                                                  {"53","d1","00","ed","20","fc
","b1","5b","6a","cb","be","39","4a","4c","58","cf"},
                                                  {"d0","ef","aa","fb","43","4d
","33","85","45","f9","02","7f","50","3c","9f","a8"},
                                                  {"51","a3","40","8f","92","9d
","38","f5","bc","b6","da","21","10","ff","f3","d2"},
                                                  {"cd","0c","13","ec","5f","97
","44","17","c4","a7","7e","3d","64","5d","19","73"},
                                                  {"60","81","4f","dc","22","2a
","90","88","46","ee","b8","14","de","5e","0b","db"},
                                                  {"e0","32","3a","0a","49","06
","24","5c","c2","d3","ac","62","91","95","e4","79"},
```

```
                                                {"e7","c8","37","6d","8d","d5
","4e","a9","6c","56","f4","ea","65","7a","ae","08"},
                                                {"ba","78","25","2e","1c","a6
","b4","c6","e8","dd","74","1f","4b","bd","8b","8a"},
                                                {"70","3e","b5","66","48","03
","f6","0e","61","35","57","b9","86","c1","1d","9e"},
                                                {"e1","f8","98","11","69","d9
","8e","94","9b","1e","87","e9","ce","55","28","df"},
                                                {"8c","a1","89","0d","bf","e6
","42","68","41","99","2d","0f","b0","54","bb","16"}};

         //Se convierte cada string de la anterior matriz a su equivalente
byte y se coloca en la matriz SBOX
        for(int a=0;a<16;a++){
            for(int b=0;b<16;b++){
                SBOX[a][b]=(byte) Short.parseShort(sboxstring[a][b], 16);
            }
        }
```

**Generación de Subllaves**

Para la generación utilicé el algoritmo que nos proporciona Stallings en su libro. Solo que había un inconveniente, en el algoritmo viene un tipo “word” que almacena 4 bytes. Este tipo no existe en Java, por lo que para poder utilizar el mismo algoritmo creé el tipo.

```
public class palabra {
    public byte bait1;
    public byte bait2;
    public byte bait3;
    public byte bait4;

    public palabra(){
    }

    public palabra(byte bait1, byte bait2, byte bait3, byte bait4){
        this.bait1=bait1;
        this.bait2=bait2;
        this.bait3=bait3;
        this.bait4=bait4;
    }

    public byte[] getBytes(){
        byte[] baits=new byte[4];
        baits[0]=bait1;
        baits[1]=bait2;
        baits[2]=bait3;
        baits[3]=bait4;
        return baits;
    }
```

```
}
```
Este objeto almacenaría 4 bytes y se obtendrían por medio del método “getBytes()” que retorna un arreglo con los 4 bytes.

Para obtener la llave, convierto los caracteres del JtextField a bytes, mediante un método de la clase String que se llama “getBytes()” que recibe como argumento la codificación de los caracteres que convertirá a bytes. En mi caso usé la ISO-8859-1 ya que tiene más cantidad de caracteres y al momento de encriptar saldrían valores muy raros que no se muestran en pantalla. (Las funciones están definidas en la pág 11)

```
public void generarLlaves(){
         //Se guarda la entrada de la caja de texto en un arreglo de bytes, 
cada caracter es un byte del arreglo
        byte[]llave=null;
        try {
            llave = llavetextfield.getText().getBytes("ISO-8859-1");
        } catch (UnsupportedEncodingException ex) {
              Logger.getLogger(AES.class.getName()).log(Level.SEVERE, null, 
ex);
        }
        System.out.println("Longitud llave: "+llave.length);
        printByteArray(llave);
        palabra temp=new palabra();
        for(int i=0; i<4;i++){
                                                                word[i]=new 
palabra(llave[4*i],llave[4*i+1],llave[4*i+2],llave[4*i+3]);
            System.out.println("Palabra "+i+": ");
            imprimirPalabras(word[i]);
        }

        byte[] RC=genRC();
        int cont=0;
        for (int i=4;i<44;i++){
            temp = word[i-1];
            System.out.print("\n");
            imprimirPalabras(temp);
            if (i % 4 == 0){
                byte[] baits=temp.getBytes();
                System.out.println("temp: ");
                printByteArray(baits);
                baits=rotarIzq(baits,1);
                System.out.println("Rotar ");
                printByteArray(baits);
                for(int b=0;b<4;b++){
                    int[] filacolumna=getFilaColumna(baits[b]);
                    baits[b]=SBOX[ filacolumna[0] ][ filacolumna[1] ];
             System.out.println("SBOX "+b+" :"+Integer.toHexString(baits[b] 
& 0xFF));
```

```
                }
                baits[0]=xor(baits[0],RC[cont]);
                cont++;
                temp=new palabra(baits[0],baits[1],baits[2],baits[3]);
            }
            word[i]=xor2arrays(word[i-4].getBytes(),temp.getBytes());
            System.out.println("Palabra "+i+": ");
            imprimirPalabras( word[i]);
        }
    }
```

**Encriptado**

Para el encriptado utilizo el mismo método para obtener la entrada a un arreglo de 128 bytes, que después paso a una matriz de 4x4, la cual es el estado en el cuál se realizaran las operaciones.

Para agregar la primera llave antes de las rondas saco las primeras 4 palabras de mi arreglo de palabras, las convierto a matriz de 4x4 bytes y hago el XOR. (Para ver el método de palabrasAMatriz(), ir a la sección de funciones en la pág 11).

Después genero una matriz de 4 de ancho x 11 de alto para almacenar las otras palabras; 4 por renglón.

En la mezcla de columnas no utilizo la matriz constante para la multiplicación de matrices, porque Stallings menciona que una columna de la mezcla puede estar dada por 4 operaciones ya fijas

```
public void encriptar(){
        try {
               //Generar matriz estado a partir de la caja de texto. Cada
caracter será un byte
                //El método getBytes es un método predefinido de la clase
String y da como resultado un arreglo de bytes

byte[][]estado=generarEstado(entradatextfield.getText().getBytes("ISO-8859-
1"));

            System.out.println("ESTADO");
            imprimirEstado(estado);
            //Agregar llave
                                palabra[]primeraspalabras=new   palabra[]
{word[0],word[1],word[2],word[3]};
            estado=xormatrices(estado,palabrasAMatriz(primeraspalabras));

            System.out.println("xor primero");
            imprimirEstado(estado);
               //Acomodo de las llaves a matriz para poder usarlas en las
rondas
            palabra[][]palabras=new palabra[11][4];
```

```
            int cont=4;
            for(int a=0;a<10;a++){
                for(int b=0;b<4;b++){
                    palabras[a][b]=word[cont];
                    cont++;
                }
            }

            //Rondas
            for(int x=0;x<9;x++){
                //Sustitucion de bytes
                for(int a=0;a<4;a++){
                    for(int b=0;b<4;b++){
                         //Se obtiene la fila con los 4 bits de la izq y la
columna con los bits de la derecha
                        int[] filacolumna=getFilaColumna(estado[a][b]);
                        estado[a][b]=SBOX[ filacolumna[0] ][ filacolumna[1]
];
                    }
                }
                System.out.println("Sustitucion");
                imprimirEstado(estado);

                //Rotación de filas
                byte aux[]=new byte[4];
                for(int a=0;a<4;a++){
                    for(int b=0;b<4;b++){
                        //Obtener la fila de la matriz estado para rotarla
                        aux[b]=estado[a][b];
                    }

                         //rotación a la izquierda dependiendo de la fila
fila0=0 posiciones fila3=3 posiciones
                    aux=rotarIzq(aux,a);

                    for(int c=0;c<4;c++){
                        //Regresar la fila ya rotada a la matriz estado
                        estado[a][c]=aux[c];
                    }
                }

                System.out.println("Rotacion");
                imprimirEstado(estado);


                byte[][] mezclada=new byte[4][4];
                //Mezcla de columnas
                for(int a=0;a<4;a++){
```

```
                                        mezclada[0][a]=xorarray(new  byte[]
{multiplicarpor2(estado[0][a]),xor(multiplicarpor2(estado[1][a])
                            ,estado[1][a]),estado[2][a],estado[3][a]});
                              mezclada[1][a]=xorarray(new byte[]{estado[0]
[a],multiplicarpor2(estado[1][a]),
                                 xor(multiplicarpor2(estado[2][a]),estado[2]
[a]),estado[3][a]});
                              mezclada[2][a]=xorarray(new byte[]{estado[0]
[a],estado[1][a],
                                                    multiplicarpor2(estado[2]
[a]),xor(multiplicarpor2(estado[3][a]),estado[3][a])});
                                        mezclada[3][a]=xorarray(new  byte[]
{xor(multiplicarpor2(estado[0][a]),estado[0][a]),
                                                      estado[1][a],estado[2]
[a],multiplicarpor2(estado[3][a])});
                }

                System.out.println("Mezcla");
                imprimirEstado(mezclada);


                //Agregar llave
                        palabra[] cuatropalabras=new palabra[]{palabras[x]
[0],palabras[x][1],palabras[x][2],palabras[x][3]};
                estado=xormatrices(mezclada,palabrasAMatriz(cuatropalabras)
);

                System.out.println("XOR LLAVE");
                imprimirEstado(estado);

            }

            //Ultima ronda
            //Sustitucion de bytes
            for (int a = 0; a < 4; a++) {
                for (int b = 0; b < 4; b++) {
                       //Se obtiene la fila con los 4 bits de la izq y la
columna con los bits de la derecha
                    int[] filacolumna = getFilaColumna(estado[a][b]);
                    estado[a][b] = SBOX[filacolumna[0]][filacolumna[1]];
                }
            }
            System.out.println("Sustitucion Final");
            imprimirEstado(estado);

            //Rotación de filas
            byte aux[] = new byte[4];
            for (int a = 0; a < 4; a++) {
                for (int b = 0; b < 4; b++) {
```

```
                    //Obtener la fila de la matriz estado para rotarla
                    aux[b] = estado[a][b];
                }

                  //rotación a la izquierda dependiendo de la fila fila0=0
posiciones fila3=3 posiciones
                aux = rotarIzq(aux, a);

                for (int c = 0; c < 4; c++) {
                    //Regresar la fila ya rotada a la matriz estado
                    estado[a][c] = aux[c];
                }
            }
            System.out.println("Rotacion final");
            imprimirEstado(estado);
            //Agregar llave
                 palabra[] cuatropalabras = new palabra[]{palabras[9][0],
palabras[9][1], palabras[9][2], palabras[9][3]};
            estado = xormatrices(estado, palabrasAMatriz(cuatropalabras));
            System.out.println("XOR llave final");
            imprimirEstado(estado);

            //Imprimir salida
            //salida.setText(salida.getText()+(estadotoASCII(estado)));
            String hex="";
            for(int a=0;a<4;a++){
                for(int b=0;b<4;b++){
                    hex+=Integer.toHexString(estado[b][a] & 0xFF);
                }
            }
            salida.setText("HEX: \n"+hex);
                                    salida.setText(salida.getText()+"\nASCII:
\n"+estadotoASCII(estado));
        } catch (UnsupportedEncodingException ex) {
              Logger.getLogger(AES.class.getName()).log(Level.SEVERE, null,
ex);
        }

    }
```

**Funciones**

En esta parte están definidas todas las funciones utilizadas en mi programa. Esto me permitió implementar operaciones repetitivas como rotaciones, conversiones y xor más fácilmente.

```
private byte[] rotarIzq(byte[] arreglo, int posiciones){
        for (int i = 0; i < posiciones; i++) {
```

```
        byte aux = arreglo[0];

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

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

    return arreglo;
}
public byte xor(byte byte1, byte byte2){
    // convertir a enteros y hacer el xor
    int one = (int) byte1;
    int two = (int) byte2;
    int xor = one ^ two;

    // convertir de nuevo a byte
    return (byte) (0xff & xor);
}

public byte xorarray(byte[] array){
    for(int x=0;x<array.length-1;x++){
        int one = (int) array[x];
        int two = (int) array[x+1];

        array[x+1]=(byte) (0xff & (one ^ two));
    }

    return array[array.length-1];
}

public palabra xor2arrays(byte[] array1, byte[]array2){

    byte[] temp=new byte[4];
    for(int x=0;x<4;x++){
        temp[x]=xor(array1[x],array2[x]);
    }

    return new palabra(temp[0],temp[1],temp[2],temp[3]);
}

public byte[][] xormatrices(byte[][] matriz1, byte[][]matriz2){
    byte[][] matriz=new byte[4][4];
    for(int a=0;a<4;a++){
        for(int b=0;b<4;b++){
            matriz[a][b]=xor(matriz1[a][b],matriz2[a][b]);
```

```
            }
        }
        return matriz;
    }

    public byte multiplicarpor2(byte bait){
        String baitbinario=Integer.toBinaryString(bait & 0xFF);

        int ceros = 8 - baitbinario.length();
                for (int a = 0; a < ceros; a++) {
                    baitbinario = "0" + baitbinario;
                }

        if(baitbinario.charAt(0)=='1'){
            byte constante=(byte)Short.parseShort("00011011", 2);
            byte temp=(byte)((bait & 0xff) << 1);
            temp=xor(temp,constante);
            return temp;
        }else{
            byte temp=(byte)((bait & 0xff) << 1);
            return temp;
        }
    }

    public int[] getFilaColumna(byte bait){
         //Se obtiene la fila con los 4 bits de la izq y la columna con los
bits de la derecha
         //Se convierte el byte a su cadena hexadecimal FF por ejemplo y se
sacan subcadenas de cada caracter y se convierten a su valor decimal
          //En la posición 0 se encuentra el valor decimal de los 4 bits de
la izq y en la 1 los 4 de la derecha
        int[]filacolumna=new int[2];
        boolean segundo=false;
        try {
            String bt = Integer.toBinaryString(bait & 0xFF);
            if (bt.length() < 8) {
                        int ceros = 8 - Integer.toBinaryString(bait &
0xFF).length();
                for (int a = 0; a < ceros; a++) {
                    bt = "0" + bt;
                }
            }
                filacolumna[0] = Integer.parseInt(bt.substring(0,4), 2);
                filacolumna[1] = Integer.parseInt(bt.substring(4,8), 2);
        } catch (Exception e) {
             JOptionPane.showMessageDialog(null, Integer.toHexString(bait &
0xFF));
        }
```

```
        return filacolumna;
    }

    public byte[] genRC(){
                                    String    []    stringsrc=new    String[]
{"01","02","04","08","10","20","40","80","1B","36"};

        byte[] RC=new byte[10];
        for(int x=0;x<10;x++){
            RC[x]=(byte) Short.parseShort(stringsrc[x], 16);
        }

        return RC;
    }


    public byte[][] generarEstado(byte[] arreglo){
        byte[][]estado=new byte[4][4];
        int cont=0;
        for(int a=0;a<4;a++){
            for(int b=0;b<4;b++){
                estado[b][a]=arreglo[cont];
                cont++;
            }
        }
        return estado;
    }

    public byte[][] palabrasAMatriz(palabra[] cuatrowords){
        byte[][]palabrasmatriz=new byte[4][4];
        System.out.print("WORDS\n");
        for(int a=0;a<4;a++){
            for(int b=0;b<4;b++){
                palabrasmatriz[b][a]=cuatrowords[a].getBytes()[b];
            }
        }
        imprimirEstado(palabrasmatriz);
        return palabrasmatriz;
    }

    public String estadotoASCII(byte[][]estado){
        String ascii="";
        byte[]estadolineal=new byte[16];
        int cont=0;
        for(int a=0;a<4;a++){
            for(int b=0;b<4;b++){
                estadolineal[cont]=estado[b][a];
                cont++;
```

```
            }
        }
        ascii=new String(estadolineal, Charset.forName("ISO-8859-1"));
        return ascii;
    }

    void printByteArray(byte[] array){
        for(int x=0;x<array.length;x++){
            System.out.print(Integer.toHexString(array[x] & 0xFF));
        }
    }


    void imprimirPalabras(palabra w){
        printByteArray(w.getBytes());
    }

    void imprimirEstado(byte[][] estado){
        for(int a=0; a<4;a++){
            for(int b=0;b<4;b++){
                System.out.print(Integer.toHexString(estado[a][b] & 0xFF)+"
");
            }
            System.out.print("\n");
        }
    }
```