Tecnológico de Estudios Superiores del Oriente del Estado de México

Ing. Sistemas Computacionales
Turno: Vespertino
Grupo: “8S21”

Materia: Seguridad Informática
Profesor: Leonardo Palacios Luengas

Alumnos:
Gonzales Mendoza José Antonio
Sánchez Victoria Miguel Ángel
Moreno Nava Ramón
Cervantes Valdés Francisco Javier
Domínguez Andrade Cesar Esteban

Programa de encontrar la clave de cifrados Vigeneré

ATAQUE DE KASISKI

1)Esta parte es la fundamental descifra el mensaje cifrado que metamos en la consola de comandos ya con la contraseña obtenida.

```
for(int f=0;f<Abcedario.length();f++)
{
    if(Msg[c]==Abcedario.charAt(f))
    {
        x=f;

    }
    if(claveEquals[c]==Abcedario.charAt
    {
        y=f;
    }
}
z=(y-x);

if(z<=0)
{
    if(z==0)
    {
        salida+="A";
    }
    else
    {
    for(int j=1;j<=Abcedario.length();j+
    {
     cont++;
     if(cont==(z*-1))
     {
         salida+=Abcedario.charAt(j);
         break;
     }
    }
```

```
 String salida="";
String contraseñaobtenida ;
 String Abcedario="ABCDEFGHIJKLMNÑOPQRSTUVWXYZ";
String clave=contraseñaobtenida;
char []claveEquals=new char [Mensaje.length()];
char []Msg=Mensaje.toUpperCase().toCharArray();
int cont=0;
for(int c=0;c<Mensaje.length();c++)
{

    if(Mensaje.charAt(c)==' ')
    {
        c++;
    }
    claveEquals[c]=clave.charAt(cont);
    cont++;
    if(cont==clave.length())
```

```
{
    Scanner leer = new Scanner(System.in);
    String cadena;
    char opc;
    do{
    System.out.println("Escoge una opccion"+"\n1.-DESCUBRIR CONTRASEÑA"+"\n2.-ESCRIBIR CONTRA
    opc=leer.nextLine().charAt(0);

    if(opc=='1'){
            System.out.println("Introduce el mensaje para descubrir contraseña");
            cadena=leer.nextLine();
            System.out.println("la contraseña es : "+Descubrir(cadena));
        }
    else if(opc=='2'){
          System.out.println("Introduce la contraseña obtenida");
            cadena=leer.nextLine();


    }
    else if(opc=='3'){

            System.out.println("Introduce el mensaje a descifrar");
            cadena=leer.nextLine();
            System.out.println("mensaje en claro :"+Descifrar(cadena));
            }
```

```
Escoge una opccion
1.-DESCUBRIR CONTRASEÑA
2.-ESCRIBIR CONTRASEÑA
3.-DESIFRAR MENSAJE
0.-SALIR
1
Introduce el mensaje para descubrir contraseña
PBVRQ VICAD SKAÑS DETSJ PSIED BGGMP SLRPW RÑPWY EDSDE ÑDRDP CRCPQ MNPWK
UBZVS FNVRD MTIPW UEQVV CBOVN UEDIF QLONM WNUVR SEIKA ZYEAC EYEDS ETFPH
LBHGU ÑESOM EHLBX VAEEP UÑELI SEVEF WHUNM CLPQP MBRRN BPVIÑ MTIBV VEÑID
ANSJA MTJOK MDODS ELPWI UFOZM QMVNF OHASE SRJWR SFQCO TWVMB JGRPW VSUEX
INQRS JEUEM GGRBD GNNIL AGSJI DSVSU EEINT GRUEE TFGGM PORDF OGTSS TOSEQ
OÑTGR RYVLP WJIFW XOTGG RPQRR JSKET XRNBL ZETGG NEMUO TXJAT ORVJH RSFHV
NUEJI BCHAS EHEUE UOTIE FFGYA TGGMP IKTBW UEÑEN IEEU
la contraseña es : ABER
```

```
1.-DESCUBRIR CONTRASEÑA
2.-ESCRIBIR CONTRASEÑA
3.-DESIFRAR MENSAJE
0.-SALIR
2
Introduce la contraseña
ABER
Escoge una opccion
1.-DESCUBRIR CONTRASEÑA
2.-ESCRIBIR CONTRASEÑA
3.-DESIFRAR MENSAJE
0.-SALIR
3
Introduce el mensaje a descifrar
PBVRQ VICAD SKAÑS DETSJ PSIED BGGMP SLRPW RÑPWY EDSDE ÑDRDP CRCPQ MNPWK
UBZVS FNVRD MTIPW UEQVV CBOVN UEDIF QLONM WNUVR SEIKA ZYEAC EYEDS ETFPH
LBHGU ÑESOM EHLBX VAEEP UÑELI SEVEF WHUNM CLPQP MBRRN BPVIÑ MTIBV VEÑID
ANSJA MTJOK MDODS ELPWI UFOZM QMVNF OHASE SRJWR SFQCO TWVMB JGRPW VSUEX
INQRS JEUEM GGRBD GNNIL AGSJI DSVSU EEINT GRUEE TFGGM PORDF OGTSS TOSEQ
OÑTGR RYVLP WJIFW XOTGG RPQRR JSKET XRNBL ZETGG NEMUO TXJAT ORVJH RSFHV
NUEJI BCHAS EHEUE UOTIE FFGYA TGGMP IKTBW UEÑEN IEEU
mensaje en claro :PARAQ UELAC OSAZO MESOR PREND ACOMO OTROS AROSH ECOME DZADO YACON UNOSS
```

Se agarró el mensaje de las diapositivas que era M = “Para que la cosa no me sorprenda...”. ✌

Método Kasiski

```
run:
Ataque Kasiski
Ingrese el texto: LNUDVMUYRMUDVLLPXAFZUEFAIOVWVMUOVMUEVMUEZCUDVSYWCIVCFGUCUNYCGALLGRCYTIJTRNNPJQOPJEMZITYLIAYYKRYEFDUDCAMAVRMZEAMBLEXPJCCQIEHPJTYXVNMLAEZTIMUOFRUFC
La longitud del texto: 145
Texto ingresado:
LNUDV MUYRM UDVLL PXAFZ UEFAI OVWVM UOVMU EVMUE ZCUDV SYWCI VCFGU CUNYC GALLG RCYTI JTRNN PJQOP JEMZI TYLIA YYKRY EFDUD CAMAV RMZEA MBLEX PJCCQ IEHPJ TYXVN MLAEZ TIMUO FRUFC
Cadena: Separacion
UDV:    8
VMU:    24
UDV:    32
VMU:    4
MUO:    108
VMU:    4
MUE:    4
Maxico Comun Divisor: 4
El texto se dividio en 4 Subcriptogramas:
SubC0: LVRVXUIVVVZVCFUGGTRJJIIKFCVELJIJVAIFC
SubC1: NMMLAEOMMMCSIGNARINQETARDARAECETNEMR
SubC2: UUULFFVUUUUYVUYLCJNOMYYYUMMMXCHYMZUU
SubC3: DYDPZAWOEEDWCCCLYTPPZLYEDAZBPQPXLTOF
       A  B  C  D  E  F  G  H  I  J  K  L  M  N  Ñ  O  P  Q  R  S  T  U  V  W  X  Y  Z
SubC0  1  0  3  0  1  3  2  0  5  4  1  2  0  0  0  0  0  0  2  0  1  2  8  0  1  0  1
SubC1  5  0  2  1  5  0  1  0  2  0  0  1  6  4  0  1  0  1  4  1  2  0  0  0  0  0  0
SubC2  0  0  2  0  0  2  0  1  0  1  0  2  5  1  0  1  0  0  0  0  0  11 2  0  1  6  1
SubC3  2  1  3  4  3  1  0  0  0  0  0  0  3  0  0  0  2  5  1  0  0  2  0  0  2  1  3  3
Posibles claves:
GOJZ
VELLO
RAUL
BUILD SUCCESSFUL (total time: 9 seconds)
```

El programa pedirá que introduzcan el texto

```
run:
Ataque Kasiski
Ingrese el texto:
```

Yo dite un texto corto de 145 caracteres. Se puede notar el puntero en la imagen de abajo, solo basta con presionar enter para que se desgloce toda la informacion.

```
run:
Ataque Kasiski
Ingrese el texto: LNUDVMUYRMUDVLLPXAFZUEFAIOVWVMUOVMUEVMUEZCUDVSYWCIVCFGUCUNYCGALLGRCYTIJTRNNPJQOPJEMZITYLIAYYKRYEFDUDCAMAVRMZEAMBLEXPJCCQIEHPJTYXVNMLAEZTIMUOFRUFC
```

Este metodo es el que imprime “Ingrese el texto: ” y guarda en una variable String el texto, pasa por unos filtros para quitar los espacios o saltos de linea, me sireve para poder manipular mejor la informacion o el texto.

```
public String Cadena()
{
    String texto;
    System.out.print("Ingrese el texto: ");
    texto = sc.nextLine();
    texto = texto.replace( " ", "");
    texto = texto.replaceAll("\\s", "");
    texto = texto.trim();
    return texto;
}
```

Con este metodo obtengo la longitud del texto

```
La longitud del texto: 145
```

```
public int Longitud( String cadena )
{
    int longitud = cadena.length();
    return longitud;
}
```

En este metodo guardo el texto en una arreglo para separar cada letra y uso la longitud para saber el tamaño del arreglo que realizara

```
public void Texto( String texto[], String cadena, int longitud )
{
    for( int a = 0; a < longitud; a++ )
    {
        texto[a] = Character.toString( cadena.charAt(a));
    }
}
```

Imprime el texto que esta en el arreglo y se imprime un espacio cada 5 caracteres y un salto de linea cada 150 caracteres.
No se nota el salto de linea con este ejemplo ya que son 145 caracteres.

```
public void MostrarTexto( String texto[] )
{
    for( int a = 0; a < texto.length; a++ )
    {
        System.out.print( texto[a] );
        if( (a+1) % 5 == 0 )
        {
            System.out.print(" ");
            if( (a+1) % 150 == 0 )
            {
                System.out.println("");
            }
        }
    }
}
```

```
Texto ingresado:
LNUDV MUYRM UDVLL PXAFZ UEFAI OVWVM UOVMU EVMUE ZCUDV SYWCI VCFGU CUNYC GALLG RCYTI JTRNN PJQOP JEMZI TYLIA YYKRY EFDUD CAMAV RMZEA MBLEX PJCCQ IEHPJ TYXVN MLAEZ TIMUO FRUFC
```

Secuencias es un metodo que realiza la busqueda de palabras en este caso de tres cifras “A,B,C” y las compara con “D,E,F”

```
public void Secuencias( String texto[] )
{
    int a = 1, b = 2, c = 3;
    int d = 0, e = 0, f = 0;
    int g = 1;
    Secuencias0( texto, a, b, c, d, e, f, g );
}
```

Secuencias0 es un metodo recursivo que depende de Secuencias1, y Secuencias2 en este metodo si llega a encontrar una palabra por asi decirlo, que sea igual la guar lo guarda en un Arreglo de tipo Objeto.
Secuencias1 retorna un valor de tipo int
Secuencias2 retorna un valor de tipo String

```
public void Secuencias0( String texto[], int a, int b, int c, int d, int e, int f, int g )
{
    if ( c < texto.length )
    {
        d = b;
        e = c;
        f = e + 1;
        int h = Secuencias1( texto, a, b, c, d, e, f, g );
        String secuencia = Secuencias2( texto, a, b, c, d, e, f );
        if( h > 1 )
        {
            secuencias.add( new Secuencia( secuencia, h ) );
            Secuencias0( texto, a+1, b+1, c+1, d, e, f, g );
        }
        else
        {
            Secuencias0( texto, a+1, b+1, c+1, d, e, f, g );
        }
    }
    else
    {
    }
}
```

Secuencias1 busca la palabra ejemplo “U,D,V”, Cada letra esta uvicada en un numero ya que el texto tiene 145 letras ya sea que U = a la primer posicion ‘0’, D = la posicion 1 y V = 2.
U,D,V esta almacenada en la variable secuencia ?,?,? la siguiente variable esta almacena en secuencia0 y este va incrementado su posicion, hasta encontrar una posicion en la que las palbras coincidan, al coincidir las palabras realiza la resta de sus posicione para retornar la distancia entre estas.

```
public int Secuencias1( String texto[], int a, int b, int c, int d, int e, int f, int g )
{
    if( f < texto.length )
    {
        String secuencia = texto[a] + texto[b] + texto[c];
        String secuencia0 = texto[d] + texto[e] + texto[f];
        if( secuencia.equals(secuencia0) )
        {
            g = d - a;
        }
        else
        {
            return Secuencias1( texto, a, b, c, d+1, e+1, f+1, g );
        }
    }
    return g;
}
```

Secuencias2 hace lo mismo pero retorna la palabra que existe mas de una ves

```
public String Secuencias2( String texto[], int a, int b, int c, int d, int e, int f )
{
    String tmp = "";
    if( f < texto.length )
    {
        String secuencia = texto[a] + texto[b] + texto[c];
        String secuencia0 = texto[d] + texto[e] + texto[f];
        if( secuencia.equals(secuencia0) )
        {
            tmp = secuencia;
        }
        else
        {
            return Secuencias2( texto, a, b, c, d+1, e+1, f+1 );
        }
    }
    return tmp;
}
```

```
Cadena:  Separacion
UDV:     8
VMU:     24
UDV:     32
VMU:     4
MUO:     108
VMU:     4
MUE:     4
```

Muestra las secuencias, las secuencias se guardaron un objetos, por cada secuencia un objeto y los objetos se guardaron en un ArrayList..
El metodo manda a llamar a el maximo comun divisor

```
public void MostrarSecuencias()
{
    int distancia[] = new int[secuencias.size()];
    for( int i = 0; i < secuencias.size(); i++)
    {
        System.out.println( secuencias.get(i).obtenerSecuencia() + ": \t" + secuencias.get(i).obtenerDistancia() );
        distancia[i] = secuencias.get(i).obtenerDistancia();
    }
    mcd =MaximoComunDenominador( distancia );
    System.out.println( "Maxico Comun Divisor: "+ mcd );
}
```

La variable distancias de tipo arrglo tiene almacenado todas las distancias de aquellas palabras que se repetian y este metodo ocupa esas distancias para retornar el maximo comun divisor.

```
public int MaximoComunDenominador( int distancia[] )
{
    int numero[] = distancia;
    int minimo = numero[0];
    for( int i : numero )
    {
        if( i < minimo )
        {
            minimo = i;
        }
    }
    int mcd = 0;
    for( int i = 1; i <= minimo; i++ )
    {
        for( int j = 0; j < numero.length; j++ )
        {
            for( int k = 0; k < j; k++ )
            {
                if( numero[k]%i == 0 && numero[j]%i == 0 )
                {
                    mcd = i;
                }
            }
        }
    }
    return mcd;
}
```

En este metodo empezamos a realizar los SubCriptogramas, con la ayuda del MCD podemos saber cuantos SubCriptogramos necesitamos hacer. El metodo es recursivo haciendo una cadena String de los caracteres seleccionados, cuando termina de recorrer el texto, la cadena con los caracteres seleccionados, se guardan en un objeto y el objeto se agrega a un ArrayList

```
public void CrearSubCriptogramas( String texto[], int longitud  )
{
    int a, b;
    for( a = 0; a < mcd; a++ )
    {
        b = a;
        String cadena = "";
        String cripto = CrearSubCriptograma0( texto, cadena, a, b, longitud );
        subCripto.add( new SubCriptograma( cripto ) );
    }
}
public String CrearSubCriptograma0( String texto[], String cadena, int a, int b, int longitud )
{

    if ( b < longitud )
    {
        cadena += texto[b];
        return CrearSubCriptograma0( texto, cadena, a, b+4, longitud );
    }
    else
    {
        b = 0;
    }
    return cadena;
}
```

En este metodo imprimimos los SubCriptogramas que se realizaron

```
public void MostrarSubCriptogramas()
{
    for( int i = 0; i < subCripto.size(); i++ )
    {
        System.out.println( "SubC" + i + ": " + subCripto.get(i).obtenerSubCriptograma() );
    }
}
```

```
El texto se dividio en 4 Subcriptogramas:
SubC0: LVRVXUIVVVZVCFUGGTRJJIIKFCVELJIJVAIFC
SubC1: NMMLAEOMMMCSIGNARINQETARDARAECETNEMR
SubC2: UUULFFVUUUUYVUYLCJNOMYYYUMMMXCHYMZUU
SubC3: DYDPZAWOEEDWCCCLYTPPZLYEDAZBPQPXLTOF
```

En este metodo estamos pasando el SubCriptograma a un arreglo para separa las letras para manipular mejor la frecuencia de cada una de las letras,

```
public void ArregloCriptogramas()
{
    for( int j = 0; j < subCripto.size(); j++ )
    {
        String cadena = subCripto.get(j).obtenerSubCriptograma();
        int longitud = Longitud( cadena );
        String texto[] = new String[ longitud ];
        for( int i = 0; i < longitud; i++ )
        {
            texto[i] = Character.toString( cadena.charAt(i) );
        }
        letra.add( new Letra( texto ));
    }
}
```

con el metodo de frecuencia buscamos la existencia de cada una de las letras del alfabeto contado con un arreglo de 27 letras. Las frecuencias las guarda en un objeto de tipo ArrayList.

```
public void FreciuenciaCriptograma()
{
    for( int i = 0; i < mcd; i ++ )
    {
        String letras[] = letra.get(i).obtenerLetra();
        int frecuencia[] = new int[27];
        for( int j = 0; j < abc.length; j++ )
        {
            int contador = 0;
            for( int k = 0; k < letras.length; k++ )
            {
                String a = abc[j];
                String b = letras[k];
                if( a.equals(b) )
                {
                    contador += 1;
                }
            }
            frecuencia[j] = contador;
        }
        frecuencias.add( new Frecuencia( frecuencia ) );
    }
}
```

Se muestra la frecuencia de cada letra

```
public void MostrarFrecuencia()
{
    System.out.print("\t");
    for( int i = 0; i < abc.length; i++ )
    {
        System.out.print( abc[i] + "  " );
    }
    System.out.print( "\n" );
    for( int i = 0; i < frecuencias.size(); i++ )
    {
        System.out.print( "SubC" + i + "\t" );
        int frecuencia[] = frecuencias.get(i).obtenerFrecuencia();
        for( int j = 0; j < frecuencia.length; j++ )
        {
            System.out.print( frecuencia[j] + "  " );
        }
        System.out.print("\n");
    }
}
```

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
SubC0	1	0	3	0	1	3	2	0	5	4	1	2	0	0	0	0	0	0	2	0	1	2	8	0	1	0	1
SubC1	5	0	2	1	5	0	1	0	2	0	0	1	6	4	0	1	0	1	4	1	2	0	0	0	0	0	0
SubC2	0	0	2	0	0	2	0	1	0	1	0	2	5	1	0	1	0	0	0	0	0	11	2	0	1	6	1
SubC3	2	1	3	4	3	1	0	0	0	0	0	3	0	0	0	2	5	1	0	0	2	0	0	2	1	3	3

Las posibles claves son obtenidas de un algoritmo para el lenguaje español, la frecuencia para el idioma español es A, E y O

Carácter	A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
Posición	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

La A es la posicion 0, + 4 posiciones esta la E, + 11 posiciones esta la O
En el Pimer SubCriptograma escojemos las que cubran la mayor frecuencia como RVG(2, 8, 2)
Siendo R la posicion 0 + 4 esta la letra V + 11 esta la letra G teniendo una frecuencia de 2, 8, 2 y esto se realiza en los demas Subcriptogramas quedando de esta manera;
SubC0 = RVG(2, 8, 2)
SubC1 = AEO(5, 5, 1)
SubC2 = UYJ(11, 6, 1)
SubC3 = LOZ(3, 2, 3)
Con eso tenemos las

```
Posibles claves:
GOJZ
VELLO
RAUL
```