INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

CRIPTOGRAFÍA

Sesión de Laboratorio 2

Hill Cipher

Grupo: 3CV2

Alumnos:

Martínez Galindo Angélica

Montaño Castañeda Daniel

Maestra:

Días Santiago Sandra

Contenido

Objetivo

En esta sesión vamos a trabajar con el cifrado de Hill. Además, vamos a hacer un ataque conociendo el texto plano para el cifrado de Hill.

Teoría

1. Breve descripción del cifrado de Hill.

Fue inventado por Lester S. Hill en 1929. A diferencia de los demás es extensible para trabajar en diferentes bloques de tamaño de las letras. Así que técnicamente es un cifrado de sustitución poligráfico, ya que puede funcionar en dígrafos, trigrafos bloques de letras o teóricamente cualquier tamaño de bloque. [1]

2. Definición de ataque conociendo el texto plano.

El atacante conoce o puede adivinar el texto de alguna parte del texto cifrado. La tarea es desencriptar el resto del bloque cifrado utilizando esta información. Esto puede ser hecho determinando la clave utilizada para encriptar la información, o a través de algún atajo. Uno de los mejores ataques modernos de texto plano conocido es el "criptoanálisis lineal" contra cifradores de bloques.[2]

3. Recursos del adversario cuando se realiza un ataque teniendo el texto plano correspondiente.

En este caso, si también conociera el tamaño de las matrices usadas, sería muy simple la obtención de la llave, en caso de que no conozca el tamaño de las matrices usadas seria solo un poco más difícil, ya que solo tendría que probar con todos los posibles tamaños, lo cual no sería tan difícil.[3]

4. ¿Es posible hacer un ataque de solo texto cifrado a un cifrado de Hill? ¿Cómo se podría hacer?

Un ataque de solo texto cifrado se puede hacer si se conoce el tamaño de las matrices usadas generando todas las posibles matrices llave, en dado caso de que este dato no se conozca sería más complicado, pero aun así se podría realizar variando el tamaño de las matrices, aunque esto involucraría un poco más de tiempo ya que si las matrices usadas son muy grandes la complejidad del problema incrementaría exponencialmente. [4]

[1] Daniel Rodríguez. http://crypto.interactive-maths.com/hill-cipher.html.
[2] Cristian Borghello. http://www.segu-info.com.ar/proyectos/p1_ataques.htm
[3] y [4] Esta parte fue contestada según lo visto en clase

Desarrollo

Ejercicios de programación

Programa 1

1. Diseño de un programa en C / C ++ para cifrar y descifrar utilizando el cifrado de Hill. El programa debe ofrecer las siguientes opciones.

a) Generación de claves. El usuario debe poder escoger entre m=2 y m=3. El programa debe generar una matriz al azar que pueda ser usada como llave. Esta llave debe ser guardada en un archivo de texto y el nombre de dicho archivo puede ser escogido por el usuario.

b) Cifre. El programa debe cifrar un archivo de texto de cualquier tamaño, el usuario debe dar el nombre del archivo del texto plano, de la llave y del texto cifrado.

c) Descifre. El programa debe descifrar un archivo de texto de cualquier tamaño, el usuario debe dar el nombre del archivo del texto plano, de la llave y del texto cifrado.

Código 1

practica2_1.c

```
/*
    Practica 2 Ejercicio 1

    Integrantes:
        Martínez Galindo Angélica
        Montaño Castañeda Daniel
    Grupo:
        3CV2
    Fecha:
        29 de Febrero del 2016
    Compilación:
        (Windows) gcc practica2_1.c practica2.c -o practica2_1
    Ejecución:
        (Windows) practica2_1

*/

#include "practica2.h"

int main()
{
    int opcion,salir=0,m;
    char texto_plano[20];
    char texto_cifrado[20];
    char texto_llaves[20];
```

```
    char texto_descifrado[20];

    while(salir == 0) //mientras el usuario no seleccione la opcion de
salir se mostrara el siguiente menú
    {
        printf("\n Eliga una de las siguientes opciones: ");
        printf("\n  1) Generar de llaves");
        printf("\n  2) Cifrar");
        printf("\n  3) Descifrar");
        printf("\n  4) Salir \n");
        scanf("%d",&opcion);

        if(opcion == 1)
        {
            printf("\n Escriba la dimension de la matriz (2 o 3) \n");
            scanf("%d",&m);
            if(m==2 || m==3)
                generarLlave(m); //generamos una llave de numeros
aleatorios de la dimension "m"
            else
            {
                printf("\n Solo de permiten matrices de 2 o 3 dimensiones
\n");
                exit(0);
            }

        }
        else if(opcion==2)
        {
            printf("\n Escriba el archivo que contiene el texto a cifrar:
");
            scanf("%s",texto_plano);
            printf("\n Escriba el archivo que contiene la llave: ");
            scanf("%s",texto_llaves);
            printf("\n Escriba el archivo donde se guardara el texto
cifrado: ");
            scanf("%s",texto_cifrado);

            cifrar(texto_plano,texto_llaves,texto_cifrado);
        }
        else if(opcion==3)
        {
            printf("\n Escriba el archivo que contiene el texto a
descifrar: ");
            scanf("%s",texto_plano);
            printf("\n Escriba el archivo que contiene la llave: ");
            scanf("%s",texto_llaves);
            printf("\n Escriba el archivo donde se guardara el texto
descifrado: ");
            scanf("%s",texto_descifrado);

            descifrar(texto_plano,texto_llaves,texto_descifrado);
        }
        else if(opcion==4)
        {
            salir++;
        }
```

```
        else
        {
            printf("\n Elija una de las opciones validas \n");
            exit(0);
        }

    }


    return 0;

}
```

practica2.c

```
/*
    Definicion de las funciones utilizadas
*/
#include "practica2.h"


int gcd(int a, int b) //calcula el maximo comun divisor de dos numeros
{
    if (a == 0)
        return b;
    return gcd(b%a, a);
}

int inverso(int x,int m) //calcula el inverso de un numero
{
    int i;
    for(i=1; i<m; i++)
    {
        if( (x*i)%m ==1)
            return i;
    }
}

//creamos una matriz de dimensiones filas X columnas
matriz crearMatriz(int filas, int columnas)
{
    int i,j;
    matriz m;
    m = (int **)malloc(filas*sizeof(int*)); //creamos el espacio en
memoria

    for (i=0;i<filas;i++)
        m[i] = (int*)malloc(columnas*sizeof(int));

    for(i=0; i<filas; i++)  //incializamos todos los elementos de la
matriz con 0
    {
        for(j=0; j<columnas; j++)
            m[i][j] = 0;
    }
```

```
    return m;
}

//mandamos a imprimir la matriz
void imprimeMatriz(matriz m, int filas, int columnas)
{
    int i,j;
    for(i=0; i<filas; i++)
    {
        for(j=0; j<columnas; j++)
        {
            printf("%d ",m[i][j]);
        }
        printf("\n");
    }
}

//Multiplicacion de dos matrices dadas con sus dimensiones
correspondientes
matriz multiplicarMatrices(matriz a, int filas_a, int columnas_a, matriz
b, int filas_b, int columnas_b)
{
    int i,j,k;
    matriz m = crearMatriz(filas_a,columnas_b); //matriz resultante

    for(i=0; i<filas_a; i++)
    {
        for(j=0; j<columnas_b; j++)
        {
            for(k=0; k< filas_b; k++)
                m[i][j]= m[i][j]+a[i][k]*b[k][j];
        }
    }
    return m;
}

//Multiplicacion de una matricez con un escalar
matriz multiplicarMatrizConEscalar(int esc, matriz m, int filas, int
columnas)
{
    int i,j;
    matriz Mult = crearMatriz(filas,columnas);  //matriz resultante
    for(i=0; i<filas; i++)
    {
        for(j=0; j<columnas; j++)
            Mult[i][j] = m[i][j]*esc;   //multiplicamos todos los
elementos de la matriz por el escalar
    }

    return Mult;

}

//Transpuesta de una matriz
matriz matrizTranspuesta (matriz a, int filas, int columnas)
{
    int i,j;
```

```
    matriz m = crearMatriz(columnas,filas); //matriz resultante

    for (i = 0; i < filas; i++)
    {
        for(j = 0 ; j < columnas ; j++ )
            m[j][i] = a[i][j];  //volteamos los elementos de la matriz
    }
    return m;

}

//Adjunta de una matriz
matriz matrizAdjunta(matriz a,int dim)
{
    matriz adjunta = crearMatriz(dim,dim);  //matriz resultante

    if(dim == 2)     //Para cuando la matriz solo es de 2 dimensiones
    {
        adjunta[0][0] = a[1][1];
        adjunta[0][1] = -a[0][1];
        adjunta[1][0] = -a[1][0];
        adjunta[1][1] = a[0][0];
    }
    else if(dim == 3)//Para cuando la matriz es de 3 dimensiones
    {
        adjunta[0][0] = det(a[1][1],a[1][2],a[2][1],a[2][2]);
        adjunta[0][1] = -det(a[1][0],a[1][2],a[2][0],a[2][2]);
        adjunta[0][2] = det(a[1][0],a[1][1],a[2][0],a[2][1]);

        adjunta[1][0] = -det(a[0][1],a[0][2],a[2][1],a[2][2]);
        adjunta[1][1] = det(a[0][0],a[0][2],a[2][0],a[2][2]);
        adjunta[1][2] = -det(a[0][0],a[0][1],a[2][0],a[2][1]);

        adjunta[2][0] = det(a[0][1],a[0][2],a[1][1],a[1][2]);
        adjunta[2][1] = -det(a[0][0],a[0][2],a[1][0],a[1][2]);
        adjunta[2][2] = det(a[0][0],a[0][1],a[1][0],a[1][1]);
    }

    return adjunta;
}

//Sacamos modulo 26 a ada uno de los elementos de la matriz
matriz matrizModulo(matriz a, int filas, int columnas, int m)
{
    matriz mod = crearMatriz(filas,columnas);
    int i,j;

    for (i = 0; i < filas; i++)
    {
        for(j = 0 ; j < columnas ; j++ )
        {
            mod[i][j] = a[i][j] % m;
            if(mod[i][j] < 0) //en caso que tengamos un numero negativo
            {
                while(mod[i][j]<0)//si tenemos algun numero negativo le
sumamos m hasta que se vuelva positivo
                    mod[i][j] = mod[i][j]+m;
```

```
            }
        }
    }

    return mod;
}

//Calculamos la inversa de una matriz dada
matriz matrizInversa(matriz m,int dim,int determinante)
{
    int i,j;
    matriz inversa = crearMatriz(dim,dim);
    matriz adj = matrizAdjunta(m,dim);
    matriz trans =  matrizTranspuesta (adj,dim,dim);
    int inverso_det = inverso(determinante,26); //sacamos el inverso del
determinante
    if(dim>2)
        inversa =
multiplicarMatrizConEscalar(inverso_det,trans,dim,dim);//inversa de una
matriz 2x2
    else
        inversa =
multiplicarMatrizConEscalar(inverso_det,adj,dim,dim);//inversa de una
matriz 3x3

    inversa = matrizModulo(inversa,dim,dim,26);

    return inversa;
}

//Determinante de elementos concretos de una matriz
int det(int a, int b, int c, int d)
{
    return (a*d-b*c);
}

//Determinante de una matriz de dimensiones cualesquiera
int determintanteMatriz(matriz a,int n)
{
    int det=0, p, h, k, i, j;
    matriz temp = crearMatriz(n,n);//creamos una matriz temporal

    if(n==1)
        return a[0][0]; //determinante de una matriz 1x1
    else if(n==2)
        det=(a[0][0]*a[1][1]-a[0][1]*a[1][0]); //determinante de una
matriz 2x2
    else
    {
        for(p=0;p<n;p++)        //determinante de una matriz mayor de 2
        {
            h = 0;
            k = 0;
            for(i=1;i<n;i++)
            {
                for( j=0;j<n;j++)
                {
```

```
                    if(j==p)
                     continue;

                    temp[h][k] = a[i][j];
                    k++;
                    if(k==n-1)
                    {
                        h++;
                        k = 0;
                    }
                }
            }
            det=det+a[0][p]*pow(-1,p)*determintanteMatriz(temp,n-1);
//funcion recursiva
        }
  }

    free(temp);
    return det;
}


//Funcion de cifrado
void cifrar(char* texto_plano,char* texto_llaves,char* texto_cifrado)
{

    FILE *entrada;
    FILE *llave;
    char ch;
    int caracter,i = 0,j=0;
    int dim,det,m=26,bandera = 0;
    matriz K,P,C,CM;


    entrada = fopen(texto_plano,"r+");
    llave = fopen(texto_llaves,"r+");

    dim = numeroDimensiones(texto_llaves);
    K = leerMatriz(texto_llaves);
    det = determintanteMatriz(K,dim);
    printf("\nK = \n");
    imprimeMatriz(K,dim,dim);

    if(det < 0)
    {
        while(det<0)
            det = det+m;
    }


    if(entrada == NULL || llave == NULL)
    {
        printf("ERROR en apertura de archivos\n");
        exit(0);
    }
    else
    {
```

```
        P = crearMatriz(dim,1);
        while(feof(entrada) == 0) //mientras haya elementos de nuestro
texto a cifrar
        {
            if(j<dim)
            {
                P[j][0] = fgetc(entrada)-65;    //vamos metiendo los
caracteres en sus equivalentes numeros a la matriz
                j++;

            }
            else
            {
                printf("\nP = \n");
                imprimeMatriz(P,dim,1);
                C = multiplicarMatrices(K,dim,dim,P,dim,1);
//multiplicacion K*P
                printf("\nC = \n");
                imprimeMatriz(C,dim,1);
                CM = matrizModulo(C,dim,1,m);//sacamos el modulo de (K*P)
mod 26
                printf("\nCM = \n");
                imprimeMatriz(CM,dim,1);
                escribirMatriz(texto_cifrado,CM,dim,1); //escribimos en
el archivo
                P = crearMatriz(dim,1);
                j=0;
            }
        }

    }

    free(K);    //liberamos de la memoria
    free(C);
    free(P);

    fclose(entrada);
    fclose(llave);

}

//Funcion de cifrado
void descifrar(char* texto_cifrado,char* texto_llaves,char*
texto_descifrado)
{
    FILE *entrada;
    FILE *llave;
    char ch;
    int caracter,i = 0,j=0;
    int dim,det,m = 26,bandera = 0;
    matriz K,KI,P,C,CM;

    entrada = fopen(texto_cifrado,"r+");
    llave = fopen(texto_llaves,"r+");

    dim = numeroDimensiones(texto_llaves);
    K = leerMatriz(texto_llaves);
```

```
    det = determintanteMatriz(K,dim)%m;

    if(det < 0)
    {
        while(det<0)
            det = det+m;
    }

    KI = matrizInversa(K,dim,det);
    printf("\ninversa de K = \n");
    imprimeMatriz(KI,dim,dim);

    if(entrada == NULL || llave == NULL)
    {
        printf("ERROR\n");
        exit(0);
    }
    else
    {
        P = crearMatriz(dim,1);
        while(feof(entrada) == 0)
        {
            if(j<dim)
            {
                P[j][0] = fgetc(entrada)-65;
                j++;

            }
            else
            {
                printf("\nP = \n");
                imprimeMatriz(P,dim,1);
                C = multiplicarMatrices(KI,dim,dim,P,dim,1);
                printf("\nC = \n");
                imprimeMatriz(C,dim,1);
                CM = matrizModulo(C,dim,1,m);
                printf("\nCM = \n");
                imprimeMatriz(CM,dim,1);
                escribirMatriz(texto_descifrado,CM,dim,1);
                P = crearMatriz(dim,1);
                j=0;
            }
        }

    }

    free(K);//liberamos de la memoria
    free(C);
    free(P);

    fclose(entrada);
    fclose(llave);


}

//Generamos una llave aleatoria de dimensiones m X m dada por el usuario
```

```
void generarLlave(int dimension)
{
    matriz M;
    M = crearMatriz(dimension,dimension);
    FILE *llave;
    llave = fopen("llave.txt","w+"); //dicha llave se guardara en el
archivo de texto llave.txt
    int i,j,det,bandera = 0;
    srand (time(NULL));

    while(bandera == 0)
    {
        for(i=0; i < dimension; i++)
        {
            for(j=0; j < dimension; j++)
                M[i][j]= rand()%260;    //generamos numeros aleatorios
        }

        det = determintanteMatriz(M,dimension);
        //validacion de llave
        if(gcd(det,26)==1 && det%26 !=0) //el determinante de la matriz
debe de ser diferente de 0 y el mcd del determinante con 26 debe ser 1
        {
            for(i=0; i < dimension; i++)
            {
                for(j=0; j < dimension; j++)
                    fprintf(llave,"%d\n",M[i][j]);  //escribimos en el
archivo
            }

            bandera++;
        }
    }

    fclose(llave);

}

//Mandamos a escribir los elementos de la matriz a un archivo de texto
void escribirMatriz(char* nombre_arch,matriz m,int filas,int columnas)
{
    FILE *arch;
    arch = fopen(nombre_arch,"a+");
    int i,j;

    if(arch == NULL)
        printf("\n Error");
    else
    {
        for(i=0; i < filas; i++)
        {
            for(j=0; j < columnas; j++)
            {
                fprintf(arch,"%c",65+m[i][j]);  //escribimos en el
archivo
                printf("%c",65+m[i][j]);
            }
```

```
        }
    }

    fclose(arch);
}

//Leemos de un archivo de texto los elementos de una matriz y devolvemos
dicha matriz
matriz leerMatriz(char* archivo)
{
    FILE *arch;
    int dim,j,i,num;
    arch = fopen(archivo,"r");

    dim = numeroDimensiones(archivo);
    matriz K = crearMatriz(dim,dim);

    if(arch == NULL)
        printf("\n Error");
    else
    {
        for(i=0; i < dim; i++)
        {
            for(j=0; j < dim; j++)
            {
                fscanf(arch,"%d",&num); //leemos y guardamos en num
                K[i][j] = num;
            }
        }
    }
    fclose(arch);

    return K;

}

//Devolvemos el numero de dimensiones basandonos en el numero de
elementos que contenga el archivo
int numeroDimensiones(char* nombre_arch)
{
    FILE *arch;
    arch = fopen(nombre_arch,"r");
    int dim=0;
    char ch;

    if(arch == NULL)
        printf("\n Error");
    else
    {
        while ((ch = fgetc(arch)) != EOF)
        {
            if(ch == '\n')
                dim++;
        }
        dim++;
    }
```

```
    fclose(arch);

    return sqrt(dim);   //devolvemos la raiz cuadrada de el numero de
lineas que se haya leido
}
```

practica2.h

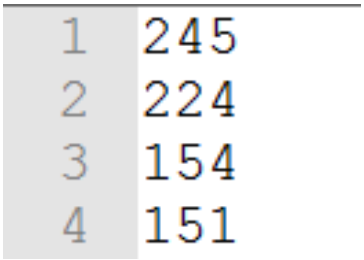
```
/*
    Declaramos las funciones a utilizar
*/

#ifndef PRACTICA2_H
#define PRACTICA2_H

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

typedef int** matriz;

matriz crearMatriz(int filas,int columnas);
void imprimeMatriz(matriz m, int filas, int columnas);
matriz multiplicarMatrices(matriz a, int filas_a, int columnas_a, matriz
b, int filas_b, int columnas_b);
int determintanteMatriz(matriz a,int n);
matriz matrizTranspuesta (matriz m, int filas, int columnas);
matriz matrizAdjunta(matriz a, int dim);
int determinante(int a, int b, int c, int d);
matriz leerMatriz(char* archivo);
void escribirMatriz(char* nombre_arch,matriz m,int filas,int columnas);
matriz matrizModulo(matriz a, int filas, int columnas, int m);
matriz matrizInversa(matriz m,int dim,int determinante);
matriz multiplicarMatrizConEscalar(int esc, matriz m, int filas, int
columnas);
int gcd(int a, int b);
int inverso(int x,int m);
void cifrar(char* texto_plano,char* texto_llaves,char* texto_cifrado);
void descifrar(char* texto_plano,char* texto_llaves,char*
texto_descifrado);
void generarLlave(int dimension);
int numeroDimensiones(char* nombre_arch);


#endif
```

Pruebas

```
C:\Users\Angelica\Desktop\CRIPTO\Practica 2\practica2>gcc practica2_1.c practica2.c -o practica2_1

C:\Users\Angelica\Desktop\CRIPTO\Practica 2\practica2>practica2_1
```

Lo primero que aparecerá es un pequeño menú. Si el usuario selecciona la opción 1 se generará una llave K de dimensiones 2x2 o 3x3, el usuario decidirá. Teniendo en cuenta que mcd (det(K),26) tiene que ser igual a 1.

```
 Eliga una de las siguientes opciones:
  1) Generar de llaves
  2) Cifrar
  3) Descifrar
  4) Salir
1

 Escriba la dimension de la matriz (2 o 3)
2
```



Estos son los elementos de nuestra matriz 2x2 que se generó con números aleatorios, utilizando el formato de las matrices se verían así:

$$K = \begin{bmatrix} 245 & 224 \\ 154 & 151 \end{bmatrix}$$

Para esto utilizamos la función:

```
matriz leerMatriz(char* archivo);
```

Dicha llave K, será guardara en el archivo llave.txt

Cabe mencionar que la generación de las llaves se realizará "n" veces hasta que se generé una que cumpla con las siguientes condiciones:

1. El determinante de la matriz clave privada K, tiene que ser distinto de cero.
2. El máximo común divisor entre el determinante de la matriz clave privada K, y 26 tiene que ser igual a 1.

Podemos verificar que la llave que se muestra anteriormente cumple con ambas condiciones.

- Para nuestra condición 1
  Det(K) = $\begin{vmatrix} 245 & 224 \\ 154 & 151 \end{vmatrix}$ = 245*151 - 154*224 = 36995 - 34496 = 2499 mod 26 = 3

- Para nuestra condición 2
  Mcd(3,26) = 1

Es el mismo procedimiento para el caso en el que el usuario quiera una matriz 3x3.

Ahora bien, si el usuario desea cifrar ya un texto en claro que tenga en mente deberá seleccionar la opción 2 de nuestro menú principal, posteriormente se le solicitarán tres archivos, el primero tendrá el texto a cifrar, el segundo la llave que se utilizará y el tercero es donde se guardara el texto ya cifrado.
Para cifrar tenemos los siguientes componentes para este ejemplo:

- K = matriz llave clave privada de dimensiones 3x3
- P = matriz que va guardando parcialmente los caracteres del texto a cifrar de dimensiones 3x1
- C = matriz que guarda el resultado de la multiplicación de K*P de dimensiones 3x1
- CM = matriz que guarda el módulo 26 de la matriz C, que serán ya los elementos cifrados, de dimensiones 3x1

```
Elija una de las siguientes opciones:
 1) Generar de llaves
 2) Cifrar
 3) Descifrar
 4) Salir
2

Escriba el archivo que contiene el texto a cifrar: entrada.txt

Escriba el archivo que contiene la llave: llave.txt

Escriba el archivo donde se guardara el texto cifrado: texto_cifrado.txt
```

entrada.txt

```
1 CRIPTO
```

C	R	I	P	T	O
2	17	8	15	19	14

No es necesario, pero lo mandamos a imprimir solo para que se visualice que realmente el programa hace lo que tiene que hacer.

```
K =
11 73 32
215 144 53
228 150 167

P =
2
17
8

C =
1519
3302
4342

CM =
11
0
0
LAA
```

Imprimimos nuestra llave con el formato al que estamos acostumbrados de una matriz 3x3

Para nuestra primera iteración P = $\begin{bmatrix} 2 \\ 17 \\ 8 \end{bmatrix}$

$$C = K*P = \begin{bmatrix} 11 & 73 & 32 \\ 215 & 144 & 53 \\ 228 & 150 & 167 \end{bmatrix} * \begin{bmatrix} 2 \\ 17 \\ 8 \end{bmatrix}$$

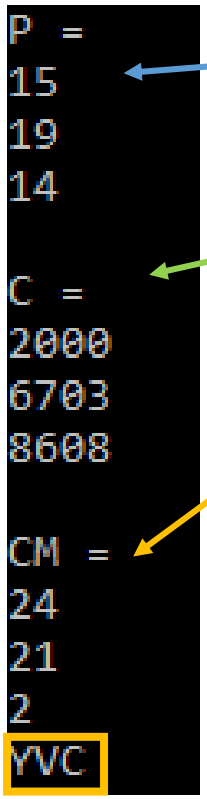
$$= \begin{bmatrix} 11*2+73*17+32*8 \\ 215*2+144*17+53*8 \\ 228*2+150*17+167*8 \end{bmatrix}$$

$$= \begin{bmatrix} 1519 \\ 3302 \\ 4342 \end{bmatrix}$$

$$CM = C \bmod 26 = \begin{bmatrix} 1519 \\ 3302 \\ 4342 \end{bmatrix} mod\ 26 = \begin{bmatrix} 11 \\ 0 \\ 0 \end{bmatrix}$$

Su equivalente en caracteres es:

11	0	0
L	A	A



Para nuestra segunda iteración P = $\begin{bmatrix}15\\19\\14\end{bmatrix}$

C = K*P = $\begin{bmatrix}11 & 73 & 32\\215 & 144 & 53\\228 & 150 & 167\end{bmatrix}$ * $\begin{bmatrix}15\\19\\14\end{bmatrix}$ = $\begin{bmatrix}2000\\6703\\8608\end{bmatrix}$

CM = C mod 26 = $\begin{bmatrix}2000\\6703\\8608\end{bmatrix}$ $mod\ 26 = \begin{bmatrix}24\\21\\2\end{bmatrix}$

Su equivalente en caracteres es:

24	21	2
Y	V	C

llave.txt

```
Eliga una de las siguientes opciones:
  1) Generar de llaves
  2) Cifrar
  3) Descifrar
  4) Salir
1

 Escriba la dimension de la matriz (2 o 3)
3
```

```
1  11
2  73
3  32
4  215
5  144
6  53
7  228
8  150
9  167
```

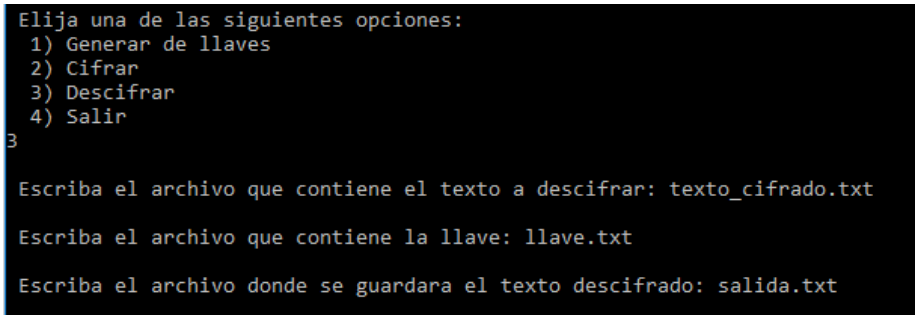
K = $\begin{bmatrix}11 & 73 & 32\\215 & 144 & 53\\228 & 150 & 167\end{bmatrix}$

texto_cifrado.txt

```
1  LAAYVC
```

11	0	0	24	22	2
L	A	A	Y	V	C

Si el usuario selecciona de nuestro menú principal la opción 3, se descifrara un texto. Se le solicitará ingresar el nombre del archivo que contiene el texto cifrado, el nombre del archivo que contiene la llave que se utilizó para el cifrado y finalmente el nombre del archivo en el que se desea guardar el texto descifrado.



Para este caso es el mismo procedimiento que en la encriptación como se mostró anteriormente, sin embargo, en este caso las multiplicaciones ser harán con K^{-1}. Para ello recordaremos como se saca la inversa de una matriz mediante la siguiente ecuación.

$$A^{-1} = \frac{1}{det\,(A)}\ adj\,(A)$$

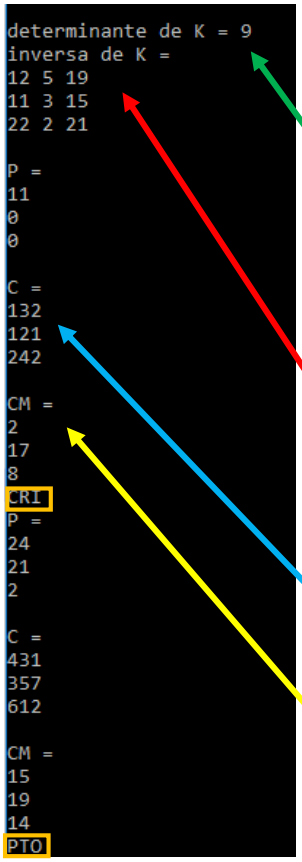
- Inversa de una matriz 3x3
  - Los pasos a seguir son:
    1. Encontrar determinante de la matriz 3 X 3
    2. Encontrar menor
    3. Encontrar Cofactor
    4. Encontrar adjunto
    5. Reemplazar resultados por debajo de la fórmula
- Inversa de una matriz 2x2

$$A^{-1} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{ad - bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

texto_cifrado.txt

1 LAAYVC

11	0	0	24	22	2
L	A	A	Y	V	C



Imprimimos la inversa de K pero podemos verificarla:

$$K = \begin{bmatrix} 11 & 73 & 32 \\ 215 & 144 & 53 \\ 228 & 150 & 167 \end{bmatrix}$$

$$\text{Det}(K) = |K| = \begin{vmatrix} 11 & 73 & 32 \\ 215 & 144 & 53 \\ 228 & 150 & 167 \end{vmatrix} =$$

11*144*167+73*53*228+32*215*150 – 32*144*228-11*53*150-73*215*167 = 264528+882132+1032000-1050624-87450-2621065 =2178660-3759139 = -1580479 mod 26 = 9

$$K^{-1} = \frac{1}{\det(K)} * (Adj(K))^T$$

$$C = K^{-1} * P = \begin{bmatrix} 12 & 5 & 19 \\ 11 & 3 & 15 \\ 22 & 2 & 21 \end{bmatrix} * \begin{bmatrix} 11 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 132 \\ 121 \\ 242 \end{bmatrix}$$

$$CM = C \bmod 26 = \begin{bmatrix} 132 \\ 121 \\ 242 \end{bmatrix} \quad mod\ 26 = \begin{bmatrix} 2 \\ 17 \\ 8 \end{bmatrix}$$

llave.txt

$$K = \begin{bmatrix} 11 & 73 & 32 \\ 215 & 144 & 53 \\ 228 & 150 & 167 \end{bmatrix}$$

salida.txt

1 CRIPTO

C	R	I	P	T	O
2	17	8	15	19	14

Programa 2

2. Diseñe un programa que realice un ataque al cifrado de Hill conociendo un texto cifrado y su respectivo texto plano considerando llaves de matrices de 2X2.

Código 2

```
#include "practica2.h"

int main()
{
    FILE *texto;//archivo de entrada
    FILE *texto_c;//archivo de entrada
    char texto_plano[20];
    char texto_cifrado[20];
    int m=26;

    printf("\n Escriba el archivo que contiene el texto a cifrar: ");
    scanf("%s",texto_plano);
    printf("\n Escriba el archivo donde se guardara el texto cifrado: ");
    scanf("%s",texto_cifrado);

    texto = fopen(texto_plano,"r");
    texto_c = fopen(texto_cifrado,"r");

    if (texto == NULL)
        printf("\nError de apertura del archivo. \n\n");
    if (texto_c == NULL)
        printf("\nError de apertura del archivo. \n\n");
    else
    {
        matriz mensaje=crearMatriz(2,2);
        matriz cipherText=crearMatriz(2,2);

        int i,j;
        //Ciclos para el llenado de la matriz mensaje
        for(i=0; i<2; i++)
```

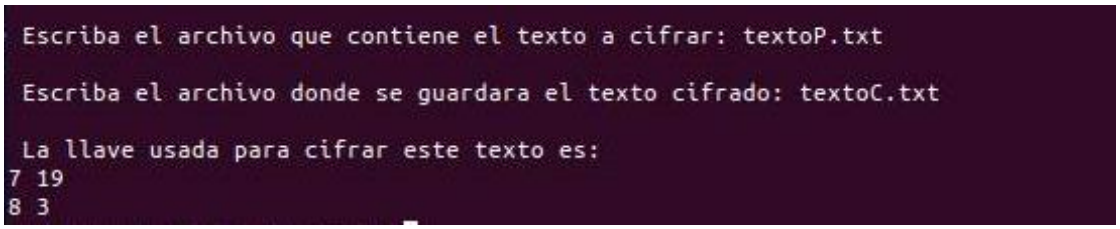
```
        {
            for(j=0; j<2; j++)
            {
                mensaje[i][j]=(int)getc(texto)-65;
            }
        }
        //Ciclos para el llenado de la matriz cifrada
        for(i=0; i<2; i++)
        {
            for(j=0; j<2; j++)
            {
                cipherText[i][j]=(int)getc(texto_c)-65;
            }
        }
        mensaje=matrizAdjunta(mensaje,2);
        mensaje=matrizTranspuesta(mensaje,2,2);
        int det=determintanteMatriz(mensaje,2)%m;

        if(det<0)
            det=det+m;

        int det_i=inverso(det,m);
        mensaje=multiplicarMatrizConEscalar(det_i,mensaje,2,2);
        mensaje=matrizModulo(mensaje,2,2,m);
        matriz K=multiplicarMatrices(mensaje,2,2,cipherText,2,2);
        K=matrizModulo(K,2,2,m);
        printf("\n La llave usada para cifrar este texto es: \n");
        imprimeMatriz(K,2,2);
    }

    return 0;
}
```

Pruebas

Para compilar y ejecutar:

```
mws@mws-HP-Mini-110-3100:~$ gcc ./practica2_2.c -o practica2_2
mws@mws-HP-Mini-110-3100:~$ ./practica2_2
```

Solo para aclarar se utilizaron las mismos archivos que en el programa anterior, de los cuales jalamos las funciones utilizadas en el código.
Siguiendo el ejemplo hecho en clase, los archivos nombrados contienen:
texto_P.txt = FRIDAY (5, 17, 8, 3, 0, 24)
texto_C.txt = PQCFKU (15, 16, 2, 5, 10, 20)



Y como podemos ver, se obtiene la matriz llave usada para cifrar dicho texto.

7	19
8	**3**