Carlos Li Hu
David López Ramos

Práctica 1: Criptografía Clásica

Introducción **2**

1. Sustitución Monoalfabeto **2**
Método afín (USO OBLIGATORIO DE GMP) 2
Aplicación Práctica (USO OBLIGATORIO DE GMP) 3

2. Sustitución Polialfabeto **4**
Método de Hill (USO DE GMP NO OBLIGATORIO) 4
Método de Vigenere (USO DE GMP NO OBLIGATORIO) 5
Criptoanálisis del Vigenere (GMP NO OBLIGATORIO) 6

3. Método de transposición (GMP NO OBLIGATORIO) **12**

4. Producto de criptosistemas permutación **13**

Introducción

En todos los apartados, para interpretar los flags y los argumentos pasados por terminal, hemos hecho uso de la función getopt, y hemos ido procesando los diferentes argumentos para su posterior uso.
En caso de que no se introduzcan argumentos, se mostraría por pantalla el formato que tiene que tener el comando para ser ejecutado correctamente, además de un ejemplo para entenderlo correctamente.

Sobre el uso de ficheros, cabe destacar que si no se especifica fichero de entrada, se crea un fichero *teclado.txt* donde se escribe el texto que introduciremos por terminal. Y que en caso de que no se especifique fichero de salida, se mostrará por pantalla.
Además, hemos hecho uso de un fichero *auxiliar.txt*, donde almacenaremos el texto tras su correspondiente "parseo" antes de su cifrado/descifrado. El parseo se hace en la función del fichero *utils.c* llamada *parsear* que esencialmente transforma las minúsculas en mayúsculas, quita acentos, e ignora espacios, tabulaciones y otros caracteres fuera del alfabeto acordado (A-Z).
Estas decisiones de diseño han sido tomadas con la finalidad de facilitar la reutilización de código, y hacer más fácil de realizar la depuración del cifrado y descifrado de textos.

En el fichero utils.c también encontraremos los algoritmos de Euclides (mcd) y Euclides Extendido (mcdExtended) con variables enteras, las cuales usaremos en todos los apartados de la práctica salvo el primero, pues tiene su propia implementación personalizada en GMP. Es decir, sólo hemos hecho en GMP el cifrado afín.

1. Sustitución Monoalfabeto

a. Método afín (USO OBLIGATORIO DE GMP)

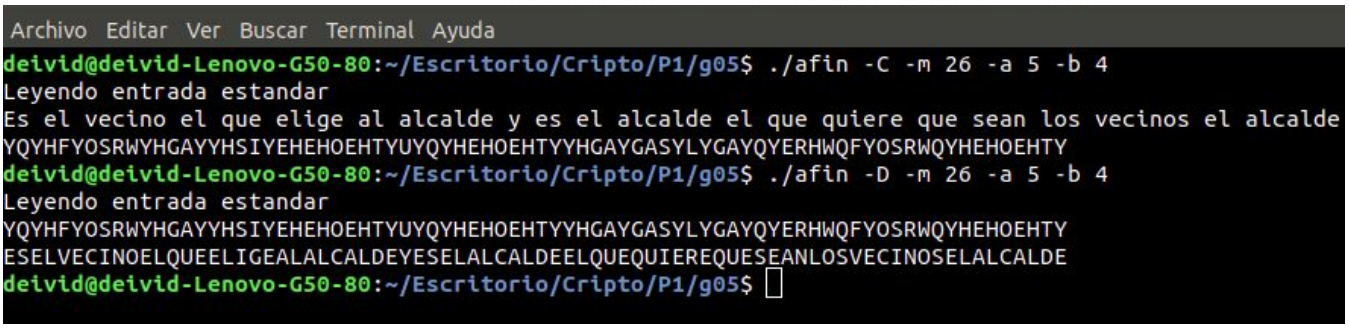
El método afín es un cifrado monoalfabético que consiste en la sustitución de cada caracter *x* por el carácter mediante el uso de la siguiente fórmula: $y = ax + b \ mod \ m$
Para verificar que la clave (a,b) determina una función afín inyectiva, se tiene que poder invertir el cifrado, es decir, tiene que existir a^{-1} para que se cumpla que $x = a^{-1}(y - b) \ mod \ m$
Que exista este inverso modular es equivalente a que se dé que mcd(a, m) = 1. Por eso mismo, en el código hemos hecho dicha comprobación al principio.

Para comprobar la inyectividad del cifrado y calcular el inverso modular de a, hemos implementado en gmp, los algoritmos de Euclides *(gmp_mcd)* y Euclides extendido *(gmp_mcdext)* de forma análoga a las funciones *mcd* y *mcdExtended*, pero haciendo uso de las librerías gmp.

Una vez explicados los algoritmos anteriores, la lógica del programa es muy sencilla, pues consiste en implementar las operaciones anteriores en gmp:

$y = ax + b \ mod \ m$ para cifrar caracter a caracter.
$x = a^{-1}(y - b) \ mod \ m$ para descifrar caracter a caracter.
Cabe destacar, que para transformar los caracteres a la aritmética módulo m, hemos restado 65 a cada carácter de la operación. De esta forma los caracteres restringen su dominio al anillo Z_m. Posteriormente, hemos vuelto a sumar 65 para que se vuelva al alfabeto original.

Veamos un ejemplo de ejecución en el que se cifra y se descifra:



Como vemos, al no tener fichero de entrada ni salida en el comando, se tomarán los valores estándar (teclado y pantalla). Tomando la frase de partida, se parsea y se cifra en mayúsculas. Copiamos el texto cifrado y lo introducimos con la opción de descifrar, volviendo a obtener el texto original en formato de mayúsculas y sin espacios.

b. Aplicación Práctica (USO OBLIGATORIO DE GMP)

El método que hemos ideado consiste en realizar el producto de los criptosistemas afín y el de permutaciones. Este se basa en la idea del algoritmo DES de aumentar la difusión y confusión del texto cifrado, por lo que la realización de ataques estadísticos se vuelve más difícil, y de esta forma se pueden dificultar los ataques de tipo A.

En el caso en que tengamos parejas de textos planos y cifrados, es decir con ataques de tipos B-E, el criptoanálisis es más sencillo, pero aún así sigue siendo más complicado que un cifrado afín estándar. Esto es porque aparte de cifrar caracter a caracter, permutamos un número n desconocido de caracteres que puede ser muy grande, lo cuál sigue haciendo la obtención de la clave difícil, sobre todo si no se elige una pareja de textos planos y cifrados adecuados.
El número de claves posibles es el producto de las claves posibles de ambos criptosistemas. Es decir $|K| = m * \varphi(m) * n!$ Siendo n el tamaño de la clave de permutación y m el módulo que se aplica al cifrado afín.

Un ejemplo puede ser con la clave (a, b) = (3, 5) en módulo 26 y la permutación (3 2 4 6 5 1)
El texto *CRIPTO* quedaría tras la permutación como *IRPOTC*
Y tras el cifrado afín nos queda *DEYVKL*
Cabe destacar que este producto de criptosistemas es conmutativo, pues se pueden intercambiar el orden de los cifrados sin alterar el resultado, pero sigue siendo más robusto que el cifrado afín estándar.

2. Sustitución Polialfabeto

a. Método de Hill (USO DE GMP NO OBLIGATORIO)

El método del cifrado de Hill se basa en encriptar el mensaje utilizado una matriz de claves, siguiendo la fórmula siguiente: $Y = KX$ donde *K* es la matriz cuadrada de dimensión *NxN* de claves, *X* representa una trozo de longitud *N* del mensaje e *Y* es el resultado de cifrar ese trozo del mensaje. Para que este método defina un criptosistema es necesario que sea inyectivo, es decir, que el determinante de la matriz *K* no sea nulo, y que sea invertible (condición que ya incluye determinante no nulo).

El proceso de implementación es sencillo, recibimos mediante parámetros de entrada (además de los posibles ficheros de entrada y salida y el módulo del alfabeto) un fichero que contiene la matriz de claves (máximo hasta 3x3) y el número *N* que será el tamaño de los bloques a cifrar. Realizamos una comprobación para ver si la matriz de claves *K* tiene inversa, y en tal caso, la calculamos, ya que será necesaria para descifrar con la siguiente fórmula $X = K^{-1}Y$

```
for (i = 0; i < n; i++) {
    /*Cifrar*/
    if (cifrar == 1) {
        resultado = mult(matrix[i], cadena, n);
    }/*Descifrar*/
    else {
        resultado = mult(inversa[i], cadena, n);
    }
```

El fichero de la matriz de claves tendrá el formato siguiente:

6	24	1
13	16	10
20	17	15

donde la separación entre filas será por saltos de línea y la separación entre columnas será por espacios o tabulador.

Además, hemos controlado que el tamaño del texto a cifrar sea múltiplo de N para poder cifrarlo. En caso de que sea menor, lo hemos rellenado escribiendo un padding de W's.
Veamos, de nuevo, un ejemplo de ejecución en el que se cifra y se descifra el mismo texto.

```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
deivid@deivid-Lenovo-G50-80:~/Escritorio/Cripto/P1/g05$ ./hill -C -m 26 -n 3 -k claves.txt
Leyendo entrada estandar
Deja a los chavalotes, Pablo. Déjalos que caminen como ellos camelen, si los chavales camelan pegarle
un poquito a la lejía o camelan pegarle un poquito a la mandanga, pues déjalos
TLDLGJYWSLPNSEHUXQNXDPHJGOZEBAIWSYQMAOZKDKWESGRTAGEXSFQYZOELHYFKYTYXMYQMCUHAHDBOLARQKDXMKAIDUZTVOQPAKI
GYEBNZKHQDSMXIRAHYTGPRGDCKWUJZOFYCJYCKDNTHSOG
deivid@deivid-Lenovo-G50-80:~/Escritorio/Cripto/P1/g05$ ./hill -D -m 26 -n 3 -k claves.txt
Leyendo entrada estandar
TLDLGJYWSLPNSEHUXQNXDPHJGOZEBAIWSYQMAOZKDKWESGRTAGEXSFQYZOELHYFKYTYXMYQMCUHAHDBOLARQKDXMKAIDUZTVOQPAKI
GYEBNZKHQDSMXIRAHYTGPRGDCKWUJZOFYCJYCKDNTHSOG
DEJAALOSCHAVALOTESPABLODEJALOSQUECAMINENCOMOELLOSCAMELENSILOSCHAVALESCAMELANPEGARLEUNPOQUITOALALEJIAOC
AMELANPEGARLEUNPOQUITOALAMANDANGAPUESDEJALOSW
deivid@deivid-Lenovo-G50-80:~/Escritorio/Cripto/P1/g05$
```

El fichero *claves.txt* corresponde a la matriz de ejemplo mostrada anteriormente. Observamos que el texto a cifrar es parseado (quitando los acentos, espacios y convirtiendo a mayúsculas) y cifrado. Copiando el texto resultante con la opción de descifrar, volvemos a obtener el texto de partida, con una “W” al final, pues es nuestro padding.

b. Método de Vigenere (USO DE GMP NO OBLIGATORIO)

Este método consiste un cifrado de bloque mediante la clave que le pasamos por argumento. De esta forma, vamos cogiendo bloques del texto plano de tamaño la clave, y vamos haciendo una suma del bloque de texto plano $\overline{x}$ con el vector clave $\overline{k}$ para obtener el bloque cifrado $\overline{y}$ de la siguiente manera: $\overline{y} = \overline{x} + \overline{k} \; mod \; m$ (lo cual es una simple suma componente a componente de cada vector). Para descifrar, basta con despejar en la fórmula y obtenemos que el texto original es: $\overline{x} = \overline{y} - \overline{k} \; mod \; m$

```
/*Cifrar*/
if (cifrar == 1) {
    cadena[i] += clave[i];
}/*Descifrar*/
else {
    cadena[i] -= clave[i];
}
```

No hay mucho más que destacar aparte de que se ha hecho el mismo control del tamaño del texto a cifrar como en el Método de Hill y que la clave que se le pasa por argumento tiene que estar en el alfabeto acordado (e.g. DBCASV es válida como clave).

Veamos un sencillo ejemplo de ejecución, en el que se cifra y se descifra para obtener el mismo texto:

```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
deivid@deivid-Lenovo-G50-80:~/Escritorio/Cripto/P1/g05$ ./vigenere -C -m 26 -k FARY
Leyendo entrada estandar
A ver si pasamos limpios a junio
FVVPXIGYXADMXLZKUIFQFJLLNONU
deivid@deivid-Lenovo-G50-80:~/Escritorio/Cripto/P1/g05$ ./vigenere -D -m 26 -k FARY
Leyendo entrada estandar
FVVPXIGYXADMXLZKUIFQFJLLNONU
AVERSIPASAMOSLIMPIOSAJUNIOWW
deivid@deivid-Lenovo-G50-80:~/Escritorio/Cripto/P1/g05$
```

Comprobamos que el programa cifra y descifra correctamente, volviendo a obtener el texto que introducimos al principio para encriptar.

c. Criptoanálisis del Vigenere (GMP NO OBLIGATORIO)

Hemos realizado 2 programas:

- kasiski.c

Este programa se ejecuta con un comando con la siguiente estructura:

./kasiski {-l Ngrama} [-i filein] [-o fileout]

Ngrama será el tamaño mínimo de subcadena a considerar para mostrar sus repeticiones (por defecto está a 3 si no se especifica).

filein es el fichero con el texto cifrado.

fileout el fichero donde escribiremos el resultado, que consistirá en las cadenas que detectamos repetidas con sus posiciones y distancias entre ellas.

Con este método sacaremos las subcadenas repetidas en un mismo fichero y las distancias entre ellas. Además imprimiremos el mcd de las cadenas repetidas con al menos 2 repeticiones (para que tenga sentido hacer el mcd). Esto se verá mejor en el apartado de ejemplos.

- *IC.c*

./IC {-lmin Ngrama1} {-lmax Ngrama2} [-i filein] [-o fileout] {-E|-C}

Ngrama1 es el tamaño mínimo de clave que probaremos en esta ejecución.

Ngrama2 es el tamaño máximo de clave que probaremos en esta ejecución.

filein es el fichero con el texto cifrado.

fileout es el fichero donde mostraremos los datos relevantes con esos tamaños de clave.

-E indica que el idioma es Inglés

-C indica que el idioma es Castellano (valor por defecto)

Con este método, calcularemos los IC de los idiomas Inglés y Castellano, y los imprimiremos en el fichero de salida. A continuación, calcularemos el IC para los tamaños fijados en el intervalo [lmin, lmax]. En caso de que la media de los IC calculados se aproxime (bajo un ERROR definido de 0.01) al IC del idioma elegido, procederemos con el resto del criptoanálisis.

Este criptoanálisis consiste en probar, fijado el tamaño que asumimos correcto, con las distintas subclaves posibles, y calcular sus respectivos IC. Y, cogeremos el IC de la subclave que más se aproxime al IC del idioma.

Podremos ver estos resultados más visualmente en el apartado de ejemplos.

- *Ejemplos*

Hemos hecho pruebas con el primer capítulo del quijote, cifrándolo con claves de tamaños 3,4 y 5 (CAR, FRAN y DAVID)

```
carlos@carlos-SVF1521N6EW:~/Escritorio/practicas/Cripto/P1/g05$ ./vigenere -C -m 26 -k CAR -i quijote.txt -o v3.txt
carlos@carlos-SVF1521N6EW:~/Escritorio/practicas/Cripto/P1/g05$ ./vigenere -C -m 26 -k FRAN -i quijote.txt -o v4.txt
carlos@carlos-SVF1521N6EW:~/Escritorio/practicas/Cripto/P1/g05$ ./vigenere -C -m 26 -k DAVID -i quijote.txt -o v5.txt
```

A continuación hemos pasado kasiski a los 3 ficheros generados con tamaño de subcadena repetida mínima 3:

```
carlos@carlos-SVF1521N6EW:~/Escritorio/practicas/Cripto/P1/g05$ ./kasiski -l 3 -i v3.txt -o k3.txt
carlos@carlos-SVF1521N6EW:~/Escritorio/practicas/Cripto/P1/g05$ ./kasiski -l 3 -i v4.txt -o k4.txt
carlos@carlos-SVF1521N6EW:~/Escritorio/practicas/Cripto/P1/g05$ ./kasiski -l 3 -i v5.txt -o k5.txt
```

Y hemos pasado IC a los 3 ficheros, probando tamaños de clave de 2 a 10:

```
carlos@carlos-SVF1521N6EW:~/Escritorio/practicas/Cripto/P1/g05$ ./IC -lmin 2 -lmax 10 -i v3.txt -o IC3.txt
carlos@carlos-SVF1521N6EW:~/Escritorio/practicas/Cripto/P1/g05$ ./IC -lmin 2 -lmax 10 -i v4.txt -o IC4.txt
carlos@carlos-SVF1521N6EW:~/Escritorio/practicas/Cripto/P1/g05$ ./IC -lmin 2 -lmax 10 -i v5.txt -o IC5.txt
```

En el fichero de salida, siempre mostraremos al principio los IC de cada idioma, para poder comparar de forma más gráfica si los datos que mostraremos a continuación son fiables.

```
IC Castellano: 0.083235
IC Inglés: 0.065770
```

- **Clave de tamaño 3**

En el test de Kasiski hemos decidido aplicar siempre la búsqueda de subcadenas de tamaño 3.

En este caso, podemos comprobar que hay subcadenas repetidas, y en muchas de ellas los mcd de las distancias son de 3, por lo que es muy probable que el tamaño de la clave sea ese.

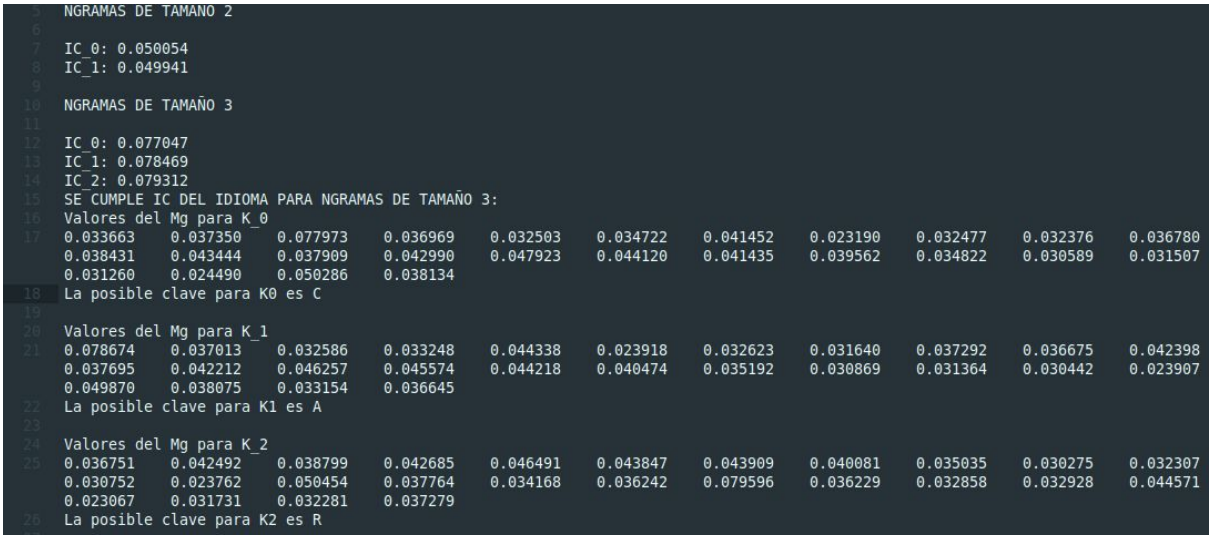
```
AIF posiciones: 7  y 3478 con distancia 3471
AIF posiciones: 7  y 6568 con distancia 6561
MCD: 3

IFE posiciones: 8  y 4073 con distancia 4065
IFE posiciones: 8  y 5402 con distancia 5394
IFE posiciones: 8  y 5441 con distancia 5433
MCD: 3

FEC posiciones: 9  y 81 con distancia 72
FEC posiciones: 9  y 483 con distancia 474
FECC    posiciones: 9  y 708 con distancia 699
FECC    posiciones: 9  y 1143 con distancia 1134
FECC    posiciones: 9  y 1602 con distancia 1593
FEC posiciones: 9  y 1812 con distancia 1803
FEC posiciones: 9  y 2637 con distancia 2628
FEC posiciones: 9  y 2685 con distancia 2676
FECC    posiciones: 9  y 2859 con distancia 2850
FECC    posiciones: 9  y 3474 con distancia 3465
FEC posiciones: 9  y 4626 con distancia 4617
FEC posiciones: 9  y 6396 con distancia 6387
FEC posiciones: 9  y 6447 con distancia 6438
FECC    posiciones: 9  y 6924 con distancia 6915
FECCMRPCYC  posiciones: 9  y 6951 con distancia 6942
FEC posiciones: 9  y 7041 con distancia 7032
FEC posiciones: 9  y 8301 con distancia 8292
MCD: 3
```

Usando el método de los IC podemos comprobar que para tamaño 2, los IC calculados no se parecen nada a los del idioma, pero con tamaño 3 son muy próximos, por tanto, hemos proseguido el criptoanálisis, y hemos calculado las tablas con los Mg, es decir, los IC calculados si probamos ciertas subclaves.

Podemos observar que se cumple que hay una subclave para cada coordenada de la clave que tiene asociado un IC más alto y cercano al del IC del idioma, que son el 3º en la primera tabla, el 1º en la segunda tabla y el 17º, que se corresponden con las letras CAR, que se corresponden con las de la clave que usamos.



```
NGRAMAS DE TAMANO 2

IC_0: 0.050054
IC_1: 0.049941

NGRAMAS DE TAMAÑO 3

IC_0: 0.077047
IC_1: 0.078469
IC_2: 0.079312
SE CUMPLE IC DEL IDIOMA PARA NGRAMAS DE TAMAÑO 3:
Valores del Mg para K_0
0.033663    0.037350    0.077973    0.036969    0.032503    0.034722    0.041452    0.023190    0.032477    0.032376    0.036780
0.038431    0.043444    0.037909    0.042990    0.047923    0.044120    0.041435    0.039562    0.034822    0.030589    0.031507
0.031260    0.024490    0.050286    0.038134
La posible clave para K0 es C

Valores del Mg para K_1
0.078674    0.037013    0.032586    0.033248    0.044338    0.023918    0.032623    0.031640    0.037292    0.036675    0.042398
0.037695    0.042212    0.046257    0.045574    0.044218    0.040474    0.035192    0.030869    0.031364    0.030442    0.023907
0.049870    0.038075    0.033154    0.036645
La posible clave para K1 es A

Valores del Mg para K_2
0.036751    0.042492    0.038799    0.042685    0.046491    0.043847    0.043909    0.040081    0.035035    0.030275    0.032307
0.030752    0.023762    0.050454    0.037764    0.034168    0.036242    0.079596    0.036229    0.032858    0.032928    0.044571
0.023067    0.031731    0.032281    0.037279
La posible clave para K2 es R
```

El resto de pruebas de Ngramas también están en el fichero, pero por no incurrir en mucha redundancia no las mostraremos. Lo único interesante que podríamos ver es que para los demás tamaños de clave no se cumple el umbral de error, salvo para los tamaños 6 y 9, lo cuál tiene sentido, pues son múltiplos de 3. Las claves que dan en esos casos son CARCAR y CARCARCAR, lo que es consistente con la clave real. Obviamente estas claves podemos obviarlas por redundancia, y seguimos aceptando la clave CAR como la correcta.

- **Clave de tamaño 4**

Ahora probaremos con tamaño 4 el test de Kasiski, aparte de muchos ejemplos donde el mcd nos da 4, hay algún caso donde nos da 1, pero los atribuímos a casos de error, los cuáles son despreciables, ya que obviamente se detectaría rápido si el tamaño de clave fuera 1.

```
WUE posiciones: 8  y 3064 con distancia 3056
WUE posiciones: 8  y 6156 con distancia 6148
WUE posiciones: 8  y 6976 con distancia 6968
WUE posiciones: 8  y 8016 con distancia 8008
MCD: 4

UEY posiciones: 9  y 81 con distancia 72
UEYF    posiciones: 9  y 1681 con distancia 1672
UEYF    posiciones: 9  y 2345 con distancia 2336
UEY posiciones: 9  y 2637 con distancia 2628
UEY posiciones: 9  y 2685 con distancia 2676
UEYF    posiciones: 9  y 3653 con distancia 3644
UEYF    posiciones: 9  y 3701 con distancia 3692
UEY posiciones: 9  y 4693 con distancia 4684
UEY posiciones: 9  y 7041 con distancia 7032
UEY posiciones: 9  y 8301 con distancia 8292
MCD: 4

EYF posiciones: 10  y 1682 con distancia 1672
EYF posiciones: 10  y 2346 con distancia 2336
EYF posiciones: 10  y 3654 con distancia 3644
EYF posiciones: 10  y 3702 con distancia 3692
EYF posiciones: 10  y 7082 con distancia 7072
EYF posiciones: 10  y 7794 con distancia 7784
MCD: 4

YFD posiciones: 11  y 3284 con distancia 3273
YFD posiciones: 11  y 4008 con distancia 3997
YFD posiciones: 11  y 4260 con distancia 4249
YFD posiciones: 11  y 6432 con distancia 6421
YFDA    posiciones: 11  y 6671 con distancia 6660
YFD posiciones: 11  y 6859 con distancia 6848
YFDA    posiciones: 11  y 8107 con distancia 8096
MCD: 1
```

Al realizar IC sobre este texto cifrado, podemos observar como en el caso anterior, que los IC de tamaños distintos de 4 toman valores muy distintos a los del IC del idioma, y con tamaño 4, se aproxima lo suficiente, por lo que se calcula la clave como en el ejemplo anterior, obteniendo la clave correcta FRAN.

```
NGRAMAS DE TAMAÑO 2

IC_0: 0.050997
IC_1: 0.060943

NGRAMAS DE TAMAÑO 3

IC_0: 0.048550
IC_1: 0.049619
IC_2: 0.048545

NGRAMAS DE TAMAÑO 4

IC_0: 0.080394
IC_1: 0.076079
IC_2: 0.076741
IC_3: 0.079960
SE CUMPLE IC DEL IDIOMA PARA NGRAMAS DE TAMAÑO 4:
Valores del Mg para K_0
0.022669    0.050438    0.037810    0.033324    0.036474    0.079843    0.036538    0.032591    0.033925    0.042756    0.022867
0.033005    0.032379    0.036457    0.038444    0.041906    0.036739    0.043297    0.048550    0.043639    0.041733    0.041327
0.036111    0.030644    0.031958    0.031049
La posible clave para K0 es F

Valores del Mg para K_1
0.036825    0.043538    0.037865    0.040840    0.046605    0.046018    0.043695    0.039997    0.035576    0.030714    0.032198
0.030606    0.024071    0.047868    0.037213    0.033343    0.038102    0.077876    0.037260    0.033297    0.034801    0.044371
0.023981    0.031810    0.031413    0.036590
La posible clave para K1 es R

Valores del Mg para K_2
0.077857    0.036643    0.032240    0.032819    0.043972    0.023839    0.032159    0.032280    0.037904    0.036666    0.043324
0.039005    0.042099    0.045737    0.044696    0.043106    0.039844    0.034712    0.029821    0.031525    0.031706    0.025241
0.050449    0.038649    0.033953    0.036227
La posible clave para K2 es A
```

```
Valores del Mg para K_3
0.046650    0.043718    0.044204    0.038975    0.033656    0.031275    0.031272    0.029925    0.024229    0.052148    0.038275
0.034026    0.036163    0.079379    0.036494    0.032498    0.033001    0.042771    0.022915    0.032135    0.032334    0.037582
0.037255    0.042343    0.038991    0.044260
La posible clave para K3 es N


NGRAMAS DE TAMAÑO 5

IC_0: 0.050129
IC_1: 0.048984
IC_2: 0.047635
IC_3: 0.048280
IC_4: 0.048752

NGRAMAS DE TAMAÑO 6

IC_0: 0.050157
IC_1: 0.063621
IC_2: 0.052289
IC_3: 0.058279
IC_4: 0.050285
IC_5: 0.061059

NGRAMAS DE TAMAÑO 7

IC_0: 0.049897
IC_1: 0.049080
IC_2: 0.048639
IC_3: 0.049400
IC_4: 0.050408
IC_5: 0.046404
IC_6: 0.048119
```

- **Clave de tamaño 5**

En este ejemplo obtenemos muchos mcd de valor 5 y alguno de valor 1 como en el caso anterior, por lo que consideramos el análisis correcto, con tamaño concluído de clave 5.

```
HLV posiciones: 10  y 965 con distancia 955
HLV posiciones: 10  y 1835 con distancia 1825
HLV posiciones: 10  y 1915 con distancia 1905
HLV posiciones: 10  y 2240 con distancia 2230
HLV posiciones: 10  y 2860 con distancia 2850
HLV posiciones: 10  y 3085 con distancia 3075
HLV posiciones: 10  y 3475 con distancia 3465
HLV posiciones: 10  y 5655 con distancia 5645
HLV posiciones: 10  y 6925 con distancia 6915
MCD: 5

LVUD    posiciones: 11  y 6671 con distancia 6660
LVUD    posiciones: 11  y 6781 con distancia 6770
LVUD    posiciones: 11  y 8286 con distancia 8275
MCD: 5

VUD posiciones: 12  y 1608 con distancia 1596
VUD posiciones: 12  y 4187 con distancia 4175
VUD posiciones: 12  y 4268 con distancia 4256
VUD posiciones: 12  y 6672 con distancia 6660
VUD posiciones: 12  y 6782 con distancia 6770
VUD posiciones: 12  y 8077 con distancia 8065
VUD posiciones: 12  y 8287 con distancia 8275
MCD: 1
```

Usando el método de IC comprobamos que concuerda con el test de Kasiski, pues los únicos IC parecidos a los del idioma son cuando tomamos el tamaño de clave 5. Además en ese caso, se calculan las subclaves de forma correcta, quedando la clave final DAVID.

```
NGRAMAS DE TAMAÑO 2

IC_0: 0.045597
IC_1: 0.046201

NGRAMAS DE TAMAÑO 3

IC_0: 0.046065
IC_1: 0.045824
IC_2: 0.046063

NGRAMAS DE TAMAÑO 4

IC_0: 0.046037
IC_1: 0.045430
IC_2: 0.045021
IC_3: 0.046811

NGRAMAS DE TAMAÑO 5

IC_0: 0.081594
IC_1: 0.080604
IC_2: 0.076616
IC_3: 0.076079
IC_4: 0.076951
SE CUMPLE IC DEL IDIOMA PARA NGRAMAS DE TAMAÑO 5:
Valores del Mg para K_0
0.038825    0.031492    0.036072    0.080268    0.036642    0.031068    0.033388    0.044737    0.023518    0.032116    0.032021
0.037062    0.037684    0.044030    0.037658    0.041519    0.046186    0.045042    0.042915    0.040390    0.033985    0.031188
0.031322    0.031244    0.024983    0.051237
La posible clave para K0 es D

Valores del Mg para K_1
0.079669    0.036147    0.031365    0.033508    0.041347    0.022434    0.031514    0.032493    0.037675    0.039398    0.042063
0.038853    0.042489    0.048000    0.043437    0.043708    0.037810    0.034553    0.031063    0.032121    0.029360    0.024034
0.051528    0.039215    0.034178    0.038628
La posible clave para K1 es A

Valores del Mg para K_2
0.023793    0.031712    0.031540    0.036471    0.035627    0.043725    0.039188    0.042819    0.046922    0.044726    0.043279
0.039792    0.035551    0.030238    0.031971    0.030882    0.023979    0.048703    0.036877    0.034032    0.037317    0.078538
0.037424    0.032689    0.034256    0.044539
La posible clave para K2 es V

Valores del Mg para K_3
0.030216    0.032066    0.030401    0.023938    0.050129    0.038252    0.034736    0.036292    0.077410    0.035947    0.033976
0.033586    0.042984    0.023576    0.033100    0.032244    0.037551    0.036867    0.041316    0.037901    0.043471    0.047687
0.043760    0.043988    0.040448    0.034749
La posible clave para K3 es I

Valores del Mg para K_4
0.036831    0.033911    0.035464    0.077950    0.037569    0.034185    0.033464    0.043713    0.023664    0.032983    0.032234
0.036870    0.036896    0.042804    0.037117    0.042898    0.045713    0.045655    0.042099    0.041802    0.036285    0.030218
0.031186    0.032238    0.023360    0.049481
La posible clave para K4 es D


NGRAMAS DE TAMAÑO 6

IC_0: 0.045882
IC_1: 0.046395
IC_2: 0.045802
IC_3: 0.046029
IC_4: 0.045278
IC_5: 0.046249

NGRAMAS DE TAMAÑO 7

IC_0: 0.047020
IC_1: 0.044712
IC_2: 0.046017
IC_3: 0.046818
IC_4: 0.046647
IC_5: 0.044871
IC_6: 0.046717
```

3. Método de transposición (GMP NO OBLIGATORIO)

El cifrado mediante transposición consiste en dividir el texto en bloques de tamaño *N,* como en Hill, para realizar el cifrado. El proceso consiste en tomar como entrada del programa una permutación de *N* números, y esta será la forma en la que los elementos por bloques se reordenen. Si no introducimos permutación con el flag *-p,* el programa generará una permutación aleatoria para cifrar y descifrar de tamaño *n.* Por ejemplo, si la permutación es "5 4 3 2 1" con 5 elementos, el bloque se ordenará en sentido contrario, convirtiendo la palabra "perro" en "orrep".

Podemos realizar este ejercicio de dos formas diferentes. En nuestro caso, hemos empleado, por simplicidad, la primera:

- **Permutación de elementos**: tomando como referencia la permutación dada y el trozo de texto, vamos escribiendo en el fichero de salida en el orden que nos indican los números de la permutación. Por ejemplo, para p = "5 4 3 2 1" y la palabra "perro", iremos escribiendo en el fichero la letra 5, luego la 4, luego la 3… obteniendo "orrep".
  Para descifrar, utilizamos exactamente el mismo método pero utilizando la permutación "inversa" para ordenar, es decir, la permutación que vuelve a dejar el texto cifrado como el de partida.

```
/*leer fichero entrada o estandar*/
if (fIn) {
    while (fread(cadena, sizeof (char), n, fIn) != 0) {

        for (i = 0; i < n; i++) {
            /*Cifrar*/
            if (cifrar == 1) {
                simbolo_out = cadena[permutacion[i] - 1];
            }/*Descifrar*/
            else {
                simbolo_out = cadena[inversa[i] - 1];
            }
```

Un ejemplo de cifrado y descifrado del mismo texto:

```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
deivid@deivid-Lenovo-G50-80:~/Escritorio/Cripto/P1/g05$ ./transposicion -C -n 6
Leyendo entrada estandar
HOLITA CARLOS
ILAOTHLRSAOC
deivid@deivid-Lenovo-G50-80:~/Escritorio/Cripto/P1/g05$ ./transposicion -D -n 6
Leyendo entrada estandar
ILAOTHLRSAOC
HOLITACARLOS
deivid@deivid-Lenovo-G50-80:~/Escritorio/Cripto/P1/g05$ ./transposicion -C -p "4 5 1 3 6 2" -n 6
Leyendo entrada estandar
HOLITA CARLOS
ITHLAOLOCRSA
deivid@deivid-Lenovo-G50-80:~/Escritorio/Cripto/P1/g05$ ./transposicion -D -p "4 5 1 3 6 2" -n 6
Leyendo entrada estandar
ITHLAOLOCRSA
HOLITACARLOS
```

Comprobamos fácilmente que el cifrado y el descifrado funcionan correctamente y se vuelve a obtener el texto que hemos introducido por teclado al principio.
En el primer cifrado-descifrado, utilizamos el programa sin la opción *-p* , por tanto, genera una permutación aleatoria de tamaño *n.* Esta permutación se guardará en un fichero *permutacion.dat* que será consultado para poder descifrar luego ese mismo texto.
En el segundo cifrado-descifrado, utilizamos el programa con la opción *-p* y la permutación deseada, por lo que para descifrar será necesario volver a incluirla.

- **Variación de Hill**: puede expresarse el método de transposición como una variante de Hill, simplemente utilizando la permutación dada para crear una matriz. Para ello, colocamos los 1's en la posición que nos indica la permutación. Por ejemplo, si tenemos p = "5 4 3 2 1", la matriz utilizada con el método de Hill será:

0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	1	0	0	0
1	0	0	0	0

  Para descifrar, simplemente utilizamos el método ya implementado para el cálculo de la inversa, con esta nueva matriz.
  Hemos decidido no implementarlo con este método, ya que actualmente Hill lo realizamos con un tamaño máximo de matriz de 3x3, por lo que consecuentemente el tamaño de permutación estaría limitado por 3.

4. Producto de criptosistemas permutación

Este método de cifrado consiste en dividir el texto original en matrices MxN (M filas y N columnas), de modo que vuelve a ser un algoritmo que trabaja por bloques. Para ello, utilizamos un *fread* de bloques M*N y con ayuda de *memcpy* vamos rellenando nuestra matriz de texto a cifrar.

```
/*leer fichero entrada o estandar*/
if (fIn) {
    while (fread(cadena, sizeof (char), m*n, fIn) != 0) {

        for (i = 0; i < m; i++) {
            memcpy(matrix[i], cadena+(n*i), n);
        }
```

Una vez hemos dividido el texto en matrices, utilizamos los argumentos de entrada *k1* y *k2* que son vectores que contienen las permutaciones de filas y columnas que vamos a realizar a cada una de esas matrices para obtener el texto cifrado.

```
/*Cifrar*/
if (cifrar == 1){
    /*permutar filas*/
    for (r = 0; r < m; r++){
        t = perm_fila[r];
        for (s = 0; s < n; s++){
            matrix2[r][s] = matrix[t-1][s];
        }
    }
    /*permutar columnas*/
    for (r = 0; r < n; r++){
        t = perm_columna[r];
        for (s = 0; s < m; s++){
            matrix3[s][r] = matrix2[s][t-1];
        }
    }
```

Veamos con un ejemplo el proceso que hemos seguido para realizar dicha operación. Partiendo de la matriz de texto cifrado:

A	B	C	D
E	F	G	H
I	J	K	L

Tomamos los elementos de la permutación de filas *k1 = "3 2 1"*, por tanto las filas quedarán en orden inverso, obteniendo la matriz:

I	J	K	L
E	F	G	H
A	B	C	D

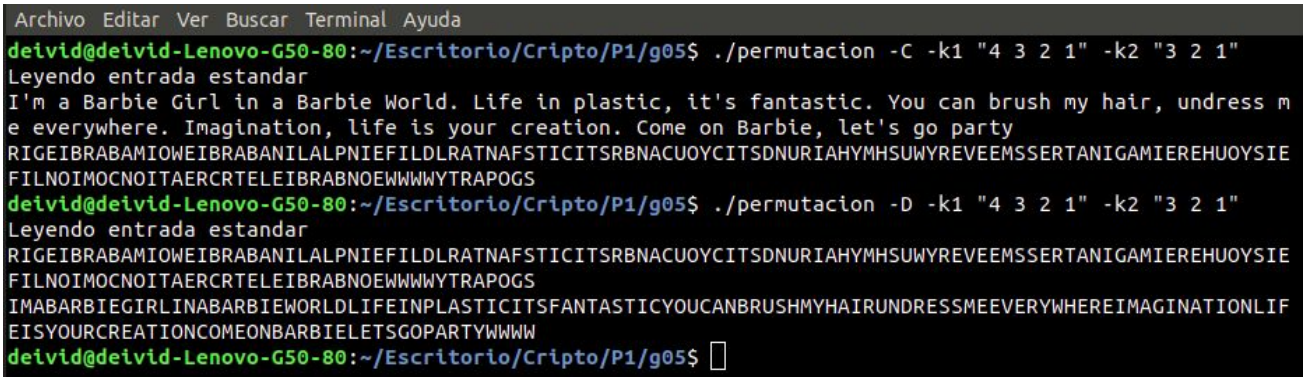
Posteriormente, tomamos los elementos de la permutación de columnas *k2 = "4 3 2 1"*, que, de nuevo, volverá a dejar en orden inverso, esta vez, las columnas:

L	K	J	I
H	G	F	E
D	C	B	A

Por tanto, lo que escribamos en el fichero de salida se escribirá el texto cifrado "LKJIHGFEDCBA".

El proceso para descifrar es exactamente el mismo, salvo que primero tendremos que permutar las columnas y luego las filas, para obtener la matriz original en cada caso. Las permutaciones para descifrar serán las que comentamos anteriormente en el método de transposición, aquellas que aplicándolas se deshace la permutación original.

Veamos un ejemplo de ejecución de cifrado y descifrado mediante este método:



El texto introducido para cifrar es parseado, quitando comillas y signos de puntuación. Al descifrar obtenemos el texto original con el parseo mencionado y con el padding al final.

¿Que posible criptoanálisis podría sufrir este doble cifrado de permutación y bajo qué supuestos de modelo de seguridad?

Asumiendo que no tenemos ningún par de textos cifrados y descifrados (Ataque de tipo A), se podría intentar criptoanalizar probando tamaños de bloque MxN para hallar la dimensión de la matriz (cada bloque de texto), con algún algoritmo del estilo Kasiski, con el que se tengan en cuentan las posibles repeticiones del mismo trozo de código cifrado de tamaño MxN a lo largo del texto y de esta manera podemos obtener el tamaño del bloque. A continuación, habría que averiguar los tamaños M y N por separado.
A partir de ahí, para cada tamaño posible M y N, se tendrían que probar las posibles combinaciones de permutaciones de filas y columnas que se hubieran podido realizar al cifrar, para hallar cuáles de ellas dan un texto original con sentido.
Para realizar este criptoanálisis, sería necesario un texto cifrado muy grande para que el test de Kasiski sea efectivo, pues básicamente buscamos claves de tamaño MxN, que pueden ser muy grandes.

En caso de que tengamos pares de textos cifrados y descifrados (ataques de tipo B-E), entonces, si tuviéramos algún bloque de texto cifrado donde no se repita ninguna letra, podríamos romper el algoritmo, pues al tener el texto correspondiente descifrado, existe una asignación única de posiciones posible, por lo que seríamos capaces de descifrar el resto del texto sin problema.