

Laura Sánchez Herrera

Daniel Mateo Moreno

Pareja: 5

Práctica 1: Criptografía Clásica

1. Sustitución Monoalfabeto

A. Método afín

Para construir el programa que realice cifrados y descifrados de textos mediante el método Afín, primero, hemos creado un fichero que contiene todo este proceso (es decir, el main) que se llama **afin.c**. Además, hemos añadido una librería **afin.h** con algunas definiciones importantes, como las declaraciones de las funciones de descifrar (descifrar()) y cifrar (cifrar()), y la función encargada de procesar los argumentos introducidos (read_args_afin()).

Esta última función procesa los números a, b y m introducidos por argumento, asociandolos a su correspondientes estructuras de gmp. Si se introducen ficheros de entrada y salida, o no, el programa indica posteriormente al módulo principal el tipo de modo en el que se encuentra (mostrar y leer todo por teclado, leer de un fichero de entrada o escribir en un fichero de salida, o ambos a la vez).

Tras obtener el modo, el programa principal comprueba que los números a y m son coprimos entre sí, es decir, que cumplan que su MCD es igual a 1. Esto se realiza en el módulo **euclides_ext.c**, en la función de euclides_ext(). Hemos decidido que el algoritmo de Euclides y de Euclides extendido se encuentren en la misma función, ya que para realizar el extendido es necesario que se cumpla que a y m son coprimos, lo que se comprueba antes con el algoritmo de Euclides normal, y si no lo son esta función simplemente devuelve ERROR. Si son coprimos, esta función continúa para calcular el inverso multiplicativo de a en el espacio m.

Estos algoritmos se han realizado como hemos visto en teoría, es decir, para Euclides normal hemos realizado la descomposición secuencial de los restos hasta encontrar el resto 0 y mirando si el anterior a ese es 1. Si se cumple esto, ambos son coprimos. Si no, poseen un mayor divisor común distinto de 1 y no son coprimos.

El algoritmo de Euclides extendido es algo más complejo. Primero, en el bucle anterior de Euclides normal, hemos almacenado en un array de estructuras de gmp los números resultantes en cada iteración de descomposición (r_n, r_n+1, r_n+2 y el cociente). Con estos números almacenados, hemos creado un bucle inverso que va desde la igualdad a 1 hasta el principio, mediante la sustitución y recursividad vistas en clase de este algoritmo. Cuando aparecen a y m en esta igualdad se finaliza el bucle y obtenemos los factores multiplicativos que cumplen que $1 = a*x + m*y$,

siendo x e y estos factores (en nuestra función son las variables c y x). Con el factor x conocido, realizamos su módulo sobre m y obtenemos el inverso multiplicativo de a, lo introducimos en la variable inv y devolvemos OK.

Falta indicar que habíamos creado un header **euclides_ext.h** para este módulo.

A continuación, dejamos dos capturas de la ejecución del euclides extendido y el normal, ya que poseen salidas por pantalla para observar su proceso y resultados. En la primera, a y m son coprimos, por lo que haya su inverso. En la segunda, no son coprimos, por lo que indica el error y no continúa con el cifrado/descifrado.

```
X541@LAPTOP-DanMat27 /c/Users/x541/desktop/cripto/p1
$ ./afin -C -m 27 -a 7 -b 2
> Numeros introducidos:
m | a : 27 | 7

> Desarrollo del algoritmo de euclides:
27 = 3 x 7 + 6
7 = 1 x 6 + 1
6 = 6 x 1 + 0

mcd(27,7) = 1 => SI SON COPRIMOS

> Se va a proceder a hallar el inverso multiplicativo modular por el algoritmo de euclides extendido:
6 = 27 - 3 x 7

1 = 4*7 mod 27
Inverso de 7 en Z_27 es = 4

# Ha elegido cifrar #

Introduzca lo que quiere cifrar:
```

```
X541@LAPTOP-DanMat27 /c/Users/x541/desktop/cripto/p1
$ ./afin -C -m 27 -a 9 -b 2
> Numeros introducidos:
m | a : 27 | 9

> Desarrollo del algoritmo de euclides:
27 = 3 x 9 + 0

mcd(27,9) = 9 => NO SON COPRIMOS
9 en Z_27 no tiene inverso multiplicativo modular
Imposible cifrar por el metodo afin porque a no tiene inverso multiplicativo en este espacio
```

Además, como se puede ver, la forma de ejecutar el ejecutable **afin** es, después de haber ejecutado el **Makefile** con “> make”, de la siguiente forma:

> ./afin {-C|-D} {-m num_m} {-a num_a} {-b num_b} [-i filein.txt] [-o fileout.txt]

Los dos últimos argumentos son optativos y hacen variar el modo, antes explicado, del programa. Los demás son obligatorios. Los paréntesis y las llaves no se escriben. En **comandos.txt** le incluimos ejemplos de ejecuciones, además de las de las capturas (los ficheros son ejemplos y deberán existir con el nombre que se quiera para que funcione la entrada y salida por ficheros).

Cabe añadir que en el programa, para evitar problemas en estos algoritmos, comprobamos antes que m sea mayor que a. Si no, informamos de este error, ya que el espacio m tiene que ser mayor que a.

Tras comprobar si son coprimos a y m, y obtener el inverso multiplicativo de a en el espacio m, el programa procede a cifrar o descifrar el texto que reciba por teclado o fichero de entrada (si es por teclado, se avisa al usuario para que escriba el texto a cifrar o descifrar por teclado, como se puede ver en las próximas capturas).

Para cifrar (en cifrar()), el programa recorre carácter a carácter del texto plano para aplicar la fórmula del método Afín sobre el número correspondiente al carácter (en este caso, el espacio es siempre 26 ya que se realiza sobre el ascii de A a Z, que corresponde al intervalo de 65 a 90). Esta ecuación es $c = a*x + b \text{ mod } m$, siendo x el carácter a cifrar y c lo obtenido. Y, una vez cifrado todo el texto plano, se imprime por pantalla o se escribe en un fichero indicado.

Luego, para descifrar es igual pero se aplica la fórmula $d = (x-b)*a^{-1} \text{ mod } m$, siendo x el carácter a descifrar y d lo obtenido. Una vez descifrado el texto cifrado, se imprime por pantalla o se escribe en un fichero indicado.

A continuación, dos ejemplos cifrando el texto plano “HOLA QUE TAL” y descifrando el texto cifrado correspondiente (en m=26 como hemos dicho, con una a coprima y una b elegidas al azar como ejemplo de clave).

```
X541@LAPTOP-DanMat27 /c/Users/x541/desktop/cripto/p1
$ ./afin -C -m 26 -a 7 -b 5
> Numeros introducidos:
m | a : 26 | 7

> Desarrollo del algoritmo de euclides:
26 = 3 x 7 + 5
7 = 1 x 5 + 2
5 = 2 x 2 + 1
2 = 2 x 1 + 0

mcd(26,7) = 1 => SI SON COPRIMOS

> Se va a proceder a hallar el inverso multiplicativo modular por el algoritmo de euclides extendido:
2 = 7 - 1 x 5
5 = 26 - 3 x 7

1 = -11*7 mod 26
Inverso de 7 en Z_26 es = 15

# Ha elegido cifrar #

Introduzca lo que quiere cifrar:
HOLA QUE TAL

Cifrando HOLA QUE TAL
Texto cifrado: CZEZF NPH IFE
```

```

X541@LAPTOP-DanMat27 /c/Users/x541/desktop/cripto/p1
$ ./afin -D -m 26 -a 7 -b 5
> Numeros introducidos:
m | a : 26 | 7

> Desarrollo del algoritmo de euclides:
26 = 3 x 7 + 5
7 = 1 x 5 + 2
5 = 2 x 2 + 1
2 = 2 x 1 + 0

mcd(26,7) = 1 => SI SON COPRIMOS

> Se va a proceder a hallar el inverso multiplicativo modular por el algoritmo de euclides extendido:
2 = 7 - 1 x 5
5 = 26 - 3 x 7

1 = -11*7 mod 26
Inverso de 7 en Z_26 es = 15

# Ha elegido descifrar #

Introduzca lo que quiere descifrar:
CZEF NPH IFE

Descifrando CZEF NPH IFE
Texto descifrado: HOLA QUE TAL

```

Como se puede ver, se realizan correctamente los procesos de cifrado y descifrado mediante el método Afín. También, se muestran bien las indicaciones y los resultados por pantalla.

Para acabar con este apartado, mostramos otras capturas de ejecuciones con ficheros de entrada y de salida.

```

X541@LAPTOP-DanMat27 /c/Users/x541/desktop/cripto/p1
$ ./afin -C -m 26 -a 15 -b 3 -i entrada_cifrar.txt
> Numeros introducidos:
m | a : 26 | 15

> Desarrollo del algoritmo de euclides:
26 = 1 x 15 + 11
15 = 1 x 11 + 4
11 = 2 x 4 + 3
4 = 1 x 3 + 1
3 = 3 x 1 + 0

mcd(26,15) = 1 => SI SON COPRIMOS

> Se va a proceder a hallar el inverso multiplicativo modular por el algoritmo de euclides extendido:
3 = 11 - 2 x 4
4 = 15 - 1 x 11
11 = 26 - 1 x 15

1 = 7*15 mod 26
Inverso de 15 en Z_26 es = 7

# Ha elegido cifrar #

Ciframos el fichero de entrada: entrada_cifrar.txt

Texto cifrado:
QF WSLBFN DULPDYQFN CDQCF D QRLNCYDN HYLDHTFQLN.
CFWDN MDN HFNDN HYLDWDN LNCDQ WLNCCTQDWLN D YFBULYNL.
MFN DYCTNCDN WSLBFN LQHFQCYDY DMLPYTD... LQ LM UYFHNF WL MD HYLDHTFQ.

```

```

X541@LAPTOP-DanMat27 /c/Users/x541/desktop/cripto/p1
$ ./afin -C -m 26 -a 15 -b 3 -i entrada_cifrar.txt -o entrada_descifrar.txt
> Numeros introducidos:
m | a : 26 | 15

> Desarrollo del algoritmo de euclides:
26 = 1 x 15 + 11
15 = 1 x 11 + 4
11 = 2 x 4 + 3
4 = 1 x 3 + 1
3 = 3 x 1 + 0

mcd(26,15) = 1 => SI SON COPRIMOS

> Se va a proceder a hallar el inverso multiplicativo modular por el algoritmo de euclides extendido:
3 = 11 - 2 x 4
4 = 15 - 1 x 11
11 = 26 - 1 x 15

1 = 7*15 mod 26
Inverso de 15 en Z_26 es = 7

# Ha elegido cifrar #

Ciframos el fichero de entrada: entrada_cifrar.txt
Guardamos el texto cifrado en el fichero de salida: entrada_descifrar.txt

```

```

X541@LAPTOP-DanMat27 /c/Users/x541/desktop/cripto/p1
$ ./afin -D -m 26 -a 15 -b 3 -i entrada_descifrar.txt
> Numeros introducidos:
m | a : 26 | 15

> Desarrollo del algoritmo de euclides:
26 = 1 x 15 + 11
15 = 1 x 11 + 4
11 = 2 x 4 + 3
4 = 1 x 3 + 1
3 = 3 x 1 + 0

mcd(26,15) = 1 => SI SON COPRIMOS

> Se va a proceder a hallar el inverso multiplicativo modular por el algoritmo de euclides extendido:
3 = 11 - 2 x 4
4 = 15 - 1 x 11
11 = 26 - 1 x 15

1 = 7*15 mod 26
Inverso de 15 en Z_26 es = 7

# Ha elegido descifrar #

Desciframos el fichero de entrada: entrada_descifrar.txt

Texto descifrado:
NO DEBEMOS APEGARNOS TANTO A NUESTRAS CREACIONES.
TODAS LAS COSAS CREADAS ESTAN DESTINADAS A ROMPERSE.
LOS ARTISTAS DEBEMOS ENCONTRAR ALEGRIA... EN EL PROCESO DE LA CREACION.

```

B. Criptoanálisis de Afín

Nuestra solución para aumentar la robustez y fortaleza del cifrado Afín ha sido aumentar el espacio de claves, aumentando el tamaño de la clave, que pasa a ser de un carácter a un bigrama (2 caracteres). Este aumento de robustez es mejor cuanto más grande sea el n en el n-grama pero nosotros los hemos tomado en bigramas.

Por lo tanto, la cardinalidad del espacio $|Z_{26}|$ pasa a ser $|Z_{26}^2| = |Z_{676}| = 676$
Y el número de claves pasa a ser $|K| = |Z_{26}|^*|Z_{26}| = 676^2 = 676 \cdot \phi(676)$

Con esto anterior, se puede ver que el número de claves es mucho más mayor que para claves de un solo carácter ($|K| = |Z_{26}|^*|Z_{26}| = 26^2 \cdot \phi(26)$), y hace que el método Afín mejorado por bigramas sea más robusto.

El código del Afín mejorado se encuentra en el módulo **afin_mejorado.c**. Este cifrado es igual al normal, pero el Alfabeto pasa a ser una matriz de bigramas (de AA a ZZ) donde la posición del bigrama en ella es el valor numérico al que equivale al hacer el cifrado/descifrado.

A continuación, la matriz del Alfabeto de bigramas impresa por pantalla, para que se haga una idea de su forma (los números indican los ejes x e y de la matriz):

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
0 AA AB AC AD AE AF AG AH AI AJ AK AL AM AN AO AP AQ AR AS AT AU AV AW AX AY AZ	1 BA BB BC BD BE BF BG BH BI BJ BK BL BM BN BO BP BQ BR BS BT BU BV BW BX BY BZ	2 CA CB CC CD CE CF CG CH CI CJ CK CL CM CN CO CP CQ CR CS CT CU CV CW CX CY CZ	3 DA DB DC DD DE DF DG DH DI DJ DK DL DM DN DO DP DQ DR DS DT DU DV DW DX DY DZ	4 EA EB EC ED EE EF EG EH EI EJ EK EL EM EN EO EP EQ ER ES ET EU EV EW EX EY EZ	5 FA FB FC FD FE FF FG FH FI FJ FK FL FM FN FO FP FQ FR FS FT FU FV FW FX FY FZ	6 GA GB GC GD GE GF GG GH GI GJ GK GL GM GN GO GP GQ GR GS GT GU GV GW GX GY GZ	7 HA HB HC HD HE HF HG HH HI HJ HK HL HM HN HO HP HQ HR HS HT HU HV HW HX HY HZ	8 IA IB IC ID IE IF IG IH II IJ IK IL IM IN IO IP IQ IR IS IT IU IV IW IX IY IZ	9 JA JB JC JD JE JF JG JH JI JJ JK JL JM JN JO JP JQ JR JS JT JU JV JW JX JY JZ	10 KA KB KC KD KE KF KG KH KI KJ KK KL KM KN KO KP KQ KR KS KT KU KV KW KX KY KZ	11 LA LB LC LD LE LF LG LM LI LJ LX LL LM LN LO LP LQ LR LS LT LU LV LW LX LY LZ	12 MA MB MC MD ME MF MG MH MI MJ MK ML MN MN MO MP MQ NR MS MT MU MV MW MX MY NZ	13 NA NB NC ND NE NF NG NH NI NJ NK NL NM NN NO NP NQ NR NS NT NU NV NW NX NY NZ	14 OA OB OC OD OE OF OG OH OI OJ OK OL OM ON OO OP OQ OR OS OT OU OV OW OW OX OY OZ	15 PA PB PC PD PE PF PG PH PI PJ PK PL PM PN PO PP PQ PR PS PT PU PV PW PX PY PZ	16 QA QB QC QD QE QF QG QH QI QJ QK QL QM QN QO QP QQ QR QS QT QU QV QW QX QY QZ	17 RA RB RC RD RE RF RG RH RI RJ RK RL RM RN RO RP RQ RR RS RT RU RV RW RX RY RZ	18 SA SB SC SD SE SF SG SH SI SJ SK SL SM SN SO SP SQ SR SS ST SU SV SW SX SY SZ	19 TA TB TC TD TE TF TG TH TI TJ TK TL TM TN TO TP TQ TR TS TT TU TV TW TX TY TZ	20 UA UB UC UD UE UF UG UM UI UJ UK UL UM UN UO UP UQ UR US UT UU UV UW UX UY UZ	21 VA VB VC VD VE VF VG VH VI VJ VK VL VM VN VO VP VQ VR VS VT VU VV VW VX VY VZ	22 WA WB WC WD WE WF WG WH WI WJ WK WL WM WN WO WP WQ WR WS WT WU WV MW NX WY WZ	23 XA XB XC XD XE XF XG XM XI XJ XK XL XM XN XO XP XQ XR XS XT XU XV XW XX XY XZ	24 YA YB YC YD YE YF YG YH YI YJ YK YL YM YN YO YP YQ YR YS YT YU YV YW YX YY YZ	25 ZA ZB ZC ZD ZE ZF ZG ZH ZI ZJ ZK ZL ZM ZN ZO ZP ZQ ZR ZS ZT ZU ZV ZW ZX ZY ZZ

Y, con este Alfabeto, hemos realizado el cifrado y el descifrado y vamos a mostrar algunas capturas de su buen funcionamiento:

```
X541@LAPTOP-DanMat27 /c/users/x541/desktop/cripto/p1
$ ./afin_mejorado -C -m 676 -a 15 -b 9 -i entrada_cifrar.txt
> Numeros introducidos:
m | a : 676 | 15

> Desarrollo del algoritmo de euclides:
676 = 45 x 15 + 1
15 = 15 x 1 + 0

mcd(676,15) = 1 => SI SON COPRIMOS

> Se va a proceder a hallar el inverso multiplicativo modular por el algoritmo de euclides extendido:
1 = -45*15 mod 676
Inverso de 15 en Z_676 es = 631

# Ha elegido cifrar #

Ciframos el fichero de entrada: entrada_cifrar.txt

Texto cifrado:
VL VRRRGGLK JTRMJCWMT ZJYIC J YXSTJEKT OEIJIZJWST.
HLTJQ SKT MLKJL NXRCCKT STZJP CSTDZNJTJK J DLHASEMR.
RLK JGIATZJM CIYPHMHT PWHLYIVJVJLNRNEQJ... PW OS BEDNSTE CO SB NXRBNYLN.
```

```
X541@LAPTOP-DanMat27 /c/users/x541/desktop/cripto/p1
$ ./afin_mejorado -D -m 676 -a 15 -b 9 -i entrada_descifrar.txt
> Numeros introducidos:
m | a : 676 | 15

> Desarrollo del algoritmo de euclides:
676 = 45 x 15 + 1
15 = 15 x 1 + 0

mcd(676,15) = 1 => SI SON COPRIMOS

> Se va a proceder a hallar el inverso multiplicativo modular por el algoritmo de euclides extendido:
1 = -45*15 mod 676
Inverso de 15 en Z_676 es = 631

# Ha elegido descifrar #

Desciframos el fichero de entrada: entrada_descifrar.txt

Texto descifrado:
NO DEBEMOS APEGARNOS TANTO A NUESTRAS CREACIONES.
TODAS LAS COSAS CREADAS ESTAN DESTINADAS A ROMPERSE.
LOS ARTISTAS DEBEMOS ENCONTRAR ALEGRIA... EN EL PROCESO DE LA CREACION.
```

```

X541@LAPTOP-DanMat27 /c/users/x541/desktop/cripto/p1
$ ./afin_mejorado -C -m 676 -a 93 -b 4
> Numeros introducidos:
m | a : 676 | 93

> Desarrollo del algoritmo de euclides:
676 = 7 x 93 + 25
93 = 3 x 25 + 18
25 = 1 x 18 + 7
18 = 2 x 7 + 4
7 = 1 x 4 + 3
4 = 1 x 3 + 1
3 = 3 x 1 + 0

mcd(676,93) = 1 => SI SON COPRIMOS

> Se va a proceder a hallar el inverso multiplicativo modular por el algoritmo de euclides extendido:
3 = 7 - 1 x 4
4 = 18 - 2 x 7
7 = 25 - 1 x 18
18 = 93 - 3 x 25
25 = 676 - 7 x 93

1 = 189*93 mod 676
Inverso de 93 en Z_676 es = 189

# Ha elegido cifrar #

Introduzca lo que quiere cifrar:
HOLA SOMOS DANIEL Y LAURA

Cifrando HOLA SOMOS DANIEL Y LAURA
Texto cifrado: ZGJE IGNGU XUREMR A JENZA

```

```

X541@LAPTOP-DanMat27 /c/users/x541/desktop/cripto/p1
$ ./afin_mejorado -D -m 676 -a 93 -b 4
> Numeros introducidos:
m | a : 676 | 93

> Desarrollo del algoritmo de euclides:
676 = 7 x 93 + 25
93 = 3 x 25 + 18
25 = 1 x 18 + 7
18 = 2 x 7 + 4
7 = 1 x 4 + 3
4 = 1 x 3 + 1
3 = 3 x 1 + 0

mcd(676,93) = 1 => SI SON COPRIMOS

> Se va a proceder a hallar el inverso multiplicativo modular por el algoritmo de euclides extendido:
3 = 7 - 1 x 4
4 = 18 - 2 x 7
7 = 25 - 1 x 18
18 = 93 - 3 x 25
25 = 676 - 7 x 93

1 = 189*93 mod 676
Inverso de 93 en Z_676 es = 189

# Ha elegido descifrar #

Introduzca lo que quiere descifrar:
ZGJE IGNGU XUREMR A JENZA

Descifrando ZGJE IGNGU XUREMR A JENZA
Texto descifrado: HOLA SOMOS DANIEL Y LAURA

```

A continuación, vamos a hablar del método para criptoanalizar esta versión mejorada del Afín. Este método consiste en el mismo criptoanálisis que se realizaría para el Cifrado Afín normal (de Alfabeto Ascii).

Lo que hay que hacer es, tras obtener un fragmento de un texto cifrado, contar las frecuencias de los bigramas y obtener los que más se repiten (mayor frecuencia). Una vez los tenemos, suponemos que esos dos bigramas con mayor frecuencia coinciden con los dos bigramas más frecuentes en el idioma inglés, que son TH y HE (los siguientes más frecuentes, por si falla el criptoanálisis con estos dos, son AN, ER, IN...).

Con esta suposición, establecemos un sistema de dos ecuaciones, donde decimos que TH, aplicando la clave, da como resultado el valor ($v(B_1)$) del bigrama más repetido en el cifrado. Y hacemos lo mismo con HE y el segundo bigrama.

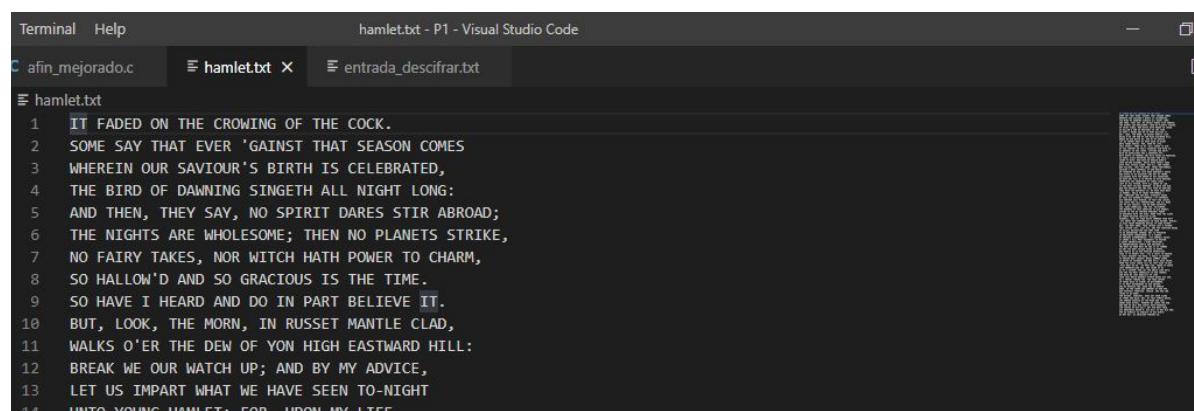
- $v(B_1) = v(TH)*a + b$
- $v(B_2) = v(HE)*a + b$

Ahora, hemos cifrado un fragmento de texto de la obra Hamlet (en inglés) con la clave $a=17$ y $b=5$. En el texto resultante cifrado de Hamlet hemos contado las frecuencias y nos ha dado que los bigramas más repetidos han sido PU y RV. Por lo tanto, para hallar la clave usada tenemos las siguientes ecuaciones donde los valores de los bigramas, como se ha dicho antes, son las posiciones en la matriz de bigramas:

$$\begin{aligned} - & v(PU) = v(TH)*a + y == 410 = 501*a + b \\ - & v(RV) = v(HE)*a + y == 463 = 186*a + b \\ -53 & = 315*a == 623 = 315*a == 623*(315^{-1}) = a \\ \mathbf{a} & = 623*191 \bmod 676 = 118993 \bmod 676 = \mathbf{17} \\ \mathbf{b} & = 410 - 501*17 = -8107 \bmod 676 = \mathbf{5} \end{aligned}$$

Como vemos, hemos tenido la suerte de que han coincidido bien los bigramas con más frecuencia en el texto cifrado con las frecuencias generales del idioma inglés, por lo que hemos obtenido la clave original y válida que es $k = (a=17, b=5)$. Si no hubiera sido así, hubiésemos cambiado los bigramas en el sistema de ecuaciones o probado con los siguientes de la lista de frecuencias. Como consecuencia, el descifrado del texto cifrado con esta clave da como resultado el texto original de Hamlet.

Texto plano original:



```

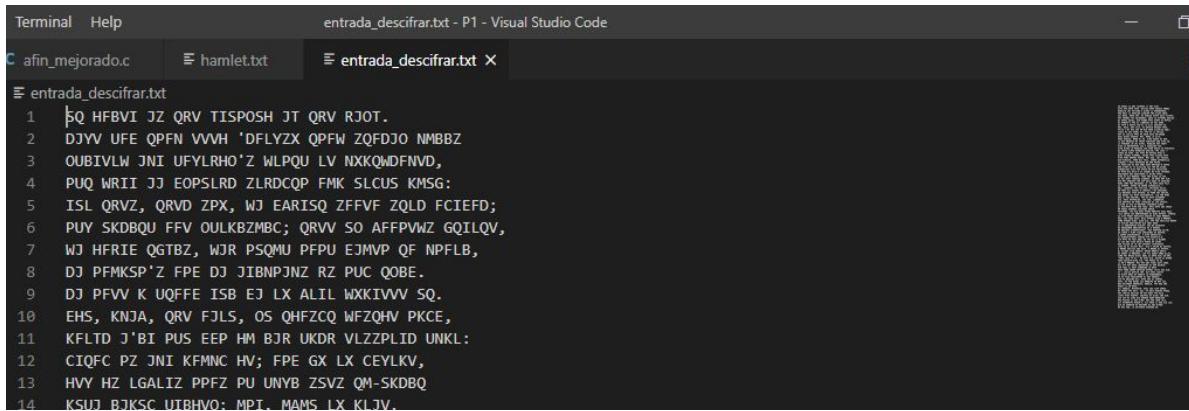
Terminal Help
hamlet.txt - P1 - Visual Studio Code

afin_mejorado.c    hamlet.txt x  entrada_descifrar.txt

hamlet.txt
1 IT FADED ON THE CROWING OF THE COCK.
2 SOME SAY THAT EVER 'GAINST THAT SEASON COMES
3 WHEREIN OUR SAVIOUR'S BIRTH IS CELEBRATED,
4 THE BIRD OF DAWNING SINGETH ALL NIGHT LONG:
5 AND THEN, THEY SAY, NO SPIRIT DARES STIR ABROAD;
6 THE NIGHTS ARE WHOLESOME; THEN NO PLANETS STRIKE,
7 NO FAIRY TAKES, NOR WITCH HATH POWER TO CHARM,
8 SO HALLOW'D AND SO GRACIOUS IS THE TIME.
9 SO HAVE I HEARD AND DO IN PART BELIEVE IT.
10 BUT, LOOK, THE MORN, IN RUSSET MANTLE CLAD,
11 WALKS O'ER THE DEW OF YON HIGH EASTWARD HILL:
12 BREAK WE OUR WATCH UP; AND BY MY ADVICE,
13 LET US IMPART WHAT WE HAVE SEEN TO-NIGHT
14 UNTO YOUNG HAMLET: FOR UPON MY LIFE

```

Texto cifrado utilizado en el criptoanálisis:



```
Terminal Help entra_descifrar.txt - P1 - Visual Studio Code
afin_mejorado.c hamlet.txt entra_descifrar.txt
entra_descifrar.txt
1  $Q HFBVI JZ QRV TISPOSH JT QRV RJOT.
2  DJV UFE QPFN VVH 'DFLYZX QPFW ZQFDJO NMBBZ
3  OUBIVLW JNI UFYLRHO'Z WLPQU LV NXKQWDFNVD,
4  PUQ WRII JJ EOPSLRD ZLRCQF FMK SLCUS KMSG:
5  ISL QRVZ, QRVD ZPX, WJ EARISQ ZFFVF ZQLD FCIEFD;
6  PUY SKDBQU FFV OULKBMBC; QRVV SO AFFPVWZ GQILQV,
7  WJ HFRIE QGTBZ, WJR PSQMU PFFU EJMPV QF NPFLB,
8  DJ PFMKSP'Z FPE DJ JIBNPJNZ RZ PUC QOBE.
9  DJ PFVV K UQFFE ISR EJ LX ALIL WKXIVV SQ.
10 EHS, KNJA, QRV FJLS, OS QHFZCQ WFZQHV PKCE,
11 KFLTD J'BI PUS EEP HM BJR UKDR VLZZPLID UNKL:
12 CIQFC PZ JNI KFMNC HV; FPE GX LX CEYLKV,
13 HVY HZ LGALIZ PPFZ PU UNYB ZSVZ QM-SKDBQ
14 KSUJ BJKSC UTRHVO: MPT, MAMS IX KJYV.
```

Proceso de cifrado anterior al criptoanálisis, de donde obtuvimos el texto cifrado:

```
X541@LAPTOP-DanMat27 /c/users/x541/desktop/cripto/p1
$ ./afin_mejorado -C -m 676 -a 17 -b 5 -i hamlet.txt -o entrada_descifrar.txt
> Numeros introducidos:
m | a : 676 | 17

> Desarrollo del algoritmo de euclides:
676 = 39 x 17 + 13
17 = 1 x 13 + 4
13 = 3 x 4 + 1
4 = 4 x 1 + 0

mcd(676,17) = 1 => SI SON COPRIMOS

> Se va a proceder a hallar el inverso multiplicativo modular por el algoritmo de euclides extendido:
4 = 17 - 1 x 13
13 = 676 - 39 x 17

1 = -159*17 mod 676
Inverso de 17 en Z_676 es = 517

# Ha elegido cifrar #

Ciframos el fichero de entrada: hamlet.txt
Guardamos el texto cifrado en el fichero de salida: entrada_descifrar.txt
```

2. Sustitución polialfabeto

A. Método de Vigenere

El método vigenere está implementado en los ficheros **vigenere.c** y **vigenere.h**.

El programa vigenere, como pone en el enunciado, acepta los siguientes parámetros:

`./vigenere {-C|-D} {-k clave} [-i filein] [-o fileout]`

Obligatorios:

`{-C|-D}` → -C para cifrar y -D para descifrar

`{-k clave}` → la clave puede tener cualquier tamaño (n) y debe estar formada por caracteres que formen parte del alfabeto (65 - 90 en ASCII).

Opcionales:

[-i file] → fichero de entrada con el texto plano (-C) o el texto cifrado (-D).

[-o file] → fichero de salida donde escribir el texto cifrado (-C) o el texto plano (-D).

Ejemplo de ejecución:

```
laura@Universidad:~/Escritorio/cripto/P1$ ./vigenere -C -k HOLA
# Ha elegido cifrar #

Introduzca lo que quiere cifrar:
HOLA PROFESOR

Cifrando HOLA PROFESOR
Texto cifrado: OCWA WFZFLGZR

laura@Universidad:~/Escritorio/cripto/P1$ ./vigenere -D -k HOLA
# Ha elegido descifrar #

Introduzca lo que quiere descifrar:
OCWA WFZFLGZR

Descifrando OCWA WFZFLGZR
Texto descifrado: HOLA PROFESOR
```

Primero comprobamos que no haya errores en los argumentos, y si los hay informamos sobre ello. Para ello, aparte de contar que estén el mínimo número de argumentos requeridos en este caso 4, utilizamos la función `read_args_vigenere` la cual lee los argumentos pasados por la terminal, informa de los diferentes errores posibles y devuelve el modo. El modo puede tomar los valores **TECLADO**, **IFILE**, **OFILE** o **IOFILES** dependiendo de si se ha introducido fichero de entrada (ifile), de salida (ofile), ambos ficheros (iofiles) o ninguno (teclado).

Una vez leídos los argumentos pasados por la terminal ya sabemos si hay que cifrar o descifrar y el modo. Tanto para cifrar y descifrar en el caso de que el modo sea **TECLADO** pedimos por terminal el texto a cifrar o descifrar. Si el modo es **IFILE** leemos el fichero de entrada y ciframos o desciframos su contenido mostrando por terminal el resultado. Por el contrario, si se trata de **OFILE** pedimos por terminal el texto a cifrar o descifrar y escribimos el resultado en el fichero de salida. Y por último si el modo es **IOFILES** leemos el fichero de entrada y ciframos o desciframos su contenido y escribimos el resultado en el fichero de salida.

Las funciones de cifrar y descifrar las hemos implementado aparte, solo cifra/descifra caracteres que pertenecen al alfabeto (65 - 90 en ASCII).

Cifrado → void `cifrar(char *texto, char *k)`

Para cifrar vamos recorriendo el `texto` y cuando se trata de un carácter que pertenece al alfabeto lo ciframos. Para cifrarlo, primero restamos 65 (valor de A en ASCII) al carácter para sacar su valor en Z26, realizamos lo mismo con el carácter de la clave que toque. Una vez obtenidos estos valores los sumamos y hacemos el

módulo m, en nuestro caso m es igual a 26, y al resultado le sumamos 65 para obtener el valor del nuevo carácter en ASCII.

```
valor_caracter = texto[cont] - 65;
valor_clave = k[b] - 65;
valor_nuevo_caracter = valor_caracter + valor_clave;

if (valor_nuevo_caracter < 0){
    aux = valor_nuevo_caracter + 26;
} else{
    aux = valor_nuevo_caracter % 26;
}

texto[cont] = aux + 65;
```

Según vamos obteniendo los caracteres cifrados los vamos sustituyendo en *texto* y actualizando la posición de la clave, sumando uno o volviendo a la posición 0 en el caso de haber completado todos los caracteres de la clave.

Ejemplo cifrando un texto pequeño y clave CRIPTOGRAFIA:

```
vigenere.c           entrada_cifrar.txt
LAURA UNO UNA DE LAS POSIBLES FUNCIONES QUE TENEMOS PARA LEER UN FICHERO DE
TEXTO ES FGETS ESTA FUNCION LEE UNA LINEA COMPLETA DEL FICHERO DE TEXTO Y NOS LA
DEVELVE TIENE LOS SIGUIENTES PARAMETROS VAMOS QUE ES UN TEXTO NARRATIVO CUAL
ES SU DEFINICION Y LAS CARACTERISTICAS QUE LO DIFERENCIAN DE OTROS COMO PUEDEN
SER LOS DESCRIPTIVOS POR EJEMPLO RECORDREMOS QUE UN TEXTO SEA DEL TIPO QUE SEA
ES UN CONJUNTO DE ENUNCIADOS QUE UNIDOS MANTIENEN UNA COHERENCIA Y UNA UNIDAD
QUE COBRA SENTIDO ADEMÁS LA INTENCION DE ESTAS REDACCIONES HA DE SER COMUNICATIVA
EN EL CASO QUE AQUÍ NOS OCUPA TRAS DEFINIR EL TEXTO TAMBIEN INTERESARIA SABER EXACTAMENTE
EN QUE CONSISTE EL ACTO DE NARRAR PODEMOS REFERIRNOS A EL COMO LA INTENCION DE
CONTAR UNA HISTORIA YA SEA FICTICIA O VERIDICA. AHORA QUE YA CONOCEMOS LOS DOS
COMPONENTES BASICOS DEL TEXTO NARRATIVO VAMOS A ARRANCAR CON ESTA LECCION DE
UNPROFESOR PARA SEGUIR PROFUNDIZANDO SOBRE LA DEFINICION Y CARACTERISTICAS DEL
TEXTO NARRATIVO.LINEA COMPLETA DEL FICHERO DE TEXTO Y NOS LA DEVUELVE.
TIENE LOS SIGUIENTES PARAMETROS
```

```
laura@Universidad:~/Escritorio/cripto/P1$ ./vigenere -C -k CRIPTOGRAFIA -i entrada_cifrar.txt
# Ha elegido cifrar #

Ciframos el fichero de entrada: entrada_cifrar.txt

Texto cifrado:
NRCGT ICL USI DG CIH ICYZBQMS HLVRBCTVS VCE VVVTFCY GANI LGVZ JG TOTHJZO FV
BTQHU VS KOEVJ MHMO LLNHQOP CMT NBG CISMA EFUEESZR DJT FKTPTKC JV TJFTQ P VDL ZG
UEACENMM IBSTV LTA SKXXXBZVS UIRCQMIKY MEFUOU HCT XG AE TJFTQ EIGKOZZVT KUCC
MH LI JVFNVIEZWC R ZGJ CFZAEMGBZZCFA QWV TD WWLVRJVCKRV SX CZIOX KOOF XJRKE
S1Z LQJ LTQXZPYQVQJ XKD SPVMUTO TVKDKRKDOX YUG LV IXLZF SJI DGC BXIC WLE XMA
GJ CC VCTAUSBO FV MCNBZAIWS SLM JGWJFS RINVZMCXB AEA HWHGIMCVNG P USI UPZLPW
EAV CTJRC JMCMWJF AIMMCJ TP BBZVNHOOP UM TLHGJ RJLAETQDGSY YA IM SG1 KDFITZCFBIXR
MC XZ IRST YUG RYJB BUJ OHCPK KZPL RKWISQR GC BTQHU KARJIGE QCMSXVSFZIC JIQXF KOAHBAOVIX
ST HUJ KOPJQHMS KC AHBO FV VPKFGI PTLEOFA GXTKIIWVOU R MA VCSF LF QNVVVRBCT UE
HMNVRZ JGO NZSYWRKR GP LSG WIHBIEZI D OSXZDNKA. CYWT EAV YF KOPFKTFCY COX LOU
TWBICTVNYMS DRAXVCY UEQ BEZKW CTFXRTNDO XRUDL O GIRFVCC1 KDG SYKA QMCEZWC WS
AEPWWFGJWG IOXR SJOUKI XGHTAEONHPAUW HHXPV LF LEHZVXWUE Y HIRCTBTWKYKIHIS FVT
IXLZF NFZRCKQKH.ZOEEF KOOGTTMO JVL KQCJVD WS ZVXYN Y PFA AT RKMUJTVG.
KOTGS RFS XQGWZMCMY GAWIMGKZDL
```

Lo ejecutamos también de esta manera para luego descifrarlo y comprobar que coincide con el texto plano original.

```
laura@Universidad:~/Escritorio/cripto/P1$ ./vigenere -C -k CRIPTOGRAFIA -i entrada_cifrar.txt -o salida_cifrar_texto_pequeño.txt
# Ha elegido cifrar #

Ciframos el fichero de entrada: entrada_cifrar.txt
Guardamos el texto cifrado en el fichero de salida: salida_cifrar_texto_pequeño.txt
```

Ejemplo cifrando un texto grande y clave UAM:

vigenere.c don_quixote.txt

1 CHAPTER I
2 WHICH TREATS OF THE CHARACTER AND
3 PURSUITS OF THE FAMOUS GENTLEMAN DON
4 QUIXOTE OF LA MANCHA
5 IN A VILLAGE OF LA MANCHA, THE NAME OF WHICH I HAVE NO DESIRE TO CALL TO
6 MIND, THERE LIVED NOT LONG SINCE ONE OF THOSE GENTLEMEN THAT KEEP A LANCE IN THE
7 LANCE-RACK, AN OLD BUCKLER, A LEAN HACK, AND A GREYHOUND FOR COURSING. AN OLLA OF
8 RATHER MORE BEEF THAN MUTTON, A SALAD ON MOST NIGHTS, SCRAPS ON SATURDAYS, LENTILS
9 ON FRIDAYS, AND A PIGEON OR SO EXTRA ON SUNDAYS, MADE AWAY WITH THREE-QUARTERS
10 OF HIS INCOME. THE REST OF IT WENT IN A DOUBLET OF FINE CLOTH AND VELVET BREECHES
11 AND SHOES TO MATCH FOR HOLIDAYS, WHILE ON WEEK-DAYS HE MADE A BRAVE FIGURE IN
12 HIS BEST HOMESPUN. HE HAD IN HIS HOUSE A HOUSEKEEPER PAST FORTY, A NIECE UNDER
13 TWENTY, AND A LAD FOR THE FIELD AND MARKET-PLACE, WHO USED TO SADDLE THE HACK AS
14 WELL AS HANDLE THE BILL-HOOK. THE AGE OF THIS GENTLEMAN OF OURS WAS BORDERING ON
15 FIFTY; HE WAS OF A HARDY HABIT, SPARE, GAUNT-FEATURED, A VERY EARLY RISER AND A GREAT
16 SPORTSMAN. THEY WILL HAVE IT HIS SURNAME WAS QUIXADA OR QUESADA (FOR HERE THERE
17 IS SOME DIFFERENCE OF OPINION AMONG THE AUTHORS WHO WRITE ON THE SUBJECT), ALTHOUGH FROM REASONABLE CONJECTURES IT SEEMS PLAIN THAT HE WA
18 THIS, HOWEVER, IS OF BUT LITTLE IMPORTANCE TO OUR TALE; IT WILL BE ENOUGH NOT TO STRAY
19 A HAIR'S BREADTH FROM THE TRUTH IN THE TELLING OF IT.
20 YOU MUST KNOW, THEN, THAT THE ABOVE-NAMED GENTLEMAN WHENEVER HE WAS AT
21 LEISURE (WHICH WAS MOSTLY ALL THE YEAR ROUND) GAVE HIMSELF UP TO READING BOOKS OF
22 CHIVALRY WITH SUCH ARDOUR AND AVIDITY THAT HE ALMOST ENTIRELY NEGLECTED THE PURSUIT
23 OF HIS FIELD-SPORTS, AND EVEN THE MANAGEMENT OF HIS PROPERTY; AND TO SUCH A PITCH
24 DID HIS EAGERNESS AND INFATUATION GO THAT HE SOLD MANY AN ACRE OF TILLAGELAND TO
25 BUY BOOKS OF CHIVALRY TO READ, AND BROUGHT HOME AS MANY OF THEM AS HE COULD
26 GET, BUT OF ALL THERE WERE NONE HE LIKED SO WELL AS THOSE OF THE FAMOUS FELICIANO
27 DE SILVA'S COMPOSITION, FOR THEIR LUCIDITY OF STYLE AND COMPLICATED CONCEITS WERE
28 AS PEARLS IN HIS SIGHT, PARTICULARLY WHEN IN HIS READING HE CAME UPON COURTSHIPS
29 AND CARTELS, WHERE HE OFTEN FOUND PASSAGES LIKE "THE REASON OF THE UNREASON WITH
30 WHICH MY REASON IS AFFLICTED SO WEAKENS MY REASON THAT WITH REASON I MURMUR AT
31 YOUR BEAUTY;" OR AGAIN, "THE HIGH HEAVENS, THAT OF YOUR DIVINITY DIVINELY FORTIFY
32 YOU WITH THE STARS. RENDER YOU DESERVING OF THE DESERT YOUR GREATNESS DESERVES."

198
199 -TY PRESENTED ITSELF, FOR IT SEEMED TO HIM IMPOSSIBLE TO RELIEVE HIMSELF WITHOUT MAKING SOME NOISE, AND HE GROUND HIS TEETH AND SQUEEZED
200
201
202
203
204
205 JRTER NOISE OR DISTURBANCE HE FOUND HIMSELF RELIEVED OF THE BURDEN THAT HAD GIVEN HIM SO MUCH DISCOMFORT. BUT AS DON
206
207
208 HIS FINGERS, SAYING IN A RATHER SNUFFING TONE, "SANCHO, IT STRIKES ME THOU ART IN GREAT FEAR."
209

Descifrado → void descifrar(char* texto, char *k)

El descifrado es igual que el cifrado únicamente se diferencian en que ahora al valor del carácter se le resta el valor del carácter de la clave.

```
valor_caracter = texto[cont] - 65;
valor_clave = k[b] - 65;
valor_nuevo_caracter = valor_caracter - valor_clave;

if (valor_nuevo_caracter < 0){
    aux = valor_nuevo_caracter + 26;
} else{
    aux = valor_nuevo_caracter % 26;
}

texto[cont] = aux + 65;
```

Vamos a comprobar que se descifran correctamente los textos que hemos cifrado anteriormente.

Ejemplo descifrado texto pequeño y clave CRIPTOGRAFIA:

```
laura@Universidad:~/Escritorio/cripto/P1$ ./vigenere -D -k CRIPTOGRAFIA -i salida_cifrar_texto_pequeño.txt
# Ha elegido descifrar #

Desciframos el fichero de entrada: salida_cifrar_texto_pequeño.txt

Texto descifrado:
LAURA UWU UNA DE LAS POSIBLES FUNCIONES QUE TENEMOS PARA LEER UN FICHERO DE TEXTO ES FGETS ESTA FUNCION LEE UNA LINEA COMPLETA DEL FICHERO DE TEXTO Y NOS LA DEVUELVE TIENE LOS SIGUIENTES PARAMETROS VEAMOS QUE ES UN TEXTO NARRATIVO CUAL ES SU DEFINICION Y LAS CARACTERISTICAS QUE LO DIFERENCIAN DE OTROS COMO PUEDEN SER LOS DESCRIPTIVOS POR EJEMPLO RECORDEMOS QUE UN TEXTO SEA DEL TIPO QUE SEA ES UN CONJUNTO DE ENUNCIADOS QUE UNIDOS MANTIENEN UNA COHERENCIA Y UNA UNIDAD QUE COBRA SENTIDO ADEMÁS LA INTENCIÓN DE ESTAS REDACCIONES HA DE SER COMUNICATIVA EN EL CASO QUE AQUÍ NOS OCUPA TRAS DEFINIR EL TEXTO TAMBIÉN INTERESARIA SABER EXACTAMENTE EN QUÉ CONSISTE EL ACTO DE NARRAR PODEMOS REFERIRNOS A EL COMO LA INTENCIÓN DE CONTAR UNA HISTORIA YA SEA FICTICIA O VERIDICA. AHORA QUE YA CONOCEMOS LOS DOS COMPONENTES BASICOS DEL TEXTO NARRATIVO VAMOS A ARRANCAR CON ESTA LECCIÓN DE UNPROFESOR PARA SEGUIR PROFUNDIZANDO SOBRE LA DEFINICION Y CARACTERISTICAS DEL TEXTO NARRATIVO.LINEA COMPLETA DEL FICHERO DE TEXTO Y NOS LA DEVUELVE.

TIENE LOS SIGUIENTES PARAMETROS
```

Coincide con el texto plano original (entrada_cifrar.txt)

Ejemplo descifrado texto grande y clave UAM:

```
laura@Universidad:~/Escritorio/cripto/P1$ ./vigenere -D -k UAM -i salida_cifrar_texto_grande.txt -o descifrado_texto_grande.txt
# Ha elegido descifrar #

Desciframos el fichero de entrada: salida_cifrar_texto_grande.txt
Guardamos el texto descifrado en el fichero de salida: descifrado_texto_grande.txt
laura@Universidad:~/Escritorio/cripto/P1$
```

descifrado_texto_grande.txt

CHAPTER I
WHICH TREATS OF THE CHARACTER AND
PURSUITS OF THE FAMOUS GENTLEMAN DON
QUIXOTE OF LA MANCHA
IN A VILLAGE OF LA MANCHA, THE NAME OF WHICH I HAVE NO DESIRE TO CALL TO
MIND, THERE LIVED NOT LONG SINCE ONE OF THOSE GENTLEMEN THAT KEEP A LANCE IN THE
LANCE-RACK, AN OLD BUCKLER, A LEAN HACK, AND A GREYHOUND FOR COURSING. AN OLLA OF
RATHER MORE BEEF THAN MUTTON, A SALAD ON MOST NIGHTS, SCRAPS ON SATURDAYS, LENTILS
ON FRIDAYS, AND A PIGEON OR SO EXTRA ON SUNDAYS, MADE AWAY WITH THREE-QUARTERS
OF HIS INCOME. THE REST OF IT WENT IN A DOUBLET OF FINE CLOTH AND VELVET BREECHES
AND SHOES TO MATCH FOR HOLIDAYS, WHILE ON WEEK-DAYS HE MADE A BRAVE FIGURE IN
HIS BEST HOMESPUN. HE HAD IN HIS HOUSE A HOUSEKEEPER PAST FORTY, A NIECE UNDER
TWENTY, AND A LAD FOR THE FIELD AND MARKET-PLACE, WHO USED TO SADDLE THE HACK AS
WELL AS HANDLE THE BILL-HOOK. THE AGE OF THIS GENTLEMAN OF OURS WAS BORDERING ON
FIFTY; HE WAS OF A HARDY HABIT, SPARE, GAUNT-FEATURED, A VERY EARLY RISER AND A GREAT
SPORTSMAN. THEY WILL HAVE IT HIS SURNAME WAS QUIXADA OR QUESADA (FOR HERE THERE
IS SOME DIFFERENCE OF OPINION AMONG THE AUTHORS WHO WRITE ON THE SUBJECT), ALTHOUGH FROM REASONABLE CONJECTURES IT SEEMS PLAIN THAT HE WAS CALLED QUESANA.
THIS, HOWEVER, IS OF BUT LITTLE IMPORTANCE TO OUR TALE; IT WILL BE ENOUGH NOT TO STRAY
A HAIR'S BREADTH FROM THE TRUTH IN THE TELLING OF IT.
YOU MUST KNOW, THEN, THAT THE ABOVE-NAMED GENTLEMAN WHENEVER HE WAS AT
LEISURE (WHICH WAS MOSTLY ALL THE YEAR ROUND) GAVE HIMSELF UP TO READING BOOKS OF
CHIVALRY WITH SUCH ARDOUR AND AVIDITY THAT HE ALMOST ENTIRELY NEGLECTED THE PURSUIT
OF HIS FIELD-SPORTS, AND EVEN THE MANAGEMENT OF HIS PROPERTY; AND TO SUCH A PITCH
DID HIS EAGERNESS AND INFATUATION GO THAT HE SOLD MANY AN ACRE OF TILLAGELAND TO
BUY BOOKS OF CHIVALRY TO READ, AND BROUGHT HOME AS MANY OF THEM AS HE COULD
GET. BUT OF ALL THERE WERE NONE HE LIKED SO WELL AS THOSE OF THE FAMOUS FELICIANO
DE SILVA'S COMPOSITION, FOR THEIR LUCIDITY OF STYLE AND COMPLICATED CONCEITS WERE
AS PEARLS IN HIS SIGHT, PARTICULARLY WHEN IN HIS READING HE CAME UPON COURTSHIPS
AND CARTELS, WHERE HE OFTEN FOUND PASSAGES LIKE "THE REASON OF THE UNREASON WITH
WHICH MY REASON IS AFFLICTED SO WEAKENS MY REASON THAT WITH REASON I MURMUR AT
YOUR BEAUTY;" OR AGAIN, "THE HIGH HEAVENS, THAT OF YOUR DIVINITY DIVINELY FORTIFY
YOU WITH THE STARS, RENDER YOU DESERVING OF THE DESERT YOUR GREATNESS DESERVES."

INT, ANOTHER STILL GREATER DIFFICULTY PRESENTED ITSELF, FOR IT SEEMED TO HIM IMPOSSIBLE TO RELIEVE HIMSELF WITHOUT MAKING SOME NOISE, AND HE GROUNDED HIS TEETH AND SQUEEZED
HIS
ENT
EEDED SO WELL, THAT WITHOUT ANY FURTHER NOISE OR DISTURBANCE HE FOUND HIMSELF RELIEVED OF THE BURDEN THAT HAD GIVEN HIM SO MUCH DISCOMFORT. BUT AS DON
NOT
RELIEF BY COMPRESSING IT BETWEEN HIS FINGERS, SAYING IN A RATHER SNUFFING TONE, "SANCHO, IT STRIKES ME THOU ART IN GREAT FEAR."

```
laura@Universidad:~/Escritorio/cripto/P1$ diff don_quixote.txt descifrado_texto_grande.txt
laura@Universidad:~/Escritorio/cripto/P1$
```

De nuevo se descifra correctamente, es igual que el texto plano (don_quixote.txt).

B. Criptoanálisis del Vigenere

Para poder criptoanalizar documentos cifrados mediante el método de Vigenere primero necesitamos obtener la longitud de la clave. Esta la podemos averiguar de dos formas diferentes:

Test de Kasiski

La primera forma es con el Test de Kasiski, el cual es capaz de obtener la longitud de la clave (n) con la que se ha cifrado el texto buscando repeticiones de un conjunto de caracteres y midiendo la distancia entre ellas.

Está implementado en los ficheros **kasiski.c** y **kasiski.h**.

El programa kasiski acepta los siguientes parámetros:

`./kasiski {-l tamRepeticiones} [-i filein]`

Obligatorio:

`{-l tamRepeticiones}` → indica el tamaño de las repeticiones, es decir, cuantos caracteres tiene la repetición a buscar.

Opcional:

`[-i filein]` → fichero de entrada con el texto cifrado por el método vigenere.

Ejemplo de ejecución:

```
laura@Universidad:~/Escritorio/cripto/P1$ ./vigenere -C -k LAURA
# Ha elegido cifrar #

Introduzca lo que quiere cifrar:
HOLA COMO ESTAS LA HOLA ALBERTO CRIPTOGRAFIA TODO BIEN WOUU HOLA

Cifrando HOLA COMO ESTAS LA HOLA ALBERTO CRIPTOGRAFIA TODO BIEN WOUU HOLA
Texto cifrado: SOFR CZMI VSEAM CA SOFR AWBYITZ CLZPEOAIAQIU KOOO VZEY WILU SOFR

laura@Universidad:~/Escritorio/cripto/P1$ ./kasiski -l 3
Introduce el texto cifrado por el metodo vigenere del cual obtendremos la longitud de la clave
SOFR CZMI VSEAM CA SOFR AWBYITZ CLZPEOAIAQIU KOOO VZEY WILU SOFR

La repetition buscada es: SOF
Posicion 15
Posicion 50
mcd( 15  50 ) = 5
n puede ser 5
n puede ser 1
laura@Universidad:~/Escritorio/cripto/P1$
```

En el anterior ejemplo podemos observar como kasiski obtiene los posibles valores para n : 5, 1 (nuestro algoritmo prueba para $1 \leq n \leq \text{mcd}$). Es correcto ya que la clave es “LAURA” y su tamaño es 5.

Al igual que en los otros programas comprobamos que los argumentos sean correctos y los leemos con la función `read_args_kasiski` donde obtenemos el modo, que en este programa solo puede ser **TECLADO** o **IFILE**.

En ambos casos llamamos a la función `obtener_longitud_clave` que es la encargada de realizar todo el algoritmo y de obtener los posibles tamaños de la clave. Esta función recibe por argumentos el texto cifrado y el tamaño de las repeticiones (-l).

Si el modo es **TECLADO** pedimos el texto cifrado por pantalla, por el contrario, si es **IFILE** leemos el texto cifrado del fichero de entrada.

Ahora explicaremos la función:

`obtener_longitud_clave(char* texto, int longitud_repeticion)`

Para poder buscar repeticiones y medir las distancias, necesitamos quitar del texto aquellos caracteres que no formen parte del alfabeto (65 al 90 en ASCII). La función `limpia_texto` se encarga de ello, devuelve el texto limpio de caracteres que no pertenecen al alfabeto. (Ejemplo: HOLA PROFESOR. → HOLAPROFESOR)

Tras esto entramos en el bucle encargado de hallar el valor de n, es decir, del tamaño de la clave. Empezamos buscando repeticiones de los `longitud_repeticion` primeros caracteres del texto y guardamos la posición del primer carácter (`posicion_primera_aparicion`), en el caso de no encontrarse ninguna repetición en todo el texto seguimos buscando repeticiones del siguiente grupo de caracteres. Es decir, si el texto cifrado es ABCDEFG y `longitud_repeticion` = 3 buscaremos primero repeticiones de ABC `posicion_primera_aparicion` = 0 y si no hay de BCD `posicion_primera_aparicion` = 1 y así sucesivamente.

En el caso de que haya repeticiones, guardamos la posición del primer carácter de cada repetición. Una vez obtenidas las posiciones, tenemos que calcular las distancias de las repeticiones al conjunto de caracteres buscado (distancia = posición - `posicion_primera_aparicion`).

El tamaño de la clave cumple lo siguiente: $n \mid \text{mcd}(\text{distancias})$, es decir, $\text{mcd}(\text{distancias}) \% n = 0$. Para calcular el máximo común divisor de las distancias, hemos utilizado el algoritmo de euclides implementado en la función `mcd` en el fichero `euclides.c`. Calculamos el mcd de dos en dos, es decir, por ejemplo si las distancias son 5, 15, y 50 habría que calcular `mcd(5, 15, 30)` y nosotros lo calculamos de la siguiente manera: `mcd(30,mcd(15,5))`.

```
aux = array_posiciones[0];

if(p != 1){
    for (i = 1; i < p; i++){
        mcd_aux = mcd(array_posiciones[i], aux);
        aux = mcd_aux;
    }
}
```

Una vez obtenido el máximo común divisor probamos para que tamaños de clave se cumple que $\text{mcd}(\text{distancias}) \% n = 0$, n tiene que ser menor o igual que el máximo común divisor para que esto se cumpla, por lo que probamos para los valores $1 \leq n \leq \text{mcd}$. Mostrando por pantalla aquellos valores de n que si lo cumplen y esos son los posibles tamaños de la clave.

Es posible que ningún valor de n divida al máximo común divisor de las distancias, en ese caso seguiremos buscando otra repetición en el caso de no haber comprobado ya todos los posibles conjuntos de caracteres.

Tanto en el caso de que no se encuentre ninguna repetición en todo el texto cifrado como en el caso de no encontrar ningún posible valor de n aunque sí que haya repeticiones, mostramos el mensaje “No se ha encontrado el valor de n”.

Vamos a comprobar su funcionalidad obteniendo el tamaño de la clave de los textos cifrados anteriormente:

Ejemplo con el texto pequeño (salida_cifrar_texto_pequeño.txt) y clave CRIPTOGRAFIA:

```
laura@Universidad:~/Escritorio/cripto/P1$ ./kasiski -l 3 -i salida_cifrar_texto_pequeño.txt
Cogeremos el texto del fichero de entrada salida_cifrar_texto_pequeño.txt del cual obtendremos la longitud de la clave.

La repeticion buscada es: NRC
No hay repeticiones de NRC

La repeticion buscada es: RCG
No hay repeticiones de RCG

La repeticion buscada es: CGT
No hay repeticiones de CGT

La repeticion buscada es: GTI
No hay repeticiones de GTI

La repeticion buscada es: TIC
No hay repeticiones de TIC

La repeticion buscada es: ICL
No hay repeticiones de ICL

La repeticion buscada es: CLU
No hay repeticiones de CLU

La repeticion buscada es: LUS
No hay repeticiones de LUS

La repeticion buscada es: USI
Posicion 380
mcd( 372 ) = 372
n puede ser 372
n puede ser 186
n puede ser 124
n puede ser 93
n puede ser 62
n puede ser 31
n puede ser 12
n puede ser 6
n puede ser 4
n puede ser 3
n puede ser 2
n puede ser 1
```

El tamaño de la clave con la que fue cifrado este texto es de 12, y nuestro programa nos devuelve $n = 12$ como posible tamaño de clave. Por lo que es correcto.

Ejemplo con el texto grande (salida_cifrar_texto_grande.txt) y clave UAM:

```
laura@Universidad:~/Escritorio/cripto/P1$ ./kasiski -l 3 -i salida_cifrar_texto_grande.txt
Cogeremos el texto del fichero de entrada salida_cifrar_texto_grande.txt del cual obtendremos la longitud de la clave.

La repetición buscada es: WHM
Posición 24
Posición 81
Posición 3249
Posición 6186
Posición 6795
Posición 11469
Posición 12345
Posición 12963
mcd( 24  81  3249  6186  6795  11469  12345  12963 ) = 3
n puede ser 3
n puede ser 1
```

Este texto fue cifrado con una clave de tamaño 3, y el programa nos devuelve $n = 3$ como posible tamaño por lo que es correcto también este resultado.

Cabe destacar que el algoritmo de kasiski puede fallar, es posible que ninguna de las n que devuelve como posible tamaño de la clave, sea realmente el tamaño de la clave con la que se cifró el texto. Ya que pueden haber repeticiones que sean simples casualidades es decir, que no sean el mismo texto plano cifrado con la misma parte de la clave.

Índice de Coincidencia

Otra forma de averiguar el posible tamaño n de la clave utilizada en el cifrado de Vigenere es mediante el cálculo del Índice de Coincidencia. Este Índice determina la probabilidad de que dos caracteres en una cadena o texto sean el mismo. Si se aplica sobre un lenguaje y un análisis de frecuencias general, obtenemos el Índice de Coincidencia de ese lenguaje, como en el caso del inglés que es igual a 0.065 (es el idioma predeterminado que utilizamos en nuestros códigos, por lo que todos los textos que usamos están en inglés).

Lo más importante del uso de este índice es que si una cadena o texto descifrado posee un IC parecido al de su idioma, quiere decir que la supuesta clave utilizada ha sido la correcta y original del cifrado, y esto es un fundamento del posterior criptoanálisis al cifrado de Vigenere.

En general, el algoritmo consiste en iterar varias veces en busca de un tamaño n de clave correcto para un texto que nos dan cifrado. Este texto se va a dividir en bloques iguales de tamaño n (la supuesta clave) en cada iteración, y luego, se van a coger de cada bloque los caracteres cuyas posiciones puedan coincidir con la subclave i de esta clave de tamaño n (como se ha visto en teoría). Y una vez tenemos estos n vectores se procede al cálculo de sus Índices de Coincidencia. Si sus ICs, mediante el conteo de frecuencias de sus caracteres, se aproximan al IC del Inglés quiere decir que este supuesto tamaño de clave n es el correcto para la

clave original del cifrado. Esto se cumple porque Vigenere es un cifrado por desplazamiento, por lo que las frecuencias de los caracteres permanece igual en el estado cifrado y descifrado (es decir, el carácter A se repetirá las mismas veces en el texto plano que su correspondiente cifrado F, por ejemplo, en el texto cifrado que tenemos), por lo que si se cumple con todos los caracteres, debe de ser el mismo IC que el de su lenguaje.

Por lo tanto, hemos aplicado todo esto en nuestro módulo **afin_c.c** y su librería **afin_c.h**. Primero, aceptamos como argumentos de entrada de este programa el tamaño de los ngramas que se considerarán para este conteo de frecuencias (ya que el ngrama indicará el tamaño de lo que se considera un carácter, como hemos visto en Afín mejorado), también, un fichero de entrada opcional con el texto cifrado y un fichero de salida opcional donde se imprimirán unos mensajes con los posibles tamaños n de clave. Estos argumentos se leen como en los anteriores programas.

Después de procesar los argumentos y de limpiar el texto de entrada (es decir, obtener solo los caracteres pertenecientes al Alfabeto A-Z en una sola cadena), se crea una estructura “bloques” que será la encargada de almacenar los vectores ordenados por posiciones de subclave en cada iteración de supuesto tamaño de clave n. El número de claves probadas, es decir el número de iteraciones, se indica con la macro “NTRY”, que en nuestro caso es 20 (es decir, se prueba hasta n=20).

Esta estructura se llena con los vectores ordenados en la función `divide_texto_bloques_n()`, la que realiza las particiones en función del valor de n y del tamaño de los ngramas (dentro de esta se utiliza la función `asigna_ngramas_a_bloques()` para introducir los caracteres en orden en cada vector).

Una vez tenemos los vectores en “bloques” se itera en función del número de vectores y se va calculando el Índice de Coincidencia de cada uno de estos con la función `calcula_ic()`. En esta función se genera la tabla de frecuencias de cada uno de los caracteres distintos del Alfabeto que aparecen en el vector y en un bucle posterior se realiza el conteo de frecuencias, en función de sus repeticiones dentro del vector. Con estos valores, se realiza la fórmula vista en teoría del Índice de Coincidencia sobre una cadena o texto ($\sum_{i=0}^{n-1} f_i * (f_i - 1) / (l(l-1))$).

Luego, si el IC de todos estos vectores (en nuestro caso aplicamos la media de los ICs calculados) del supuesto tamaño de clave n se aproxima (con un rango de 0.005 por encima y por debajo) se da por posible clave válida. Todo esto se puede ver bien en las impresiones realizadas por pantalla en la ejecución.

Para demostrar su funcionamiento, hemos cifrado el mismo fragmento de Don Quijote en inglés con una clave de tamaño 6 “INDICE”. Se puede ver en la siguiente imagen.

```
X541@LAPTOP-BanMat27 /c/Users/x541/desktop/cripto/p1
$ ./vigenere -C -k INDICE -i don_quixote.txt -o entrada_descifrar.txt
# Ha elegido cifrar #

Ciframos el fichero de entrada: don_quixote.txt
Guardamos el texto cifrado en el fichero de salida: entrada_descifrar.txt
```

Por lo que, ahora, ejecutamos el programa del Índice de Coincidencia sobre el texto cifrado de este fragmento y probará desde $n=1$ hasta $n=20$ y nos dirá por pantalla cuales son los posibles tamaños de clave originales.

Esta ejecución se realiza de la forma:

```
> ./indice_c -l 1 [-i fichero_entrada_opcional] [-o fichero_salida_opcional]
```

En nuestro caso será el tamaño de ngrama igual a 1 ya que el Alfabeto original sabemos que es el Ascii A-Z. Si no lo supiéramos, habría que ir probando con distintos tamaños de ngramas también.

La ejecución del IC sobre el fragmento del Quijote cifrado anterior es la siguiente:

```
[+] [-l] [-o] son opcionales.
X541@LAPTOP-BanMat27 /c/Users/x541/desktop/cripto/p1
$ ./indice_c -l 1 -i entrada_descifrar.txt
Tamaño del NGramma: 1

###Probando con n = 1:
Vector1 con IC = 0.043935
Media de IC = 0.043935 para tamaño n = 1

###Probando con n = 2:
Vector1 con IC = 0.047489
Vector2 con IC = 0.046422
Media de IC = 0.046956 para tamaño n = 2

###Probando con n = 3:
Vector1 con IC = 0.056028
Vector2 con IC = 0.055689
Vector3 con IC = 0.052511
Media de IC = 0.058073 para tamaño n = 3

###Probando con n = 4:
Vector1 con IC = 0.047512
Vector2 con IC = 0.046799
Vector3 con IC = 0.047423
Vector4 con IC = 0.046348
Media de IC = 0.047019 para tamaño n = 4

###Probando con n = 5:
Vector1 con IC = 0.043314
Vector2 con IC = 0.044913
Vector3 con IC = 0.043793
Vector4 con IC = 0.044018
Vector5 con IC = 0.043696
Media de IC = 0.043947 para tamaño n = 5

###Probando con n = 6:
Vector1 con IC = 0.066729
Vector2 con IC = 0.066619
Vector3 con IC = 0.065363
Vector4 con IC = 0.065439
Vector5 con IC = 0.067492
Vector6 con IC = 0.066993
Media de IC = 0.066439 para tamaño n = 6
Tamaño n = 6 con IC = 0.066439 y 1-gramas puede ser el correcto!
```

Los mensajes por pantalla continúan hasta $n=20$ de la misma manera, mostrando los ICs de cada vector existente en la iteración con tamaño de clave n probado. Se puede ver, por tanto, que el programa ha averiguado correctamente el tamaño de la clave original “INDICE” de 6 caracteres.

Ahora vamos a cifrarlo otra vez con una clave más grande como “CRIPTOGRAFIA” de 12 caracteres. Tras aplicar el programa del índice obtenemos:

```
X541@LAPTOP-DanMat27 /c/Users/x541/desktop/cripto/p1
$ ./vigenere -C -k CRIPTOGRAFIA -i don_quixote.txt -o entrada_descifrar.txt
# Ha elegido cifrar #

Ciframos el fichero de entrada: don_quixote.txt
Guardamos el texto cifrado en el fichero de salida: entrada_descifrar.txt
```

```
MINGW32/c/Users/x541/desktop/cripto/p1
X541@LAPTOP-DanMat27 /c/Users/x541/desktop/cripto/p1
$ ./indice_c -l 1 -i entrada_descifrar.txt
Tamaño del NGrama: 1

###Probando con n = 1:
Vector1 con IC = 0.039979
Media de IC = 0.039979 para tamaño n = 1

###Probando con n = 2:
Vector1 con IC = 0.042890
Vector2 con IC = 0.042641
Media de IC = 0.042765 para tamaño n = 2

###Probando con n = 3:
Vector1 con IC = 0.044618
Vector2 con IC = 0.045576
Vector3 con IC = 0.048271
Media de IC = 0.046155 para tamaño n = 3

###Probando con n = 4:
Vector1 con IC = 0.045389
Vector2 con IC = 0.045945
Vector3 con IC = 0.049676
Vector4 con IC = 0.045943
Media de IC = 0.046718 para tamaño n = 4

###Probando con n = 5:
Vector1 con IC = 0.046243
Vector2 con IC = 0.039688
Vector3 con IC = 0.048193
Vector4 con IC = 0.039497
Vector5 con IC = 0.048193
Media de IC = 0.039963 para tamaño n = 5

###Probando con n = 6:
Vector1 con IC = 0.054439
Vector2 con IC = 0.066619
Vector3 con IC = 0.049184
Vector4 con IC = 0.050665
Vector5 con IC = 0.055524
Vector6 con IC = 0.054935
Media de IC = 0.055228 para tamaño n = 6

###Probando con n = 7:
Vector1 con IC = 0.039832
```

```
MINGW32/c/Users/x541/desktop/cripto/p1
Vector10 con IC = 0.043126
Media de IC = 0.042737 para tamaño n = 10

###Probando con n = 11:
Vector1 con IC = 0.048311
Vector2 con IC = 0.039198
Vector3 con IC = 0.040756
Vector4 con IC = 0.040699
Vector5 con IC = 0.040007
Vector6 con IC = 0.039290
Vector7 con IC = 0.039728
Vector8 con IC = 0.040073
Vector9 con IC = 0.040334
Vector10 con IC = 0.039614
Vector11 con IC = 0.039412
Media de IC = 0.039947 para tamaño n = 11

###Probando con n = 12:
Vector1 con IC = 0.068526
Vector2 con IC = 0.066185
Vector3 con IC = 0.063318
Vector4 con IC = 0.064366
Vector5 con IC = 0.065858
Vector6 con IC = 0.069594
Vector7 con IC = 0.065569
Vector8 con IC = 0.066917
Vector9 con IC = 0.067728
Vector10 con IC = 0.067224
Vector11 con IC = 0.068667
Vector12 con IC = 0.065119
Media de IC = 0.066589 para tamaño n = 12
Tamaño n = 12 con IC = 0.066589 y 1-gramas puede ser el correcto!

###Probando con n = 13:
Vector1 con IC = 0.039743
Vector2 con IC = 0.040727
Vector3 con IC = 0.039882
Vector4 con IC = 0.041318
Vector5 con IC = 0.041524
Vector6 con IC = 0.039868
Vector7 con IC = 0.039598
Vector8 con IC = 0.039226
Vector9 con IC = 0.039875
Vector10 con IC = 0.040475
Vector11 con IC = 0.039134
Vector12 con IC = 0.040286
```

Se ve cómo ha vuelto a acertar con el tamaño de la clave original, que era de 12 caracteres y que el programa funciona correctamente en cualquier caso normal.

Por si quería ver una clave más larga, hemos cifrado Don Quijote con una clave de tamaño 47 “ALLAVANCONELBALONENLOSPIESYNADIELOSPODRADETENER”. Para llegar a ese tamaño de clave hemos cambiado un momento NTRY a 50.

```
X541@LAPTOP-DanMat27 /c/Users/x541/desktop/cripto/p1
$ ./vigenere -C -k ALLAVANCONELBALONENLOSPIESYNADIELOSPODRADETENER -i don_quixote.txt -o entrada_descifrar.txt
# Ha elegido cifrar #

Ciframos el fichero de entrada: don_quixote.txt
Guardamos el texto cifrado en el fichero de salida: entrada_descifrar.txt
```

Al ejecutar el programa del Índice de Coincidencia este ha sido el resultado.

```
MINGW32:/c/Users/x541/desktop/cripto/p1
X541@LAPTOP-DanMat27 /c/Users/x541/desktop/cripto/p1
$ ./indice_c -l 1 -i entrada_descifrar.txt
Tamaño del NGramas: 1

###Probando con n = 1:
Vector1 con IC = 0.040968
Media de IC = 0.040968 para tamaño n = 1

###Probando con n = 2:
Vector1 con IC = 0.040889
Vector2 con IC = 0.041078
Media de IC = 0.040943 para tamaño n = 2

###Probando con n = 3:
Vector1 con IC = 0.040920
Vector2 con IC = 0.040879
Vector3 con IC = 0.040977
Media de IC = 0.040926 para tamaño n = 3

###Probando con n = 4:
Vector1 con IC = 0.040695
Vector2 con IC = 0.041635
Vector3 con IC = 0.040989
Vector4 con IC = 0.040523
Media de IC = 0.040941 para tamaño n = 4

###Probando con n = 5:
Vector1 con IC = 0.040568
Vector2 con IC = 0.041212
Vector3 con IC = 0.041973
Vector4 con IC = 0.040837
Vector5 con IC = 0.040634
Media de IC = 0.041043 para tamaño n = 5

###Probando con n = 6:
Vector1 con IC = 0.040679
Vector2 con IC = 0.040766
Vector3 con IC = 0.040629
Vector4 con IC = 0.040957
Vector5 con IC = 0.040989
Vector6 con IC = 0.041341
Media de IC = 0.040888 para tamaño n = 6

###Probando con n = 7:
```

```
Seleccionar MINGW32:/c/Users/x541/desktop/cripto/p1
###Probando con n = 47:
Vector1 con IC = 0.063562
Vector2 con IC = 0.063894
Vector3 con IC = 0.063894
Vector4 con IC = 0.067472
Vector5 con IC = 0.066698
Vector6 con IC = 0.070293
Vector7 con IC = 0.071359
Vector8 con IC = 0.062788
Vector9 con IC = 0.063681
Vector10 con IC = 0.064415
Vector11 con IC = 0.068841
Vector12 con IC = 0.067283
Vector13 con IC = 0.066240
Vector14 con IC = 0.063586
Vector15 con IC = 0.064723
Vector16 con IC = 0.064794
Vector17 con IC = 0.067481
Vector18 con IC = 0.072823
Vector19 con IC = 0.063420
Vector20 con IC = 0.066643
Vector21 con IC = 0.068444
Vector22 con IC = 0.064818
Vector23 con IC = 0.063681
Vector24 con IC = 0.065126
Vector25 con IC = 0.072820
Vector26 con IC = 0.071004
Vector27 con IC = 0.068373
Vector28 con IC = 0.064344
Vector29 con IC = 0.074938
Vector30 con IC = 0.068931
Vector31 con IC = 0.070743
Vector32 con IC = 0.066690
Vector33 con IC = 0.064510
Vector34 con IC = 0.064312
Vector35 con IC = 0.065585
Vector36 con IC = 0.068655
Vector37 con IC = 0.068783
Vector38 con IC = 0.069813
Vector39 con IC = 0.059786
Vector40 con IC = 0.067280
Vector41 con IC = 0.065935
Vector42 con IC = 0.062785
Vector43 con IC = 0.070994
Vector44 con IC = 0.063978
Vector45 con IC = 0.066221
Vector46 con IC = 0.065147
Vector47 con IC = 0.067748
Media de IC = 0.066467 para tamaño n = 47
Tamaño n = 47 con IC = 0.066467 y 1-gramas puede ser el correcto!
```

Se puede ver que ha vuelto a averiguar el tamaño de clave correcto.

Criptoanálisis completo

Tras haber obtenido el tamaño de la clave ejecutando el Test de Kasiski o por el Índice de coincidencia explicados anteriormente pasamos a hallar cada componente de la clave de tamaño n.

Está implementado en el fichero **criptoanalisis_vigenere.c**.

El programa criptoanalisis_vigenere acepta los siguientes parámetros:

`./criptoanalisis_vigenere {-n tamClave} [-i filein]`

Obligatorio:

`{-n tamClave}` → indica el tamaño de la clave.

Opcional:

`[-i filein]` → fichero de entrada con el texto cifrado por el método vigenere.

Primero comprobamos que todos los argumentos pos terminal sean correctos y obtenemos el modo, como en los apartados anteriores. Tras esto, limpiamos el texto con la función *limpia_texto* que devuelve el texto sin caracteres que no formen parte del alfabeto (65 - 90 en ASCII).

Dividimos el texto limpio en bloques de tamaño n , el tamaño de la clave para el que estamos probando, y con estos bloques formamos los diferentes vectores que serán n vectores de longitud el número de bloques que se formen. Cada vector i tiene diferentes caracteres los cuales han sido cifrados con el carácter k_i de la clave, suponiendo que el valor de n es el correcto. La función *divide_texto_bloques_n* se encarga de ello.

Para cada vector (vector_i) probamos todos los posibles valores que puede tomar k_i, que son los valores del 0 al 25 en nuestro alfabeto Z26. Para cada valor de clave calculamos el índice de coincidencia del vector con la función *calcula_ic_criptoanalisis* y en el caso de devolver un valor entre 0,070 y 0,060 (0,065 es el IC del inglés) mostramos ese valor de k_i como posible. Ese valor es posible porque ese vector descifrado con k_i, con las frecuencias de cada carácter en el vector y las probabilidades de cada carácter en el lenguaje inglés dan un índice de coincidencia cercano a 0,065, significa que ese vector descifrado pertenece al lenguaje inglés.

Es posible que el algoritmo acepte varios posibles valores de k_i y también es posible que no encuentre ninguno.

La función `calcula_ic_criptoanalisis` inicialmente calcula las frecuencias de los diferentes caracteres en el vector. Las frecuencias de los caracteres cifrados coinciden con las frecuencias de los caracteres descifrados. Tras haber obtenido las frecuencias pasamos a calcular el índice de coincidencia siguiendo la fórmula:

$$IC(\text{vector}_j) = n/l * \sum_{i=0}^{l-1} p_i * f_{ik_j}$$

n → tamaño de la clave

l → longitud del texto cifrado

p_i → probabilidad de encontrar el carácter i en el lenguaje inglés, el carácter es el descifrado con el valor de la clave k_i que estamos probando. Estas probabilidades las sacamos de la tabla del enunciado.

f_{ik_j} → frecuencia del carácter i en el vector.

Ejemplos de ejecución:

Empezamos cifrando el texto plano **don_quixote.txt** con el método de vigenere y clave UAM:

```
laura@Universidad:~/Escritorio/cripto/P1$ ./vigenere -c -k UAM -i don_quixote.txt -o salida_vigenere_don_quixote.txt
# Ha elegido cifrar #
Ciframos el fichero de entrada: don_quixote.txt
Guardamos el texto cifrado en el fichero de salida: salida_vigenere_don_quixote.txt
```

El texto cifrado **salida_vigenere_don_quixote.txt** es sobre el que vamos a realizar el criptoanálisis.

Primero hallamos el tamaño de la clave:

Test de kasiski:

```
laura@Universidad:~/Escritorio/cripto/P1$ ./kasiski -l 3 -i salida_vigenere_don_quixote.txt
Cogeremos el texto del fichero de entrada salida_vigenere_don_quixote.txt del cual obtendremos la longitud de la clave.

La repeticion buscada es: WHM
Posicion 24
Posicion 81
Posicion 3249
Posicion 6186
Posicion 6795
Posicion 11469
Posicion 12345
Posicion 12963
mcd( 24 81 3249 6186 6795 11469 12345 12963 ) = 3
n puede ser 3
n puede ser 1
```

Devuelve posibles valores de n 3 y 1.

Índice de coincidencia:

```
laura@Universidad:~/Escritorio/cripto/P1$ ./indice_c -l 1 -i salida_vigenere_don_quixote.txt
Tamanio del NGrama: 1

###Probando con n = 1:
Vector1 con IC = 0.046733
Media de IC = 0.046733 para tamanio n = 1

###Probando con n = 2:
Vector1 con IC = 0.046668
Vector2 con IC = 0.046834
Media de IC = 0.046751 para tamanio n = 2

###Probando con n = 3:
Vector1 con IC = 0.066020
Vector2 con IC = 0.067111
Vector3 con IC = 0.066177
Media de IC = 0.066436 para tamanio n = 3
Tamanio n = 3 con IC = 0.066436 y 1-gramas puede ser el correcto!

###Probando con n = 4:
Vector1 con IC = 0.046831
Vector2 con IC = 0.046877
Vector3 con IC = 0.046423
Vector4 con IC = 0.046900
Media de IC = 0.046758 para tamanio n = 4

###Probando con n = 5:
Vector1 con IC = 0.046534
Vector2 con IC = 0.046795
Vector3 con IC = 0.046191
Vector4 con IC = 0.047373
Vector5 con IC = 0.046849
Media de IC = 0.046748 para tamanio n = 5

###Probando con n = 6:
Vector1 con IC = 0.066729
Vector2 con IC = 0.066619
Vector3 con IC = 0.065363
Vector4 con IC = 0.065439
Vector5 con IC = 0.067492
Vector6 con IC = 0.066993
Media de IC = 0.066439 para tamanio n = 6
Tamanio n = 6 con IC = 0.066439 y 1-gramas puede ser el correcto!

###Probando con n = 7:
Vector1 con IC = 0.046563
Vector2 con IC = 0.046831
```

```
Vector3 con IC = 0.045821
Vector4 con IC = 0.047513
Vector5 con IC = 0.046077
Vector6 con IC = 0.046999
Vector7 con IC = 0.046823
Media de IC = 0.046661 para tamanio n = 7

###Probando con n = 8:
Vector1 con IC = 0.047035
Vector2 con IC = 0.046587
Vector3 con IC = 0.045973
Vector4 con IC = 0.046487
Vector5 con IC = 0.046511
Vector6 con IC = 0.047070
Vector7 con IC = 0.046947
Vector8 con IC = 0.047686
Media de IC = 0.046787 para tamanio n = 8

###Probando con n = 9:
Vector1 con IC = 0.065728
Vector2 con IC = 0.065917
Vector3 con IC = 0.066945
Vector4 con IC = 0.065087
Vector5 con IC = 0.067191
Vector6 con IC = 0.064666
Vector7 con IC = 0.067017
Vector8 con IC = 0.068659
Vector9 con IC = 0.064807
Media de IC = 0.066499 para tamanio n = 9
Tamanio n = 9 con IC = 0.066499 y 1-gramas puede ser el correcto!

###Probando con n = 10:
Vector1 con IC = 0.046400
Vector2 con IC = 0.046981
Vector3 con IC = 0.045888
Vector4 con IC = 0.046947
Vector5 con IC = 0.046625
Vector6 con IC = 0.046447
Vector7 con IC = 0.046635
Vector8 con IC = 0.046248
Vector9 con IC = 0.047802
Vector10 con IC = 0.046836
Media de IC = 0.046681 para tamanio n = 10

###Probando con n = 11:
Vector1 con IC = 0.048285
Vector2 con IC = 0.046289
Vector3 con IC = 0.047832
```

```
Vector4 con IC = 0.045095
Vector5 con IC = 0.045545
Vector6 con IC = 0.047663
Vector7 con IC = 0.045989
Vector8 con IC = 0.047455
Vector9 con IC = 0.047762
Vector10 con IC = 0.046919
Vector11 con IC = 0.045870
Media de IC = 0.046791 para tamanto n = 11

###Probando con n = 12:
Vector1 con IC = 0.068526
Vector2 con IC = 0.066185
Vector3 con IC = 0.063318
Vector4 con IC = 0.064366
Vector5 con IC = 0.065858
Vector6 con IC = 0.069594
Vector7 con IC = 0.065569
Vector8 con IC = 0.066917
Vector9 con IC = 0.067728
Vector10 con IC = 0.067224
Vector11 con IC = 0.068667
Vector12 con IC = 0.065119
Media de IC = 0.066589 para tamanio n = 12
Tamanio n = 12 con IC = 0.066589 y 1-gramas puede ser el correcto!

###Probando con n = 13:
Vector1 con IC = 0.046329
Vector2 con IC = 0.047994
Vector3 con IC = 0.047797
Vector4 con IC = 0.048212
Vector5 con IC = 0.048971
Vector6 con IC = 0.047612
Vector7 con IC = 0.046180
Vector8 con IC = 0.046004
Vector9 con IC = 0.046198
Vector10 con IC = 0.048110
Vector11 con IC = 0.045365
Vector12 con IC = 0.045756
Vector13 con IC = 0.045720
Media de IC = 0.046942 para tamanio n = 13

###Probando con n = 14:
Vector1 con IC = 0.046208
Vector2 con IC = 0.048041
Vector3 con IC = 0.046084
Vector4 con IC = 0.048375
Vector5 con IC = 0.046127
```

```

Vector6 con IC = 0.046515
Vector7 con IC = 0.047360
Vector8 con IC = 0.046654
Vector9 con IC = 0.045900
Vector10 con IC = 0.045200
Vector11 con IC = 0.046217
Vector12 con IC = 0.046337
Vector13 con IC = 0.047062
Vector14 con IC = 0.046493
Media de IC = 0.046612 para tamanio n = 14

###Probando con n = 15:
Vector1 con IC = 0.063746
Vector2 con IC = 0.065191
Vector3 con IC = 0.067719
Vector4 con IC = 0.065411
Vector5 con IC = 0.068636
Vector6 con IC = 0.065534
Vector7 con IC = 0.064938
Vector8 con IC = 0.070769
Vector9 con IC = 0.064117
Vector10 con IC = 0.067739
Vector11 con IC = 0.064146
Vector12 con IC = 0.066680
Vector13 con IC = 0.068243
Vector14 con IC = 0.066676
Vector15 con IC = 0.066142
Media de IC = 0.066379 para tamanio n = 15
Tamanio n = 15 con IC = 0.066379 y 1-gramas puede ser el correcto!

###Probando con n = 16:
Vector1 con IC = 0.046759
Vector2 con IC = 0.046888
Vector3 con IC = 0.046388
Vector4 con IC = 0.045279
Vector5 con IC = 0.046987
Vector6 con IC = 0.046759
Vector7 con IC = 0.046372
Vector8 con IC = 0.046523
Vector9 con IC = 0.047242
Vector10 con IC = 0.045748
Vector11 con IC = 0.045356
Vector11 con IC = 0.045356
Vector12 con IC = 0.047544
Vector13 con IC = 0.046125
Vector14 con IC = 0.047094
Vector15 con IC = 0.047621
Vector16 con IC = 0.048911
Media de IC = 0.046725 para tamanio n = 16

###Probando con n = 17:
Vector1 con IC = 0.046861
Vector2 con IC = 0.046127
Vector3 con IC = 0.046003
Vector4 con IC = 0.044928
Vector5 con IC = 0.046660
Vector6 con IC = 0.047189
Vector7 con IC = 0.045709
Vector8 con IC = 0.044727
Vector9 con IC = 0.048732
Vector10 con IC = 0.046759
Vector11 con IC = 0.046657
Vector12 con IC = 0.046870
Vector13 con IC = 0.045785
Vector14 con IC = 0.047816
Vector15 con IC = 0.046689
Vector16 con IC = 0.046009
Vector17 con IC = 0.048546
Media de IC = 0.046592 para tamanio n = 17

###Probando con n = 18:
Vector1 con IC = 0.067783
Vector2 con IC = 0.066175
Vector3 con IC = 0.068949
Vector4 con IC = 0.063756
Vector5 con IC = 0.068567
Vector6 con IC = 0.065946
Vector7 con IC = 0.065629
Vector8 con IC = 0.067993

```

```

Vector9 con IC = 0.063457
Vector10 con IC = 0.063822
Vector11 con IC = 0.064941
Vector12 con IC = 0.069749
Vector13 con IC = 0.066297
Vector14 con IC = 0.065734
Vector15 con IC = 0.063446
Vector16 con IC = 0.068556
Vector17 con IC = 0.069113
Vector18 con IC = 0.065417
Media de IC = 0.066407 para tamanio n = 18
Tamanio n = 18 con IC = 0.066407 y 1-gramas puede ser el correcto!

###Probando con n = 19:
Vector1 con IC = 0.048385
Vector2 con IC = 0.047106
Vector3 con IC = 0.045049
Vector4 con IC = 0.047466
Vector5 con IC = 0.047238
Vector6 con IC = 0.047090
Vector7 con IC = 0.047121
Vector8 con IC = 0.048210
Vector9 con IC = 0.045250
Vector10 con IC = 0.046192
Vector11 con IC = 0.048675
Vector12 con IC = 0.046002
Vector13 con IC = 0.046645
Vector14 con IC = 0.045138
Vector15 con IC = 0.046850
Vector16 con IC = 0.046114
Vector17 con IC = 0.048055
Vector18 con IC = 0.046788
Vector19 con IC = 0.045033
Media de IC = 0.046758 para tamanio n = 19

###Probando con n = 20:
Vector1 con IC = 0.047713
Vector2 con IC = 0.048607
Vector3 con IC = 0.045743
Vector4 con IC = 0.046534
Vector5 con IC = 0.047178
Vector6 con IC = 0.046917

```

```

Vector7 con IC = 0.046418
Vector8 con IC = 0.046874
Vector9 con IC = 0.046723
Vector10 con IC = 0.046865
Vector11 con IC = 0.045701
Vector12 con IC = 0.046564
Vector13 con IC = 0.045890
Vector14 con IC = 0.048230
Vector15 con IC = 0.046380
Vector16 con IC = 0.045976
Vector17 con IC = 0.046118
Vector18 con IC = 0.045478
Vector19 con IC = 0.048415
Vector20 con IC = 0.047522
Media de IC = 0.046792 para tamano n = 20

Se ha encontrado algun valor n valido para el texto introducido!

```

Devuelve posibles valores de n 3, 6, 9, 12, 15, y 18.

Tras estos posibles valores de n empezamos a probar hasta conseguir todos los componentes de la clave:

```

laura@Universidad:~/Escritorio/cripto/P1$ ./criptoanalisis_vigenere -n 3 -i salida_vigenere_don_
quixote.txt
Criptoanalizamos el fichero de entrada: salida_vigenere_don_quixote.txt
Probando para el vector 0
Posible k_0 = U con IC = 0.065659
Probando para el vector 1
Posible k_1 = A con IC = 0.066074
Probando para el vector 2
Posible k_2 = M con IC = 0.065465

```

Devuelve que la clave es UAM, vamos a comprobarlo:

```

laura@Universidad:~/Escritorio/cripto/P1$ ./vigenere -D -k UAM -i salida_vigenere_don_quixote.tx
t -o texto_criptoanalizado.txt
# Ha elegido descifrar #

Desciframos el fichero de entrada: salida_vigenere_don_quixote.txt
Guardamos el texto descifrado en el fichero de salida: texto_criptoanalizado.txt

```

Vemos si el texto_criptoanalizado.txt tiene alguna diferencia con el original don_quixote.txt:

```

laura@Universidad:~/Escritorio/cripto/P1$ diff don_quixote.txt texto_criptoanalizado.txt
laura@Universidad:~/Escritorio/cripto/P1$ 

```

Por lo tanto, hemos realizado correctamente el criptoanálisis, la clave UAM es con la que se cifró el texto plano.

Otro ejemplo:

Ahora ciframos con el método vigenere el texto plano del fichero **hamlet.txt** con la clave CRIPTOGRAFIA:

```
laura@Universidad:~/Escritorio/cripto/P1$ ./vigenere -C -k CRIPTOGRAFIA -i hamlet.txt -o salida_vigenere_hamlet.txt
# Ha elegido cifrar #

Ciframos el fichero de entrada: hamlet.txt
Guardamos el texto cifrado en el fichero de salida: salida_vigenere_hamlet.txt
laura@Universidad:~/Escritorio/cripto/P1$
```

Hallamos el tamaño de la clave:

Test de Kasiski:

Tras probar con diferentes tamaños de repetición, el algoritmo nos daba menos valores posibles de n con tamaño de repetición 10.

```
laura@Universidad:~/Escritorio/cripto/P1$ ./kasiski -l 10 -i salida_vigenere_hamlet.txt
La repeticion buscada es: GOWINPMVKI
Posicion 1642
mcd( 24 ) = 24
n puede ser 24
n puede ser 12
n puede ser 8
n puede ser 6
n puede ser 4
n puede ser 3
n puede ser 2
n puede ser 1
laura@Universidad:~/Escritorio/cripto/P1$
```

Devuelve como posibles valores de n 24, 12, 8, 6, 4, 3, 2 y 1.

Con esta prueba llegamos a la conclusión de que el Test de Kasiski funciona peor para claves que son de mayor longitud.

Índice de coincidencia:

```
laura@Universidad:~/Escritorio/cripto/P1$ ./indice_c -l 1 -i salida_vigenere_hamlet.txt
###Probando con n = 12:
Vector1 con IC = 0.070194
Vector2 con IC = 0.060823
Vector3 con IC = 0.060512
Vector4 con IC = 0.059474
Vector5 con IC = 0.059100
Vector6 con IC = 0.067782
Vector7 con IC = 0.063180
Vector8 con IC = 0.066004
Vector9 con IC = 0.067817
Vector10 con IC = 0.059066
Vector11 con IC = 0.071653
Vector12 con IC = 0.064644
Media de IC = 0.064187 para tamano n = 12
Tamano n = 12 con IC = 0.064187 y 1-gramas puede ser el correcto!
```

Devuelve un único valor de n posible que es 12.

Tras obtener los posibles tamaños de clave, hallamos sus componentes:

Empezamos probando con n = 12 ya que es el valor común en ambos algoritmos.

```

laura@Universidad:~/Escritorio/cripto/P1$ ./criptoanalisis_vigenere -n 12 -i salida_vigenere_hamlet.txt
Criptoanalizamos el fichero de entrada: salida_vigenere_hamlet.txt
Probando para el vector 0
Posible k_0 = C con IC = 0.066989
Probando para el vector 1
Posible k_1 = R con IC = 0.061931
Probando para el vector 2
Posible k_2 = I con IC = 0.063480
Probando para el vector 3
Posible k_3 = P con IC = 0.061880
Probando para el vector 4
Posible k_4 = T con IC = 0.062819
Probando para el vector 5
Posible k_5 = O con IC = 0.066963
Probando para el vector 6
Posible k_6 = G con IC = 0.064264
Probando para el vector 7
Posible k_7 = R con IC = 0.064641
Probando para el vector 8
Posible k_8 = A con IC = 0.066381
Probando para el vector 9
Posible k_9 = F con IC = 0.061289
Probando para el vector 10
Posible k_10 = I con IC = 0.067423
Probando para el vector 11
Posible k_11 = A con IC = 0.064888

```

Nos devuelve que la clave es CRIPTOGRAFIA, lo comprobamos:

```

laura@Universidad:~/Escritorio/cripto/P1$ ./vigenere -D -k CRIPTOGRAFIA -i salida_vigenere_hamlet.txt -o resultado_criptoanalisis.txt
# Ha elegido descifrar #

Desciframos el fichero de entrada: salida_vigenere_hamlet.txt
Guardamos el texto descifrado en el fichero de salida: resultado_criptoanalisis.txt
laura@Universidad:~/Escritorio/cripto/P1$ diff resultado_criptoanalisis.txt hamlet.txt
laura@Universidad:~/Escritorio/cripto/P1$ 

```

La clave CRIPTOGRAFIA es con la que se cifró el texto hamlet.txt ya que no tiene ninguna diferencia con resultado_criptoanalisis.txt.

Es cierto, que en la realidad para comprobar si se ha realizado correctamente el criptoanálisis no se compara con el texto original porque no dispones de él, en la realidad se comprueba que al descifrarlo con la clave hallada el texto tenga sentido.

Además de estos ejemplos que hemos mostrado, hemos probado con textos más pequeños pero para esos tamaños los algoritmos fallaban. Por lo tanto, para que funcionen correctamente el texto tiene que tener un tamaño considerable.

A continuación vamos a estudiar que pasa con el criptoanálisis cuando el tamaño de la clave se acerca a la longitud del texto.

Ciframos la palabra LAURA de tamaño 5 con la clave LAURA de tamaño 5 también:

```

laura@Universidad:~/Escritorio/cripto/P1$ ./vigenere -C -k LAURA
# Ha elegido cifrar #

Introduzca lo que quiere cifrar:
LAURA

Cifrando LAURA
Texto cifrado: WAOIA

```

Ni con el índice de coincidencia ni con kasiski obtenemos ningún valor de n.

```

laura@Universidad:~/Escritorio/cripto/P1$ ./indice_c -l 1
Tamanio del NGrama: 1

Introduzca el texto a evaluar:
WAOIA

###Probando con n = 1:
Vector1 con IC = 0.100000
Media de IC = 0.100000 para tamano n = 1

###Probando con n = 2:
Vector1 con IC = 0.000000
Vector2 con IC = 0.000000
Media de IC = 0.000000 para tamano n = 2

###Probando con n = 3:
Vector1 con IC = 0.000000
Vector2 con IC = 1.000000
Vector3 con IC = -nan
Media de IC = -nan para tamano n = 3

```

Aquí el índice de coincidencia tiende a 0 para todos los valores de n, por lo tanto, ningún valor de n es apropiado.

```

laura@Universidad:~/Escritorio/cripto/P1$ ./kasiski -l 1
Introduce el texto cifrado por el metodo vigenere del cual obtendremos la longitud de la clave
WAOIA

La repeticion buscada es: W
No hay repeticiones de W

La repeticion buscada es: A
Posicion 4
mcd( 3 ) = 3
n puede ser 3
n puede ser 1
laura@Universidad:~/Escritorio/cripto/P1$ 

```

Y en kasiski solo ha dado la casualidad de que se repite un carácter y dado un valor incorrecto de n ya que es 5 y ha dado 3 y 1. No es posible que el algoritmo funcione correctamente ya que la base de este algoritmo es encontrar partes de texto plano iguales cifradas con la misma parte de la clave, en este caso es imposible que esto ocurra.

Suponemos que hemos hallado n = 5 e intentamos hallar los componentes de la clave:

```

laura@Universidad:~/Escritorio/cripto/P1$ ./criptoanalisis_vigenere -n 5
Introduzca lo que quiere criptoanalizar:
WAOIA

Criptoanalizando WAOIA
Probando para el vector 0
Posible k_0 = E con IC = 0.065400
Posible k_0 = F con IC = 0.061200
Probando para el vector 1
Posible k_1 = I con IC = 0.065400
Posible k_1 = J con IC = 0.061200
Probando para el vector 2
Posible k_2 = W con IC = 0.065400
Posible k_2 = X con IC = 0.061200
Probando para el vector 3
Posible k_3 = Q con IC = 0.065400
Posible k_3 = R con IC = 0.061200
Probando para el vector 4
Posible k_4 = I con IC = 0.065400
Posible k_4 = J con IC = 0.061200

```

Falla, el programa da varios valores e incorrectos todos.

Hemos llegado a la conclusión que para $n \rightarrow l$ (tamaño texto) el IC = 0 porque los bloques formados por caracteres cifrados con la misma subclave acaban teniendo un solo carácter, por lo que no existen frecuencias en ellos y la definición de IC no tiene sentido en este caso.

3. Cifrado de flujo

En este apartado se nos pide crear un programa capaz de realizar un cifrado de flujo. Este tipo de cifrado es muy similar a un cifrado de desplazamiento polialfabético por bloques como Vigenere, ya que su proceso de cifrado es muy parecido pero sustituyendo carácter a carácter por su valor cifrado al aplicarle su correspondiente subclave. Lo diferente respecto a Vigenere es esto último, la clave K es una sucesión o flujo de subclaves k_i generadas de manera aleatoria (realmente pseudoaleatoria) mediante un generador de claves.

El generador del flujo de claves es realmente la robustez del cifrador de flujo, ya que dependiendo de su aleatoriedad y su espacio de claves generadas, el cifrador será más o menos robusto.

Por lo tanto, primero, hemos creado **flujo.c** y su correspondiente header **flujo.h**. Y, en ellos hemos creado el programa correspondiente a realizar el cifrado y el descifrado de flujo. Este programa es muy similar a **vigenere.c**, por lo que el cuerpo del programa es, prácticamente, el mismo (los bucles de cifrado y descifrado, la ecuación aplicada a cada carácter también es un desplazamiento, y la lectura de los archivos de entrada y salida es la misma).

Sin embargo, la lectura de los argumentos de entrada son distintos a ese otro cifrador, ya que no requiere la introducción de ninguna clave por argumentos. La introducción de ficheros de entrada y salida es la misma, con el siguiente formato de ejecución:

```
> ./flujo {-C/-D} [-i fichero_entrada_opcional] [-o fichero_salida_opcional]
```

Ahora, la parte clave de este algoritmo, como hemos comentado, es el generador de flujo de claves aleatorias. Este generador de flujo consiste en que, con una semilla inicial, se produce como resultado una subclave distinta y pseudoaleatoria en cada iteración del programa. Por lo tanto, estas subclaves no se van a repetir en el flujo, o eso buscamos, ya que en realidad no es así. Estos generadores de claves aleatorias poseen un límite en concreto que provoca que el flujo de subclaves aleatorias acabe repitiendo alguna subclave antes del infinito. Es en este valor, que llamamos periodo, donde se encuentra la verdadera robustez del cifrado de flujo y que un criptoanalista deberá de estudiar y encontrar para realizar un ataque correcto. Y, si lo comparamos con Vigenere, este periodo p se podría considerar como el tamaño n de la clave utilizada, ya que a partir del periodo p se vuelve a repetir el flujo de claves, como ocurría en Vigenere. Por lo tanto, es muy probable

que con el mismo criptoanálisis mediante Kasiski o el Índice de Coincidencia se pueda obtener el tamaño del periodo y, posteriormente, cada una de las subclaves del flujo (esto si el texto que tenemos es muy grande y el periodo del flujo es pequeño, ya que si se usa un buen generador de flujo el periodo puede que sea bastante grande e implique que sea muy difícil de atacar).

En nuestro programa, como queríamos que el periodo se viese claro y no fuese muy difícil de criptoanalizar, hemos utilizado como generador de subclaves aleatorias el cifrado Afín. Para este cifrador hemos generado una semilla en concreto con todos sus números primos para tratar de aumentar el periodo ($a = 23$, $b = 7$, $m = 101$ y subclave inicial $k_0 = 3$). Con esta semilla, simplemente en el bucle de cifrado y descifrado hemos llamado al generador de flujo, pasándole la k_i de la iteración anterior (en la primera iteración se usa k_0), y hemos recuperado la clave k_i generada para aplicar el desplazamiento del carácter que le toca cifrar/descifrar en este instante. Puede ver este generador de flujo en la siguiente imagen:

```
int flujo_claves_pseudoaleatorias(int x){
    int a = 23, b = 7, m = 101; /* Semilla */
    int k; /* Clave generada */

    k = (a*x + b) % m;

    return k;
}
```

Se puede ver claramente como la ecuación de dentro es la misma que se aplica en el cifrado de Afín, pero hemos cogido una semilla en concreto para todos los cifrados. Y, cuando toque descifrar, gracias a que la semilla es la misma, el flujo de subclaves será exactamente la misma y se podrá descifrar sin problemas el texto cifrado.

Antes de mostrar las correctas ejecuciones del programa que hemos creado, queremos mostrarle visualmente este periodo del que hablamos. Si utilizamos un texto plano con el mismo carácter “AAAA...”, al cifrar se podrá observar cómo la misma secuencia cifrada se repite cada periodo p . Es decir:

```
X541@LAPTOP-DanMat27 /c/users/x541/desktop/cripto/p1
$ ./flujo -C
# Ha elegido cifrar #

Introduzca lo que quiere cifrar:
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Cifrando AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Texto cifrado: YMVSBNCDQE FUNSQWAHPHZRBGSJMDAOBCOAWHRJLIEPXXFNDYMVSBNCDQE FUNSQWAHPHZRBGSJMD
```

Se ve que el periodo $p = 50$ en este caso. Entonces, en el criptoanálisis habría que buscar este valor p , que es parecido a buscar el tamaño n de la clave en Vigenere.

A continuación, ponemos unas capturas de las ejecuciones correctas de este programa al cifrar y descifrar el fragmento de Don Quijote que hemos utilizado en apartados anteriores:

```
X541@LAPTOP-DanMat27 /c/users/x541/desktop/cripto/p1
$ ./flujo -C -i don_quixote.txt -o entrada_descifrar.txt
# Ha elegido cifrar #

Ciframos el fichero de entrada: don_quixote.txt
Guardamos el texto cifrado en el fichero de salida: entrada_descifrar.txt

X541@LAPTOP-DanMat27 /c/users/x541/desktop/cripto/p1
$ cat entrada_descifrar.txt
ATVHUIE K
ZXMHB GJUWTZ DM AGV DNSAMFTST BBD
LBIBFQXP DC QMR IYYJMT KRPWBIRUA VEJ
QBXEVSV PL DJ YDNQJB
WN W CZUWIKB DC IF ZDLOCS, ULR PDCI TZ JZYHH P WHCD EP JWBUUE HQ DOLH AF
VTvh, QWBOJ YLTQY FPX YQQW WNHPW EJE VU AONJF MMWFOEAGO HHWA BNPX E IPKZJ VQ RTZ
DBRPQ-UQGP, UA GBZ BBRRSDI, B RWJZ KAQM, BBD W NINJPSRCA CTE FMGMKJRT. CQ EPOU BX
HWTOTY TNIF HWNR WHOP NITPVE, J DIPXS LK RBVR ZDYIXF, UFHEUM BF INTBGKHXJ, MKFCUOS
CP GFIZHPB, LVH X EFDJBQ MD NG FBGTD ER XOAQSUS, TPKL ZNBE ORFK TVTFS-QQHICPZW
LU EFX VQAHH. ULR THIX TZ VL MANA XU H CFVHDNF RF TKOS CHVKQ LVH STISJG EPQZUIIF
CQT WMIRK JK MHIJO EFS NGUUGAMU, XVIHL FW HMIH-SVXU UH KMYW B FECYU JNAHJU EN
OXZ IDJU NGVQVPIP. IS HWK ZW SQW EDRPJ N KMGNWLIRRHH TFMG XENTF, P UPDTF AFMOU
TKGOHY, WUU J WIH CDO QMR IGQGV BRQ ODHQJN-CDQYE, DWV BRVE ZG BMGDZG UVE DHTT LA
ABAII XX UDLPGW ULR DLBP-MIBC. JDE HVL VE KIOK PQQTZGNON KM FDCA AXH YLWQHPUY PR
SKIJC; MY JSI KF H WHYCP IGTRF, VPOTF, UAUQK-OPIXRGB, F IHPK ZSSPL TLIW UAV Q CRLPA
ZOFSZKVMQ. TVGZ KIHS YJGM MQ WFP XHULMHW XEF SXYBFNX GH MULHHKZ (WPX ZNDH TVGSS
IO ZFVP LMCUBOJAFC AA GQMAKRD ERIAY JDE HJAONIT CZX IUIHG PB TDL JDMRIZI), XIYURSSC XSSZ THQWTHNTBA CVCQLBKWXB UW SSGNG PHHZW EPEQ WB TFF FYXGWE UHGAQRF.
```

```
X541@LAPTOP-DanMat27 /c/users/x541/desktop/cripto/p1
$ ./flujo -D -i entrada_descifrar.txt
# Ha elegido descifrar #

Desciframos el fichero de entrada: entrada_descifrar.txt

Texto descifrado:
CHAPTER I
WHICH TREATS OF THE CHARACTER AND
PURSUITS OF THE FAMOUS GENTLEMAN DON
QUIXOTE OF LA MANCHA
IN A VILLAGE OF LA MANCHA, THE NAME OF WHICH I HAVE NO DESIRE TO CALL TO
MIND, THERE LIVED NOT LONG SINCE ONE OF THOSE GENTLEMEN THAT KEEP A LANCE IN THE
LANCE-RACK, AN OLD BUCKLER, A LEAN HACK, AND A GREYHOUND FOR COURSING. AN OLLA OF
RATHER MORE BEEF THAN MUTTON, A SALAD ON MOST NIGHTS, SCRAPS ON SATURDAYS, LENTILS
ON FRIDAYS, AND A PIGEON OR SO EXTRA ON SUNDAYS, MADE AWAY WITH THREE-QUARTERS
OF HIS INCOME. THE REST OF IT WENT IN A DOUBLET OF FINE CLOTH AND VELVET BREECHES
AND SHOES TO MATCH FOR HOLIDAYS, WHILE ON WEEK-DAYS HE MADE A BRAVE FIGURE IN
HIS BEST HOMESPUN. HE HAD IN HIS HOUSE A HOUSEKEEPER PAST FORTY, A NIECE UNDER
TWENTY, AND A LAD FOR THE FIELD AND MARKET-PLACE, WHO USED TO SADDLE THE HACK AS
WELL AS HANDLE THE BILL-HOOK. THE AGE OF THIS GENTLEMAN OF OURS WAS BORDERING ON
FIFTY; HE WAS OF A HARDY HABIT, SPARE, GAUNT-FEATURED, A VERY EARLY RISER AND A GREAT
SPORTSMAN. THEY WILL HAVE IT HIS SURNAME WAS QUIXADA OR QUESADA (FOR HERE THERE
IS SOME DIFFERENCE OF OPINION AMONG THE AUTHORS WHO WRITE ON THE SUBJECT), ALTHOUGH FROM REASONABLE CONJECTURES IT SEEMS PLAIN THAT HE WAS CALLED QUEXANA.
```

4. Producto de criptosistemas permutación

En este apartado nos pedían implementar un cifrado por permutación doble. Primero el texto claro se divide en bloques de cifrado que son matrices de $M \times N$, M filas y N columnas. Luego cada bloque, es decir, cada matriz sufre una doble permutación y la unión de los resultados de estas dobles permutaciones es el texto cifrado el cual podremos descifrar.

Además del texto claro, nuestro programa recibe la clave secreta que está constituida por dos vectores k_1 y k_2 . Con k_1 hacemos la permutación por filas, es decir, el tamaño de k_1 es N el número de columnas; y con k_2 hacemos la permutación por columnas, el tamaño de k_2 es M el número de filas. Por ejemplo si el texto plano es ABCDEFGHIJKL, $k_1 = 2341$, y $k_2 = 231$ el bloque de cifrado quedaría de la siguientes manera:

$$M = 3 \text{ y } N = 4$$

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

El resultado de la primera permutación por filas con $k_1 = 2341$, permutamos cada fila con k_1 , quedaría:

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

Y el resultado de la segunda permutación por columnas con $k_2 = 231$, permutamos cada columna con k_2 , quedaría:

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

Este sería el resultado del bloque cifrado, que en este caso como no hay más bloques el resultado del texto cifrado es FGHEJKLIBCDA.

Tras esta explicación pasamos a explicar como lo hemos implementado.

Está implementado en los ficheros **permutaciones.c** y **permutaciones.h**.

El programa permutaciones acepta los siguientes parámetros:

```
./permutaciones {-C|-D} {-k1 K1 -k2 K2} [-i filein] [-o fileout]
```

Obligatorio:

{-C|-D} → indica si se quiere cifrar -C o descifrar -D.

{-k1 K1 -k2 K2} → K1 es el vector utilizado para permutar las filas y K2 para permutar las columnas.

Opcional:

[-i filein] → fichero de entrada con el texto cifrado si se quiere ha elegido la opción -D o el texto plano si se ha elegido la opción -C.

[-o fileout] → fichero de salida donde se escribe el resultado de cifrar -C o de descifrar -D.

Ejemplo de ejecución:

```
laura@Universidad:~/Escritorio/cripto/P1$ ./permutaciones -C -k1 2341 -k2 231
# Ha elegido cifrar #
Introduzca el texto:
ABCDEFGHIJKLM
FGHEJKLIBCDA
laura@Universidad:~/Escritorio/cripto/P1$ ./permutaciones -D -k1 2341 -k2 231
# Ha elegido descifrar #
Introduzca el texto:
FGHEJKLIBCDA
ABCDEFGHIJKLM
```

Vamos a imprimir el proceso para comprobar que se hace correctamente:

```
laura@Universidad:~/Escritorio/cripto/P1$ ./permutaciones -C -k1 2341 -k2 231
# Ha elegido cifrar #
Introduzca el texto:
ABCDEFGHIJKLM
Resultado de la primera permutacion:
n = 4
m = 3
B C D A
F G H E
J K L I
Resultado de la segunda permutacion:
F G H E
J K L I
B C D A
FGHEJKLIBCDA
laura@Universidad:~/Escritorio/cripto/P1$ ./permutaciones -D -k1 2341 -k2 231
# Ha elegido descifrar #
Introduzca el texto:
FGHEJKLIBCDA
Resultado de la primera permutacion:
n = 4
m = 3
B C D A
F G H E
J K L I
Resultado de la segunda permutacion:
A B C D
E F G H
I J K L
ABCDEFGHIJKLM
```

Podemos ver que las permutaciones en el cifrado se realizan correctamente, ya que coincide con el ejemplo realizado anteriormente. Además también podemos ver como el descifrado es correcto ya que el resultado de la primera permutación descifrado coincide con el resultado de la primera permutación del descifrado, más adelante explicaremos con detalle como hemos hecho el proceso de descifrado.

Al inicio del programa, al igual que en el resto de programas ya comentados, comprobamos los argumentos pasados por terminal e informamos de error en el caso de que lo hubiera, además de guardar el modo elegido (**TECLADO**, **IFILE**, **OFILE**, **IOFILES**).

Tras leer los argumentos, comprobamos que los vectores que forman la clave secreta k_1 y k_2 sean correctos. Para ello se tiene que cumplir que no se repita ningún número y que además esos números no sean ni menor que 0 ni mayor que el tamaño del vector. Esto lo hacemos en la función *comprueba_claves_correctas*.

Después de comprobar la clave secreta pasamos a leer el texto plano (-C) o cifrado (-D) del fichero de entrada en caso de haberlo, y en caso contrario, del teclado.

A continuación, llega el bucle encargado de ir creando los bloques de cifrado o de descifrado y realizar para cada uno de ellos la doble permutación, hasta finalizar el texto plano o cifrado. Los resultados de las dobles permutaciones se unen para conseguir el resultado del texto cifrado o descifrado.

Los diferentes bloques o matrices son de tamaño $M \times N$, pero para formar estos bloques solo tendremos en cuenta aquellos caracteres que formen parte del alfabeto (65 - 90 en ASCII) dejando el resto de caracteres tal cual en el texto original. Hemos decidido que si los últimos caracteres del texto que quedan por procesar no son suficientes para formar un nuevo bloque, dejamos estos caracteres sin cifrar, y por lo tanto sin descifrar.

Para cada bloque de cifrado, en el caso de haber elegido la **opción de cifrado** -C, pasamos a realizar la primera permutación a las filas con el vector k_1 como en el ejemplo. Tras obtener el resultado de esta primera permutación pasamos a realizar la segunda permutación a las columnas con el vector k_2 , que es el resultado de la doble permutación.

En cambio, en el caso de haber elegido la **opción de descifrado** -D, es necesario calcular la inversa de los vectores. En el ejemplo de antes, $k_1 = 2341$ pasa a ser $k_1 = 4123$ y $k_2 = 231$ pasa a ser $k_2 = 312$. Primero realizamos al bloque la permutación a las columnas con el vector k_2 y tras esto realizamos la segunda permutación a las filas con el vector k_1 obteniendo el resultado de la doble permutación pero inversa.

Tras cifrar o descifrar cada bloque vamos uniendo los resultados en otra variable, que es la que contiene el resultado del proceso.

Mostramos el resultado por pantalla o en caso de haber introducido el fichero de salida lo escribiremos en él.

Vamos a comprobar su funcionalidad con algunos ejemplos:

Ejemplo cifrando un texto pequeño, **entrada_cifrar.txt**, el mismo que en el ejemplo del método vigenere, k1 = 2341 y k2 = 231 :

```
laura@Universidad:~/Escritorio/cripto/P1$ ./permutaciones -C -k1 2341 -k2 231 -i entrada_cifrar.txt -o salida_cifrar.txt
# Ha elegido cifrar #
Leemos el fichero de entrada: entrada_cifrar.txt

Escribimos el resultado en el fichero de salida: salida_cifrar.txt
laura@Universidad:~/Escritorio/cripto/P1$ ./permutaciones -D -k1 2341 -k2 231 -i salida_cifrar.txt -o salida_descifrar.txt
# Ha elegido descifrar #
Leemos el fichero de entrada: salida_cifrar.txt

Escribimos el resultado en el fichero de salida: salida_descifrar.txt
laura@Universidad:~/Escritorio/cripto/P1$ diff entrada_cifrar.txt salida_descifrar.txt
laura@Universidad:~/Escritorio/cripto/P1$
```

permutaciones.c	salida_cifrar.txt
1 UWA UNUAURLLAS POS DEA FUSCIONBLEIUE QENETES NARAPLEE OS MFIC EROH UNREXTT ESODE	
2 S ETTA SFGE ON IEE LUNCFINEL COANA UA DTL FEPLEMO DR TEECHEI NOY LASTO XELVU TIEDEV	
3 OS LIGUSNE ES PERAMAENTI VESMOSATROEES N TUQUE NAR ATIRXTOEAL	
4 US SEO CVINIFIONC DEUS CARACAY L TICSS QAERIT DIOEREFE LU DENOTR CIANMO OUEDPS COR LES DON	
5 SEPTIIOS VSCREJEMELO POR PDEMRS QOECCR TENTO XE UUEL DIPOTEAA SSEA ES	
6 QUE NJUOTO NN CUUNCNADOIE ED UNEDOSI QUSIENTN UEMAN HERONCIEA CNNA UNIDU Y AE CUBRAOD	
7 QAI DOTADE SEN A ILTENNAs MDE STAEIONCACCDONEI RESDE ER S HASICANIVATOMUCL CESO AEN	
8 QUIANOS UE QA TPAS ROCU IR NL TEEFIDTAM IENBXTOERESERIAAINT R EEACTXSAB E	
9 ET QUNMENASISNE ET COEO DT NAE ACLPOD MOSERARRRIELOS NREF COM LAO ELANCien DPOINT TARNUNA	
10 COEORIT YA HIS FIC ICITSEA ERIVICAD O ARA QUE Q AH NOCOMOSEA CYDOS COM	
11 LOS TESNBAS ONEPDEL TEX COSIRRAIVOTO NTS AOARR VAM COR ESNNCACCION DOA LTROPPSORE	
12 UNE SEAUIRGPAR UNDFZANIPRO BREOLA O SDICINN YOEFIDCTEAISTRCAR DEL TEX	
13 CASIRRAAIVOTO NTA CEMPOLIN.EL DICHFTA EE TDXTOERO ES LO DEAY N E.	
14 VIENTUELV SISUIEG LOEPAR METATESNRos	
15	

permutaciones.c	salida_descifrar.txt
1 LAURA UWU UNA DE LAS POSIBLES FUNCIONES QUE TENEMOS PARA LEER UN FICHERO DE	
2 TEXTO ES FGETS ESTA FUNCION LEE UNA LINEA COMPLETA DEL FICHERO DE TEXTO Y NOS LA	
3 DEVUELVE TIENE LOS SIGUIENTES PARAMETROS VEAMOS QUE ES UN TEXTO NARRATIVO CUAL	
4 ES SU DEFINICION Y LAS CARACTERISTICAS QUE LO DIFERENCIAN DE OTROS COMO PUEDEN	
5 SER LOS DESCRIPTIVOS POR EJEMPLO RECORDREMOS QUE UN TEXTO SEA DEL TIPO QUE SEA	
6 ES UN CONJUNTO DE ENUNCIADOS QUE UNIDOS MANTIENEN UNA COHERENCIA Y UNA UNIDAD	
7 QUE COBRA SENTIDO ADEMÁS LA INTENCIÓN DE ESTAS REDACCIONES HA DE SER COMUNICATIVA	
8 EN EL CASO QUE AQUI NOS OCUPA TRAS DEFINIR EL TEXTO TAMBIEN INTERESARIA SABER EXACTAMENTE	
9 EN QUE CONSISTE EL ACTO DE NARRAR PODEMOS REFERIRNOS A EL COMO LA INTENCIÓN DE	
10 CONTAR UNA HISTORIA YA SEA FICTICIA O VERIDICA. AHORA QUE YA CONOCEMOS LOS DOS	
11 COMPONENTES BASICOS DEL TEXTO NARRATIVO VAMOS A ARRANCAR CON ESTA LECCION DE	
12 UNPROFESOR PARA SEGUIR PROFUNDIZANDO SOBRE LA DEFINICION Y CARACTERISTICAS DEL	
13 TEXTO NARRATIVO.LINEA COMPLETA DEL FICHERO DE TEXTO Y NOS LA DEVUELVE.	
14 TIENE LOS SIGUIENTES PARAMETROS	

Funciona correctamente ya que **salida_descifrar.txt** es igual que **entrada_cifrar.txt**.

Ejemplo cifrando un texto grande, **don_quixote.txt**, el mismo que en el ejemplo del método vigenere, k1 = 231 y k2 = 2314 :

```

laura@Universidad:~/Escritorio/cripto/Pi$ ./permutaciones -C -k1 231 -k2 2314 -i don_quixote.txt -o salida_cifrar.txt
# Ha elegido cifrar #
Leemos el fichero de entrada: don_quixote.txt

Escribimos el resultado en el fichero de salida: salida_cifrar.txt
laura@Universidad:~/Escritorio/cripto/Pi$ ./permutaciones -D -k1 231 -k2 2314 -i salida_cifrar.txt -o salida_descifrar.txt

# Ha elegido descifrar #
Leemos el fichero de entrada: salida_cifrar.txt

Escribimos el resultado en el fichero de salida: salida_descifrar.txt
laura@Universidad:~/Escritorio/cripto/Pi$ diff don_quixote.txt salida_descifrar.txt
laura@Universidad:~/Escritorio/cripto/Pi$ 

```

permutaciones.c	salida_cifrar.txt
1 TEP IRHAC	
2 W THEARICHS THETCH F ORAA ARDTECPU	
3 TSIOF SURTH MOAS U FEENGANMD LET	
4 NOTX OEUIQ LFNAAH MAIN	
5 ILVAGLA DE MANCA LFA, H NEMEAHT OF CHII WH AVH DOSIE NEE RALC TLO T	
6 O, DHETINME RD EOTNIVLLO INSE CG NNEOTH SEOOF GE MEE TNTLNATHP E LAKE NCA TNEH IELA	
7 RA-K,CCENAN BDKUOL ERILLE N A A,ACHNDAA , KREGNUFO HOY CRINS. GURON A OA	
8 FLLOATRMO E RERHEEBANHNU TFTOT SALAA, N ODSTONI MNHTGCRSPSA, SON URTAYDSA , SILT	
9 SENLN OAYD, SRIFNDAIGPONEA OR EX RATSO ON DANS,YSU MA WAA WYE DTHIEEROU-TH RTA	
10 O HFRSES IMEO T.NCIE H OT IFESR WTIN A NTEOUD OT FFLEBNEITHOAN CL VDT EREBLVECHEAN	
11 SDSEODH MOTCA TS FOLHDAIR OS,YLETON WH WE AYD HK-E MEA RABDEAE VREWIN IGF	
12 HESB HTS IMEO NE HPUSADHIZ HSIIN USOHO SEU AEEEKPA T SERPORFA IENY,TE CRETW	
13 NDINTENDAA , YADL TRE HFO IEFNDAMA D LKERACL, E-PTHOD EO TUS ADSTH HELEDCKA	
14 WLLEAS AS DLN TEHA E H-HLOKOILB T.GEOAF E HTH ENGLETS IANMOU S ROF ASWDERINRBO OGIFFY;T	
15 NHE OS AFWA HA ITBDYR S,, EAUGARI-NURTD,EEAFA EYRLAERV RY ARD NSEI GA	
16 TPOSEARTSR T.EYHANMI AVH JEL L HTURSAMNS I WEUQADXS A OAESUDA QR(F ERU TER OERHS IOMS	
17 E DERECCENFI OF NIIN OOP MOAHETAU G NHOTHOWRA S RTEITH SEON BJU,)LTACTEOUHROF RMH GASELEBCO NAOJENESRIT TUCSE PL INAMSETH E !	
18 .OWH, RS IVEEF OLI TTLTUTB IETARCPOMTO TRLEAUO I;LLIBE WTEN NHT OUGOO TYAA	
19 TRSAIHS OREBBOROTAOMRH FM TE IH TNUTRE HNLN OGELT IFYO	
20 MU.TSTUW,OTH KN N,ET AHETTH AB NA-EDMVOEGE MAE WNTLNENH HR WEVEES ALE	
21 SUITAE RCHIMA WH(MSY LILLASTOTH ARERO YENDUVEAHI G)SEMP UO TF LEAR BGOKOIND OSHICALV	
22 FY R SHCHUITWAR ARD NOUDVIA TYATHITDHE OSM ETAL TIN NYGLEELRCTEHETPU D ESUROF	
23 HI T P-S-DORPELST, EDENAN TH NAAEMG MENEHI PSOF OPR; YNDARTEH H C PASO STIDHI	
24 H ESNERS SGEEANDAATFATUN INITH T AGO E H MDNYAOLSAN E RF OAC ILTLAED NAGLOT BYOKOBU	
25 OSIVHLRA CF TYADE A, ROD NGHU HTROBMEOMA Y NAS F O AM HSNET CE	
26 DETGULO B.F OLLAT UTH WE E RREEONN LIKEI HE SDLLEAS WOTH OF TH SEO FES UELFMACII	
27 D SENOAIVIS OMCOAOASPNO TII TR LRCIUEIHITD SFYLT OY AEOMCLIPD NATCONCEID ES T	
28 ES AERWEAPIN HT LSP SS, TARPGHIICITYRWLH LAUN FISHRE N IDIAE HAMCG N UECO RTUONPHISAN	
29 CDSPTA W,ERHLSE HETEF FN OEUNOSAGEA PD LS@ THKEI REN OF QASEHETEARONSUN WI WH	
30 CHIHTMY SOA INRE ASCTID EFLFO SENK MSEAW RYN OHATASE WTR E SOATHI INMUR ARMU	
31 T BRAUEOUY;TOR AG CEDINATOE H @,IGHAVENSE HH T, OF YO ATHR UNIY TIVDIVDY LORFNEIIFTOUYI	
32 YH TST RSAHET R,R EQUYNDDE INV OGERS TFESDRTEE HYO REGTNAR USSERSSESVD@ VEO CR	
33 @NCO OS TTITEISHT RHETSPO ENGLETR DANMT SISHLO WI AN UDS,TEDSL AETO AKWTRSVII	
34 EG NNDRUSEO TANTEMAN TD WDTH MERMOANEOU OTNGI TF W;ATHEMHAR TLO	
35 ESTIIMH CFUOELS NDAVH MET ODEA OT EROU TRXD EADHTAHE E MO TCO IFLAIG FN AER O	
36 TPESHATIACRPUSEO PL H.S AOTN WEAT ELSYAAI AB TH WEUTOUNOHIMH CS DONDITALSNE	
37 GAN TDVEAKOKAESAEL B,IT MEE TDSE HOHAT, TM IREGS AERWT A TEURSEGE HS NHA	
38 DHOWURCIMH H,D E MEHA E VSTUADH FSCEAHI AN DYOCO BDERVLLAOV D ER E SHAMEITW ASSC	
39 RSADN H.MMONDE CED,EEWWR,EHO TH THUR@ AE@ OY EFWA DINISHBO G NK O THENITWPR	
40 E SF OMIOHATTENMR ITABNDVANTEE LREUD NANM A, AYE MASWTI HE PTMD ETE O T	
41	
196	
197 'D ELFE F,TSIR OEESEDMT ITO IMPOMHI SISTO RE LEBIELIMHELSE V WFUTOMA THIINKMEONO SGSEID NE H A,ROGHI TSNDUETED NQUS AHEZEHI	
198	
199	
200	
201	
202	
203	
204	
205 ICASTI HEUNO HD FEMSIRE IELLFEEDVTH BEOF RDUHAT HTN ED AN EIMHIVGSO H CISDMU OMC. TUTBORFAS	
206	
207	
208 ITRIN ERHFUNGISM TO @ SAE,NCHNT ITRS, OKEI TEOUH MSAR GNEAR IT FTEAR."	
209	

```

permutaciones.c           salida_descifrar.txt

1 CHAPTER I
2 WHICH TREATS OF THE CHARACTER AND
3 PURSUITS OF THE FAMOUS GENTLEMAN DON
4 QUIXOTE OF LA MANCHA
5 IN A VILLAGE OF LA MANCHA, THE NAME OF WHICH I HAVE NO DESIRE TO CALL TO
6 MIND, THERE LIVED NOT LONG SINCE ONE OF THOSE GENTLEMEN THAT KEEP A LANCE IN THE
7 LANCE-RACK, AN OLD BUCKLER, A LEAN HACK, AND A GREYHOUND FOR COURSING. AN OLLA OF
8 RATHER MORE BEEF THAN MUTTON, A SALAD ON MOST NIGHTS, SCRAPS ON SATURDAYS, LENTILS
9 ON FRIDAYS, AND A PIGEON OR SO EXTRA ON SUNDAYS, MADE AWAY WITH THREE-QUARTERS
10 OF HIS INCOME. THE REST OF IT WENT IN A DOUBLET OF FINE CLOTH AND VELVET BREECHES
11 AND SHOES TO MATCH FOR HOLIDAYS, WHILE ON WEEK-DAYS HE MADE A BRAVE FIGURE IN
12 HIS BEST HOMESPUN. HE HAD IN HIS HOUSE A HOUSEKEEPER PAST FORTY, A NIECE UNDER
13 TWENTY, AND A LAD FOR THE FIELD AND MARKET-PLACE, WHO USED TO SADDLE THE HACK AS
14 WELL AS HANDLE THE BILL-HOOK. THE AGE OF THIS GENTLEMAN OF OURS WAS BORDERING ON
15 FIFTY; HE WAS OF A HARDY HABIT, SPARE, GAUNT-FEATURED, A VERY EARLY RISER AND A GREAT
16 SPORTSMAN. THEY WILL HAVE IT HIS SURNAME WAS QUIXADA OR QUESADA (FOR HERE THERE
17 IS SOME DIFFERENCE OF OPINION AMONG THE AUTHORS WHO WRITE ON THE SUBJECT), ALTHOUGH FROM REASONABLE CONJECTURES IT SEEMS PLAIN
18 THIS, HOWEVER, IS OF BUT LITTLE IMPORTANCE TO OUR TALE; IT WILL BE ENOUGH NOT TO STRAY
19 A HAIR'S BREADTH FROM THE TRUTH IN THE TELLING OF IT.
20 YOU MUST KNOW, THEN, THAT THE ABOVE-NAMED GENTLEMAN WHENEVER HE WAS AT
21 LEISURE (WHICH WAS MOSTLY ALL THE YEAR ROUND) GAVE HIMSELF UP TO READING BOOKS OF
22 CHIVALRY WITH SUCH ARDOUR AND AVIDITY THAT HE ALMOST ENTIRELY NEGLECTED THE PURSUIT
23 OF HIS FIELD-SPORTS, AND EVEN THE MANAGEMENT OF HIS PROPERTY; AND TO SUCH A PITCH
24 DID HIS EAGERNESS AND INFATUATION GO THAT HE SOLD MANY AN ACRE OF TILLAGELAND TO
25 BUY BOOKS OF CHIVALRY TO READ, AND BROUGHT HOME AS MANY OF THEM AS HE COULD
26 GET. BUT OF ALL THERE WERE NONE HE LIKED SO WELL AS THOSE OF THE FAMOUS FELICIANO
27 DE SILVA'S COMPOSITION, FOR THEIR LUCIDITY OF STYLE AND COMPLICATED CONCEITS WERE
28 AS PEARLS IN HIS SIGHT, PARTICULARLY WHEN IN HIS READING HE CAME UPON COURTSHIPS
29 AND CARTELS, WHERE HE OFTEN FOUND PASSAGES LIKE "THE REASON OF THE UNREASON WITH
30 WHICH MY REASON IS AFFLICTED SO WEAKENS MY REASON THAT WITH REASON I MURMUR AT
31 YOUR BEAUTY;" OR AGAIN, "THE HIGH HEAVENS, THAT OF YOUR DIVINITY DIVINELY FORTIFY
32 YOU WITH THE STARS, RENDER YOU DESERVING OF THE DESERT YOUR GREATNESS DESERVES."
33 OVER CONCEITS OF THIS SORT THE POOR GENTLEMAN LOST HIS WITS, AND USED TO LIE AWAKE
34 STRIVING TO UNDERSTAND THEM AND WORM THE MEANING OUT OF THEM; WHAT ARISTOTLE
35 HIMSELF COULD NOT HAVE MADE OUT OR EXTRACTED HAD HE COME TO LIFE AGAIN FOR THAT
36 SPECIAL PURPOSE. HE WAS NOT AT ALL, EASY ABOUT THE WOUNDS WHICH DON BELIANIS
37 GAVE AND TOOK, BECAUSE IT SEEMED TO HIM THAT, GREAT AS WERE THE SURGEONS WHO HAD
38 CURED HIM, HE MUST HAVE HAD HIS FACE AND BODY COVERED ALL OVER WITH SEAMS AND
39 SCARS. HE COMMENDED, HOWEVER, THE AUTHOR'S WAY OF ENDING HIS BOOK WITH THE
40 PROMISE OF THAT INTERMINABLE ADVENTURE, AND MANY A TIME WAS HE TEMPTED TO TAKE

```



```

198
199 HE DIED HIMSELF, FOR IT SEEMED TO HIM IMPOSSIBLE TO RELIEVE HIMSELF WITHOUT MAKING SOME NOISE, AND HE GROUNDED HIS TEETH AND SQUEEZED
200
201
202
203
204
205 HE OR DISTURBANCE HE FOUND HIMSELF RELIEVED OF THE BURDEN THAT HAD GIVEN HIM SO MUCH DISCOMFORT. BUT AS DON
206
207
208 ;, SAYING IN A RATHER SNUFFING TONE, "SANCHO, IT STRIKES ME THOU ART IN GREAT FEAR."
209

```

Funciona correctamente ya que salida_descifrar.txt es igual que don_quixote.txt.

Possible criptoanálisis:

Suponiendo que el modelo de seguridad nos permite acceder a la máquina de cifrado, nos permite cifrar cualquier tipo de texto las veces que queramos, es decir, ataque por texto claro escogido.

Por ejemplo si ejecutamos el siguiente programa:

```

laura@Universidad:~/Escritorio/cripto/P1$ ./permutaciones -C -k1 3142 -k2 4213
# Ha elegido cifrar #
Introduzca el texto:
ABCDEABCABCABC
Resultado:
CABDCABCABCABC

```

El texto plano está formado por 4 filas iguales, lo que implica que luego el texto cifrado esté formado por 4 filas iguales también. Esto nos permite deducir el valor de k1 simplemente estudiando una fila. Fila plana es igual a ABCD y cifrada CADB entonces $k1 = 3142$.

Y si por ejemplo ejecutamos este programa:

```
laura@Universidad:~/Escritorio/cripto/P1$ ./permutaciones -C -k1 3142 -k2 4213
# Ha elegido cifrar #
Introduzca el texto:
ABCDEFGHIJKLMOP
Resultado:
OMPNGEHFCADBKILJ
```

Podemos visualizar perfectamente el valor de k2, ya que si inicialmente la fila 4 es MNOP y aplicando la primera permutación con $k1 = 3142$ obtenemos OMPN y ahora esta fila se encuentra en la fila 0 de la matriz, quiere decir que el primer valor de k2 es 4. Y así con todos obtenemos que el valor de $k2 = 4213$.

Es cierto, que es necesario dar con los tamaños de los vectores haciendo múltiples pruebas para que sea posible llegar a criptoanalizarlo, pero una vez obtenidos M y N sería seguir el proceso descrito anteriormente.