



INSTITUTO POLITÉCNICO  
NACIONAL



ESCUELA SUPERIOR DE CÓMPUTO

CRYPTOGRAPHY

---

## Substitution Cipher

---

*Autores:*

González Núñez Daniel Adrián  
Hernández Castellanos César Uriel

*Docente:*

Dra. Sandra Diaz Santiago

**Ingeniería en Sistemas Computacionales**

8 de marzo de 2020

# Índice

<b>1. Introducción.</b>	<b>3</b>
<b>2. Descripción del problema: Cifrado Afin</b>	<b>4</b>
2.1. Descripción de la solución. . . . .	5
2.2. Excepciones. . . . .	14
<b>3. Descripción del problema: Cifrado por palabras clave</b>	<b>15</b>
3.1. Descripción de la solución. . . . .	15
3.2. Salida del programa . . . . .	18
<b>4. Conclusión</b>	<b>18</b>
<b>Referencias</b>	<b>18</b>

## 1. Introducción.

### Cifrado afín

Se le llama algoritmo afín a todo aquel algoritmo mono alfabético genérico que realiza transformaciones afines, esto es, la operación de sustitución se realiza con base en la siguiente expresión:

$$Ci = (a * Mi + b) \bmod n \quad (1)$$

donde:

1.  $Ci$  = carácter cifrado que ocupa la posición  $i$  en el alfabeto
2.  $Mi$  = carácter en claro que ocupa la posición  $i$  en el alfabeto
3.  $a$  = constante de decimación
4.  $b$  = constante de desplazamiento  $n$  = tamaño del alfabeto

Como se puede apreciar, la constante de decimación debe ser  $a_i \neq 1$  ya que de otra forma ( $a=0$ ) se anularía la equivalencia de los alfabetos, la constante de desplazamiento debe cumplir  $1_i \neq b_i \neq n$ , ya que si fuese  $b = 0$  no habría desplazamiento alguno y entonces el resultado sería  $Cripto = Mcl$ , en tanto que si  $b$  fuese  $b_i \neq n$  caería dentro del mismo anillo, por lo que su valor se vería reducido a módulo  $n$ . Con base en estas características los sistemas de cifrado por sustitución se pueden clasificar como sigue:

1. Sustitución por desplazamiento puro: cuando  $a=1$  y  $1_i \neq b_i \neq n$ .
2. Sustitución por decimación pura: cuando  $a_i \neq 1$  y  $b=0$ .
3. Sustitución afín: cuando  $a_i \neq 1$  y  $1_i \neq b_i \neq n$ .

De manera que haciendo la comparación con la expresión con la dada para el algoritmo del César

$$Cifrado : Ci = Mi + 3 \bmod n \quad (2)$$

Se observa que  $a = 1$ ,  $b = 3$ , por lo que corresponde al grupo de algoritmos de sustitución por desplazamiento puro.

Para llevar a cabo el proceso de descifrado es necesario calcular:

$$Mi = (Ci - b) * \bmod n \quad (3)$$

Siendo  $= \text{inv}(a, n)$ , donde el factor de multiplicación  $a$  debe ser primo relativo con el cuerpo  $n$  para que exista el inverso de  $a$ .

### Cifrado de palabras clave

El cifrado de palabras clave utiliza una palabra clave para reorganizar las letras en el alfabeto. Estas letras diferentes se sustituyen por las letras en el mensaje para crear un mensaje secreto. Esa palabra clave es necesaria para descifrar el mensaje secreto.

## 2. Descripción del problema: Cifrado Afin

Considere que usamos el conjunto de caracteres imprimibles en ASCII como el alfabeto con el que se va escribir el texto plano. Considere seguir los requerimientos que se muestran a continuación:

1. Los valores para A y B deben ser elegidos por el usuario. Su programa debe verificar que A es un número válido esto se llevará a cabo utilizando su implementación del Algoritmo Extendido de Euclides.
2. Su programa debe de recibir el texto plano en un archivo de cualquier tamaño (Al menos 5kb) y el texto cifrado debe almacenarse con el mismo nombre que el archivo que contiene el texto plano, pero con extensión afn
3. Su programa debe ser capaz de cifrar y descifrar. También el descifrado debe funcionar con archivos. Considere que debe ejecutar su programa una vez para cifrar y debe volver a ejecutarlo para descifrar.

```
'9p4})1R0}&iRw8p94Hf2-[w4fOF>1484&fF>)8f)&|bmfm14#&O-&4Ap-14)|8\'R|)&|
Sw||R4}&iRw8pRwf41)1Of4Hf8AwRR:4b)YwRw)1Awfpfp4Yw|M:1-A&t-RmY48-bfp4}}
92FAwfpfp480Y:-R-b#Rb14|68D4A4R4Y:-884|&fp4b1-&f[-t41TS-8f-bfp4A-1K8)
ff1w:mf4|f-Ew&i#Rb14|-1f-#4Rb1w[F)R-&iAwfp8-Y4-bfp-84:O:w8p->umRb8f)&
&|Ym[p)&-&OY-m8>1-84)&|t4184b1-Yfp4>14M;-&m48f>41w-|F)14f-:4b-m&|Awfp
w&fp42-[w4f068fp144841w48/)RR-bfp48m1twtw&iY4|w4t)R|1)Y)FY-8f-bfp4Sw||
R4}&iRw8p1-Y)&[48FYm[p14Rwiw-m8)&|84[mR)1>1-84)&|t4184w&[Rm|w&ifp4}&iR
w8pA-1K8-bx-p&c-A41F9p-Y)8j-[[R4t4)&|Y-8f-b; )Hf-&68>1w&f8)RRbw&|fp4w1>
R)[4w&fp4>m:Rw[ )fw-&8Tuwfp-mf}}924|wfw-&8F8fm|O-bY4|w4t)R}&iRw8pf4Hf8A
-mR|p)1|RO:4>-88w:R4T#8wf8&)Y48f)f48F}}92A)8:4im&)8)6[Rm:6F)&|wf14f)w&
8[41f)w&b4)f148-bfp)f4t4&&-ATqfp)8&->p08w[ )RR-[ )fw-&F-14t4&-bbw[4F&->
)w|8f)bb-14|wf-18F:mf:--K8w&fp4\'1wiw&)R241w48)14>m:Rw8p4|w&fp4bw18f>R
Ywf4|84R4[fw-&-bfp4Y)&O>m:Rw8p4|T}|wf-18FAp-14[4wt4&-1-O)Rfw48-14H>4&8
48)&|Ap-)14-&ROt4101)14RO[-YYw88w-&4|:Ofp42-[w4fOF)144&[-m1)i4|f-)>>1-
b-1|k}&iRw8p\\w[fw-&)10[-mR|_m-f4/fp4-&i-w&iA-1K-&fp414tw8w-&-bfp)f\\w
RmY48)t)wR):R4-&[4Y-14ApwR8fY)w&f)w&w&ifp4Y4Y:418pw>|w8[-m&f8T9p-mipfp
```

Figura 1: Texto cifrado

[1]

## 2.1. Descripción de la solución.

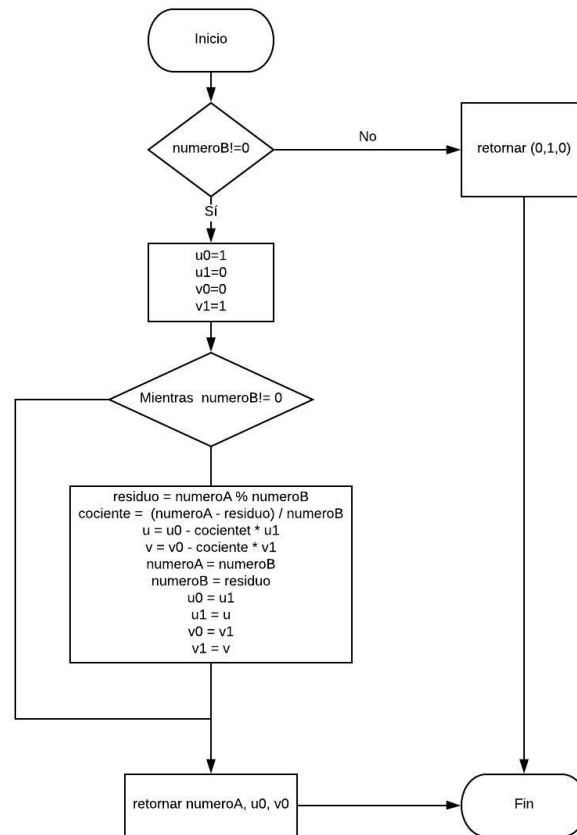


Figura 2: Diagrama de flujo de la función que implementa el algoritmo de Euclides

En la figura *figura 2* es posible observar el diagrama de flujo de la función que implementa el algoritmo extendido de Euclides.

La función recibe como parámetros dos números, que son los cuales deseamos obtener su máximo común divisor y su respectiva combinación lineal.

El algoritmo comienza comprobando que el número B sea diferente de 0, esto para no realizar operaciones de más, en caso de que el número B sea igual a 0 retornaremos directamente 0,1,0 (mcd,u,v), en caso contrario inicializamos cuatro variables que nos serán de ayuda para calcular la combinación lineal.

Posteriormente el bucle while comprobará constantemente que el número B sea diferente de cero, en el caso que el número B sea 0 es un indicador de que el algoritmo ha terminado (residuo igual a cero)

Primeramente se calcula el residuo entre el número A y el número B, para esto se utiliza el operador módulo que se encuentra implementado en la mayoría de los lenguajes de programación.

Como segundo paso se calcula el cociente, para esto simplemente despejamos el cociente, ya que es el único dato que no conocemos de la siguiente expresión:

$$\text{Dividendo} = \text{Divisor} * \text{Cociente} + \text{Residuo} \quad (4)$$

Seguidamente nos auxiliamos de una tabla para facilitar el cálculo de u y v, como se muestra en la figura siguiente:

$r$	$c$	$u$	$v$
393		$u_{-1}$	$v_{-1}$
267		$u_0$	$v_0$
126	1	$u_1$	$v_1$
15	2	$u_2$	$v_2$
6	8	$u_3$	$v_3$
3	2	$u_4$	$v_4$

Figura 3: Tabla que nos facilita el cálculo de  $u$  y  $v$

Como se puede ver en la primera tabla hemos colocado los restos y cocientes calculados previamente. En rojo tenemos los coeficientes que cumplen la operación:

$$\text{residuo} = \text{numeroA} * u + \text{numeroB} * v \quad (5)$$

```
def extendedEuclidean(numberA,numberB):
    if numberB != 0:
        u0 = 1
        u1 = 0
        v0 = 0
        v1 = 1
        while numberB != 0:
            residue = int(numberA) % int(numberB)
            quotient = (numberA - residue)/numberB
            u = u0 - quotient * u1
            v = v0 - quotient * v1
            numberA = numberB
            numberB = residue
            u0 = u1
            u1 = u
            v0 = v1
            v1 = v
        return numberA,u0,v0
    else:
        return 0,1,0
```

Figura 4: Implementación del algoritmo extendido de Euclides en Python

En la figura *figura 4* es posible observar la implementación del algoritmo extendido de Euclides, en si no hay mucho que destacar en la implementación, ya que no se hace uso de alguna biblioteca externa.

Es de destacar que esta función juega un papel importante en nuestro algoritmo criptografico, ya que con esta función es posible validar el mcd entre  $a$  y el tamaño del alfabeto, además de auxiliarnos al momento de obtener el inverso de  $a$ .

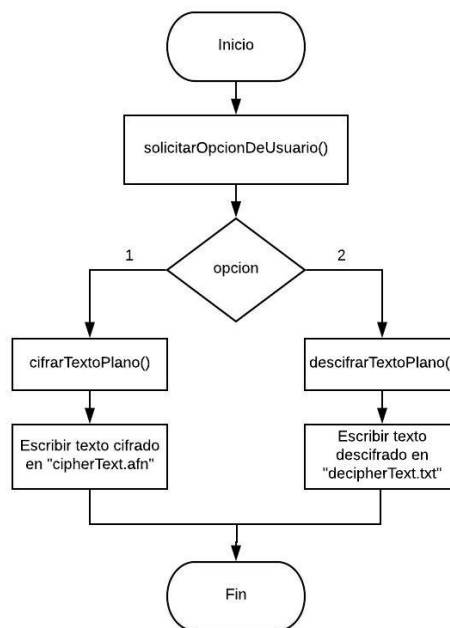


Figura 5: Diagrama de flujo de la función 'testApp()'

En la figura *figura 5* se muestra el diagrama de flujo de la función `testApp()`, la cual tiene como propósito solicitarle al usuario si desea cifrar o descifrar, siendo 1 la opción para cifrar y 2 la opción para descifrar.

Para la opción de cifrar el resultado (texto cifrado) se almacena en un archivo con extensión afn (`cipherText.afn`)

La opción descifrar almacena el resultado (texto descifrado) en un archivo con extensión txt (`decipherText.txt`)

```

def testApp():

    initialFile = "plainText"
    miComando = "cls"

    subprocess.call(miComando, shell=True)
    option = input("1: Encrypt \n2: Decrypt \nInput: ")
    subprocess.call(miComando, shell=True)

    option = int(option)
    if (int(option) == 1):
        encryptOrDecryptFromFile(initialFile, "cipherText", True)
        os.rename('cipherText.txt', initialFile + ".afn")
    elif (option == 2):
        shutil.copyfile(initialFile + ".afn", initialFile + "_.afn")
        os.rename(initialFile + ".afn", "cipherText.txt")
        os.rename(initialFile + "_.afn", initialFile + ".afn")
        encryptOrDecryptFromFile("cipherText", "decipherText", False)
        os.remove("cipherText.txt")

testApp()
  
```

Figura 6: Código fuente de la función 'testApp()'

En la figura *figura 6* es posible observar la implementación de la función `testApp()` en el lenguaje de programación Python.

La función `testApp()` no recibe ningún parámetro, ni tiene retorno valor alguno. La función comienza preguntándole al usuario sobre si desea cifrar o descifrar.

Para ambas opciones se hace una llamada a la función `encryptOrDecryptFromFile()` que recibe como parámetros el archivo inicial, el archivo destino y una bandera que indicará si se cifrará el texto o se descifrá (True y False respectivamente) Finalmente se hace uso de la biblioteca `os` y `shutil` que nos auxilia en el manejo de archivos, esto con el fin de cumplir con los requerimientos solicitados en la práctica.

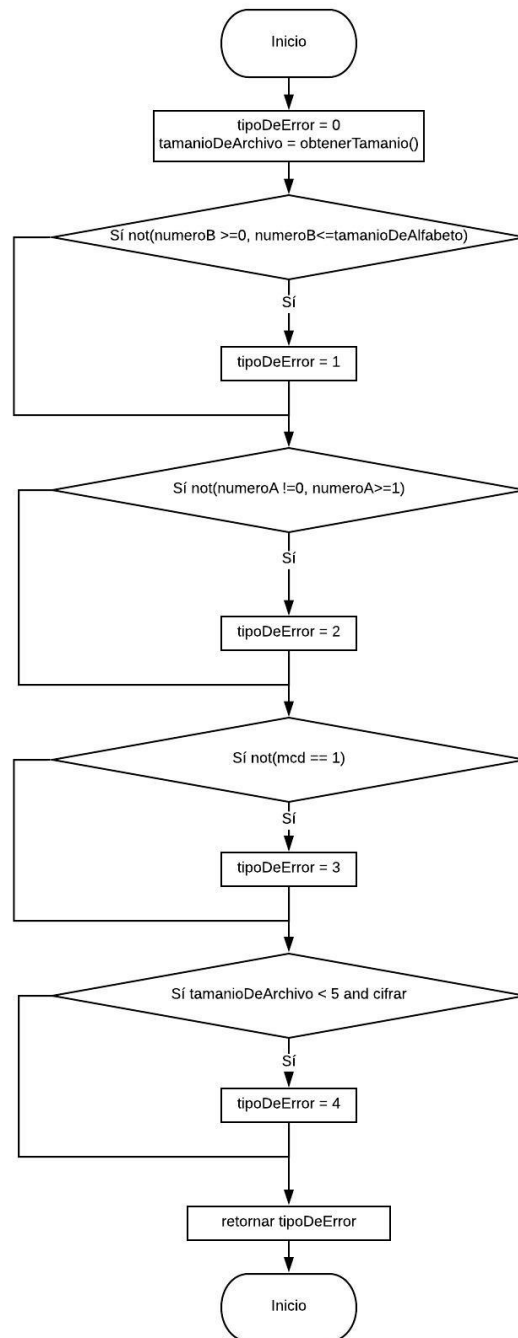


Figura 7: Función encargada de notificar errores



La función que se encuentra descrita en la *figura 7* es la encargada de notificar los errores, derivados de el ingreso de datos por el usuario o por el tamaño del archivo que contiene el texto plano. La función simplemente retornar el error que se este cometiendo.

```
def validateValues(numberA, numberB, sizeAlphabet, mcd, fileSource, type_cf):
    errorType = 0
    factor = 0.001

    fileSize = int(os.path.getsize(fileSource+".txt"))*factor

    if not( numberB >= 0 and numberB <= sizeAlphabet ):
        errorType = 1

    if not( numberA != 0 and numberA >= 1 ):
        errorType = 2

    if not( mcd == 1 ):
        errorType = 3

    if fileSize < 5 and type_cf:
        errorType = 4

    return errorType
```

Figura 8: Función encargada de notificar errores implementada en Python

En la *figura 8* se muestra la implementación de la función que notifica los errores que recibe como parámetros el número A, número B, tamaño del alfabeto, máximo común divisor entre (a,n), url del archivo a cifrar/descifrar y una bandera que nos indica la operación a realizar.

La función comprueban condiciones que se encuentran inherentes en el cifrado afin, como lo es que el número B debe de ser mayor o igual que cero y menor o igual al tamaño del alfabeto; la segunda condición es que el número A debe ser diferente de 0 y mayor o igual que uno; por último se valida que el máximo común divisor entre a y n sea unitario.

Para cumplir uno de los requerimientos de la práctica se aprovechó la función que nos proporciona la biblioteca os (getsize) la cual nos retorna el tamaño del archivo en mb, por lo que lo multiplicamos por un factor de 0.001, esto con el propósito de obtener kb y que sea más claro que se cumple el requerimiento de que al menos se acepten archivos de 5 kb. [?]

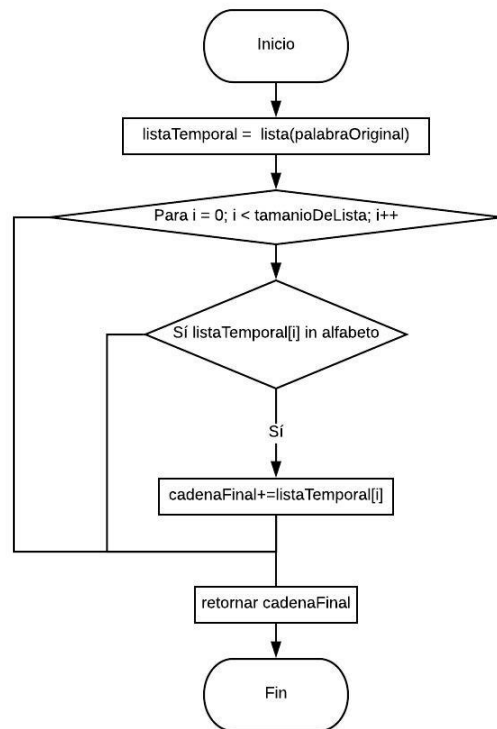


Figura 9: Función encargada de filtrar caracteres que no pertenecen al alfabeto

La función que se encuentra modelada en la *figura 9* se encarga de filtrar cada uno de los caracteres que no pertenezcan al alfabeto (caracteres del código ascii imprimibles), esto con el fin de evitar cualquier tipo de problemas al momento de cifrar.

```

def filterWord(originalWord, alphabet):

    finalString = ""
    temporaryList = list(originalWord)
    for i in range(len(temporaryList)):
        if temporaryList[i] in alphabet.values():
            finalString+=temporaryList[i]
    return finalString
  
```

Figura 10: Función encargada de filtrar caracteres que no pertenecen al alfabeto implementada en Python

En la *figura 10* se muestra la implementación en Python de la función. La función recibe como parámetros una palabra a ser filtrada y el alfabeto, primeramente se declara una lista temporal la cual contiene cada uno de los caracteres de la palabra en forma de lista, se crea un ciclo recorriendo todo la lista y si el carácter i-ésimo se encuentra en el alfabeto pasará a ser parte de la cadena que se va a retornar.

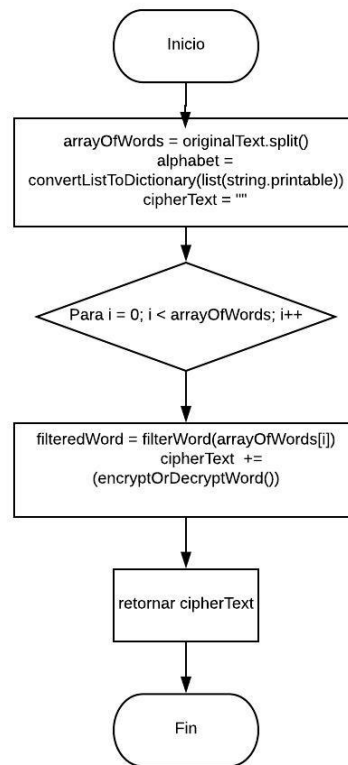


Figura 11: Función encargada de encriptar el texto plano

La función modelada por el esquema anterior, el de la *figura 11*, se encarga de describir el proceso en el cual se obtiene el mensaje cifrado a partir del texto plano del archivo inicial.

```

1  def encryptOrDecryptPlainText(originalText, numberA, numberB, sizeAlphabet, tupleValuesGCD,
   ↪  type_cf):
2      arrayOfWords = originalText.split()
3      alphabet = convertListToDictionary(list(string.printable))
4      cipherText = ""
5
6      for i in range(len(arrayOfWords)):
7          filteredWord = filterWord(arrayOfWords[i], alphabet)
8          cipherText +=
   ↪  (encryptOrDecryptWord(filteredWord, alphabet, numberA, numberB, tupleValuesGCD,
   ↪  type_cf))
9
10     return cipherText
  
```

Figura 12: Función encargada de filtrar caracteres que no pertenecen al alfabeto implementada en Python

En la *figura 12* se muestra la implementación en Python de la función. Esta función recibe el texto original, los valores de la tupla a utilizar (siendo estos A y B), el valor del GCD calculado anteriormente y una bandera llamada typecf que especifica la acción a realizar: si esta es True entonces haremos una encriptación, de lo contrario lo que haremos será desencriptar el mensaje.

La función dividirá el texto original en una arreglo lleno de palabras individuales, el cual recorrerá por cada posición y en cada palabra obtenida lo que hará será conseguir su encriptación individual para así después concatenarlo todo junto en la variable final, que es cipherText

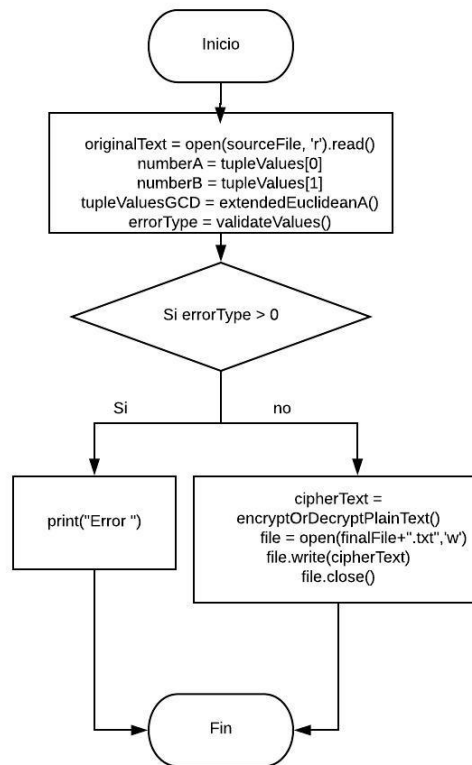


Figura 13: Función encargada de encriptar escribir el texto cifrado en un fichero

El esquema de la *figura 13*, modela el proceso para poder cifrar o descifrar el archivo correspondiente.

```

1  def encryptOrDecryptFromFile(initialFile, finalFile, type_cf):
2
3      sourceFile = initialFile+".txt"
4      originalText = open(sourceFile, 'r').read()
5      tupleValues = askValues()
6      numberA = tupleValues[0]
7      numberB = tupleValues[1]
8      tupleValuesGCD = extendedEuclideanA(numberA, sizeAlphabet)
9      errorType = validateValues(numberA,numberB,sizeAlphabet,tupleValuesGCD[0], initialFile,
10                               ↪ type_cf)
11
12  if errorType > 0:
13      print("Error "+str(errorType)+" has been made, see the documentation")
14      sys.exit()
15  else:
16      cipherText = repr(encryptOrDecryptPlainText(originalText, numberA, numberB,
17                                                    ↪ sizeAlphabet, tupleValuesGCD,type_cf))
18      file = open(finalFile+".txt",'w')
19      file.write(cipherText)
20      file.close()
  
```

Figura 14: Función encargada de cifrar o descifrar el texto en un archivo implementada en Python

En la *figura 14* se muestra la implementación en Python de la función. La función recibe como parametros el

nombre del archivo inicial, el nombre del archivo en el cual escribiremos el texto correspondiente y una bandera que nos indica si cifraremos o descifraremos el texto.

La función abrirá el archivo correspondiente para obtener todo el texto que hay en él. Después, pedirá los valores a usar para la tupla correspondiente ( es decir, los valores A y B usados en nuestro cifrado afin) y calculará el GCD entre A y el tamaño del alfabeto; esto con el objetivo de saber si el valor introducido de A es válido. Validará si es que hay algún error hasta el momento, ya sea un error en el archivo, los valores introducidos o alguna excepción del lenguaje.

Por último, si es que no hubo algún error, obtendrá el texto cifrado por medio de una llamada al método encryptOrDecryptPlainText y escribirá este contenido en el archivo final previamente establecido.

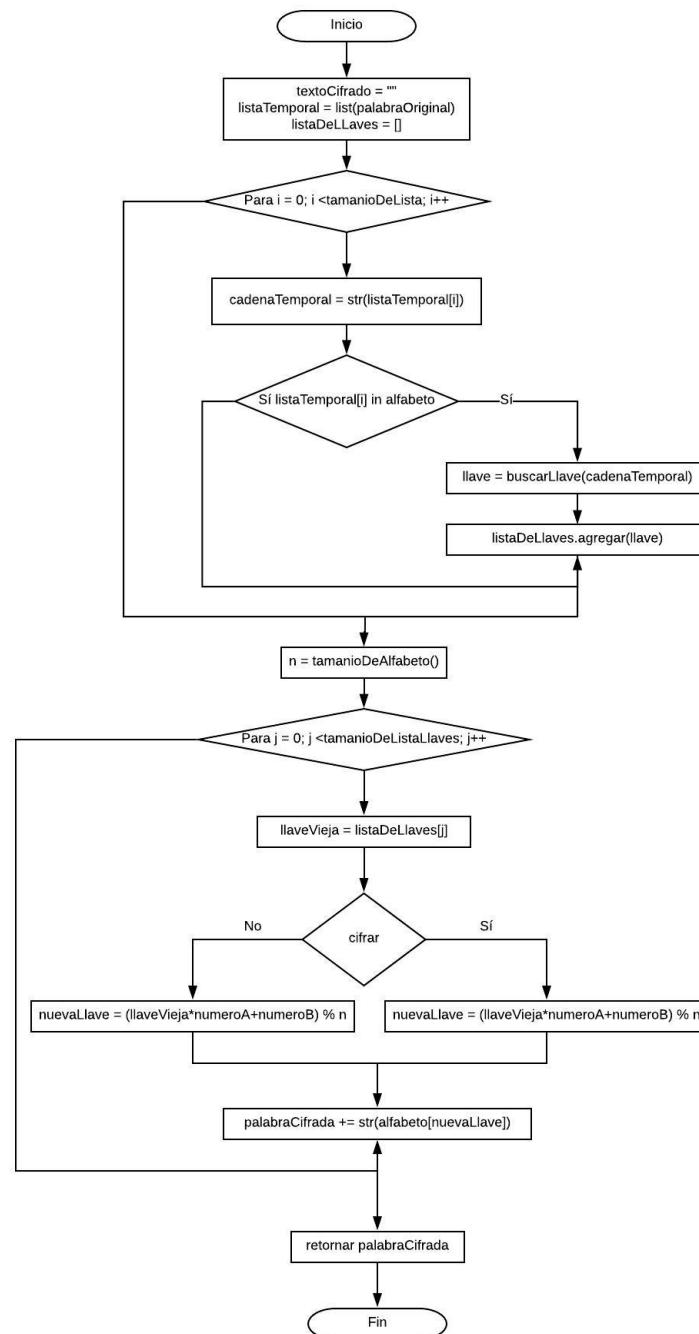


Figura 15: Función encargada de encriptar el texto plano

En la *figura 15* se muestra el diagrama de flujo de la función que tiene mayor importancia. La función recibe como parámetros una palabra, alfabeto, número A, número B, mcd, u, v y una bandera que nos indica si cifraremos o descifraremos.

Primeramente se crea una lista que contiene las llaves de cada uno de nuestros caracteres de la palabra, para posteriormente calcular la nueva llave, esto se hace por medio del método visto en clase, finalmente se retornar el texto cifrado/descifrado

```

1  def encryptOrDecryptWord(originalWord, alphabet,numberA,numberB,tupleValuesGCD,type_cf):
2
3      cipherWord      = ""
4      temporaryList = list(originalWord)
5      listOfKeys      = []
6
7      for i in range(len(temporaryList)):
8          temporaryString = str(temporaryList[i])
9          if temporaryString in alphabet.values():
10             lKey = [key for key, value in alphabet.items() if value == temporaryString][0]
11             listOfKeys.append(lKey)
12
13     for j in range(len(listOfKeys)):
14         n = int(len(alphabet))
15         if type_cf == True:
16             newKey = (oldKey*numberA+numberB) % n
17         else:
18             newKey = (tupleValuesGCD[1]*(oldKey-numberB)) % n
19         cipherWord += str(alphabet[newKey])
20     return cipherWord

```

Figura 16: Función encargada de cifrar o descifrar una palabra

## 2.2. Excepciones.

En la tabla que se muestra a continuación se muestran los posibles errores que se puedan generar a la hora de la ejecución del programa con su respectiva descripción.

Tipo	Descripción
1	El número B es menor o igual a 0 o es mayor que el tamaño del alfabeto
2	El número A es igual a 0 o menor o igual a uno
3	El máximo común divisor entre el tamaño del alfabeto y el número A es diferente de uno
4	El tamaño del archivo a cifrar tiene un tamaño menor a 5 kb

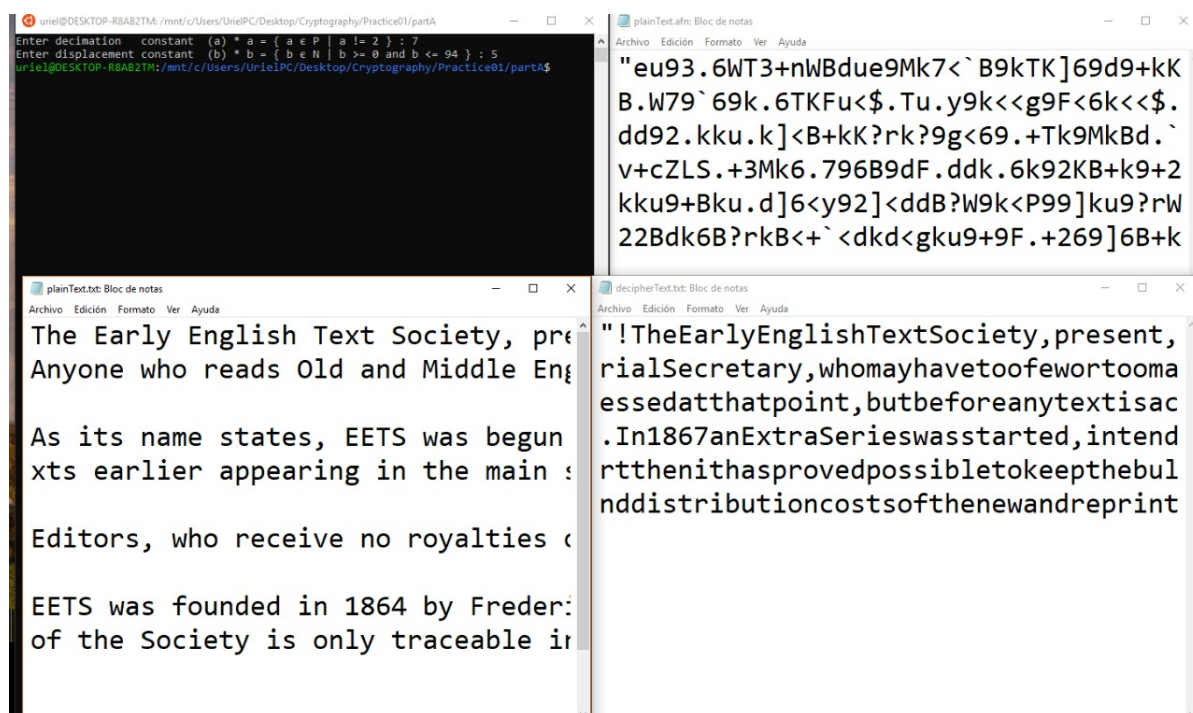


Figura 17: Salida del programa

Se decidió elegir como texto plano un artículo titulado "The Early English Text Society, present, past and future", para posteriormente ejecutar el programa e ingresar como número A 7 y número B 5, lo que nos da como resultado un archivo con extensión afn.

Para descifrar simplemente se selecciona la segunda opción del programa y este nos genera un fichero que contiene el texto descifrado





 App.py	12/02/2019 01:44 a...	Python source file	5 KB
 decipherText.txt	11/02/2019 02:14 a...	Documento de tex...	6 KB
 plainText.afn	11/02/2019 02:14 a...	Archivo AFN	6 KB
 plainText.txt	09/02/2019 06:27 ...	Documento de tex...	7 KB

Figura 18: Archivos generados

### 3. Descripción del problema: Cifrado por palabras clave

En este tipo de cifrado de sustitución lo que haremos será tomar una palabra clave y usarla para las operaciones correspondientes de desplazamiento, siempre basandonos en el modulo de nuestro alfabeto. Si escogemos, por ejemplo, la palabra CIPHER como llave esta tiene las llaves correspondientes  $K = (2, 8, 15, 7, 4, 17)$ .

Después de haber obtenido las llaves correspondientes a nuestra palabra clave, lo que haremos será dividir nuestro mensaje a cifrar en segmentos del tamaño de nuestra palabra clave, y a estos segmentos les sumaremos la llave según el índice con el cual concuerdan. Debemos recalcar que todas estas operaciones se harán modulo el tamaño de nuestro alfabeto, en este caso un alfabeto inglés.

#### 3.1. Descripción de la solución.

Debemos de aclarar que en este caso la única función que cambia con respecto a las del inciso anterior es en la que realizamos todo el proceso de cifrado y descifrado. Las demás funciones en las que leemos los archivos,

obtenemos el alfabeto en un diccionario y validamos los valores, siguen siendo las mismas por lo que no hace falta volver a poner cada una de ellas. Solo nos centraremos en describir el proceso para hacer que este cifrado funcione.

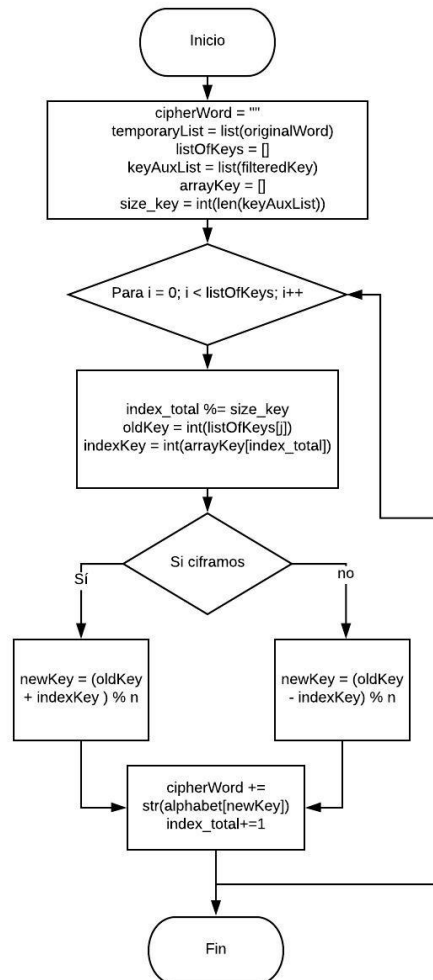


Figura 19: Función encargada de encriptar un archivo

El esquema muestra el proceso en el cual se obtiene el cifrado de una palabra a partir de una llave previamente definida.



```

1  def encryptOrDecryptWord(originalWord, alphabet,filteredKey,type_cf):
2
3      cipherWord = ""
4      temporaryList = list(originalWord)
5      listOfKeys = []
6      keyAuxList = list(filteredKey)
7      arrayKey = []
8
9      for i in range(len(temporaryList)):
10         temporaryString = str(temporaryList[i])
11         if temporaryString in alphabet.values():
12             lKey = [key for key, value in alphabet.items() if value == temporaryString][0]
13             listOfKeys.append(lKey)
14
15         for i in range(len(keyAuxList)):
16             auxKey = str(keyAuxList[i])
17             if auxKey in alphabet.values():
18                 lKey2 = [key for key, value in alphabet.items() if value == auxKey][0]
19                 arrayKey.append(lKey2)
20
21         size_key = int(len(keyAuxList))
22         n = int(len(alphabet))
23
24         global index_total
25         saltamos = False
26
27         for j in range(len(listOfKeys)):
28             index_total %= size_key
29
30             if saltamos == True:
31                 saltamos = False
32                 continue
33
34             oldKey = int(listOfKeys[j])
35             indexKey = int(arrayKey[index_total])
36
37             if oldKey == 85:
38                 siguiente = oldKey = int(listOfKeys[j+1])
39                 saltamos = True
40                 if siguiente == 85:
41                     oldKey = 85
42                 else:
43                     oldKey = 68
44
45             if type_cf == True:
46                 newKey = (oldKey + indexKey ) % n
47             else:
48                 newKey = (oldKey - indexKey) % n
49
50             newKey = int(newKey)
51             cipherWord += str(alphabet[newKey])
52             index_total+=1
53
54         return cipherWord

```

Figura 20: Función encargada de cifrar o descifrar una palabra, implementada en Python

En el código anterior se muestra la implementación en Python de la función. La función recibe como parámetros la palabra a cifrar, el alfabeto a usar, la palabra clave a usar y la bandera correspondiente para saber la operación a realizar.

El proceso inicia con el calculo de los valores de las llaves correspondientes a cada caracter de la llave usada(lineas 15 a 19) y de la palabra a cifrar(lineas 9 a 13). Guardaremos estos valores en un arreglo individual para su uso posterior, ya que haremos sumas entre los valores en los cuales coincidan los indices.

Después iniciaremos un ciclo por el tamaño de la palabra a cifrar. Lo que haremos en cada paso será sumar el valor de la llave de la palabra original en ese índice con el valor que haga un match en posición de la llave a usar. El índice que usaremos para la palabra clave será diferente al cual usemos para recorrer toda la palabra, esto con el fin de que se haga un modulo en cada paso para así nunca desbordar nuestra palabra clave.

Entonces, lo que haremos será sumar los valores de las llaves y tomar el valor modulo el tamaño de nuestro alfabeto para poder así conseguir un carácter válido para anexar a nuestro mensaje final.

### 3.2. Salida del programa

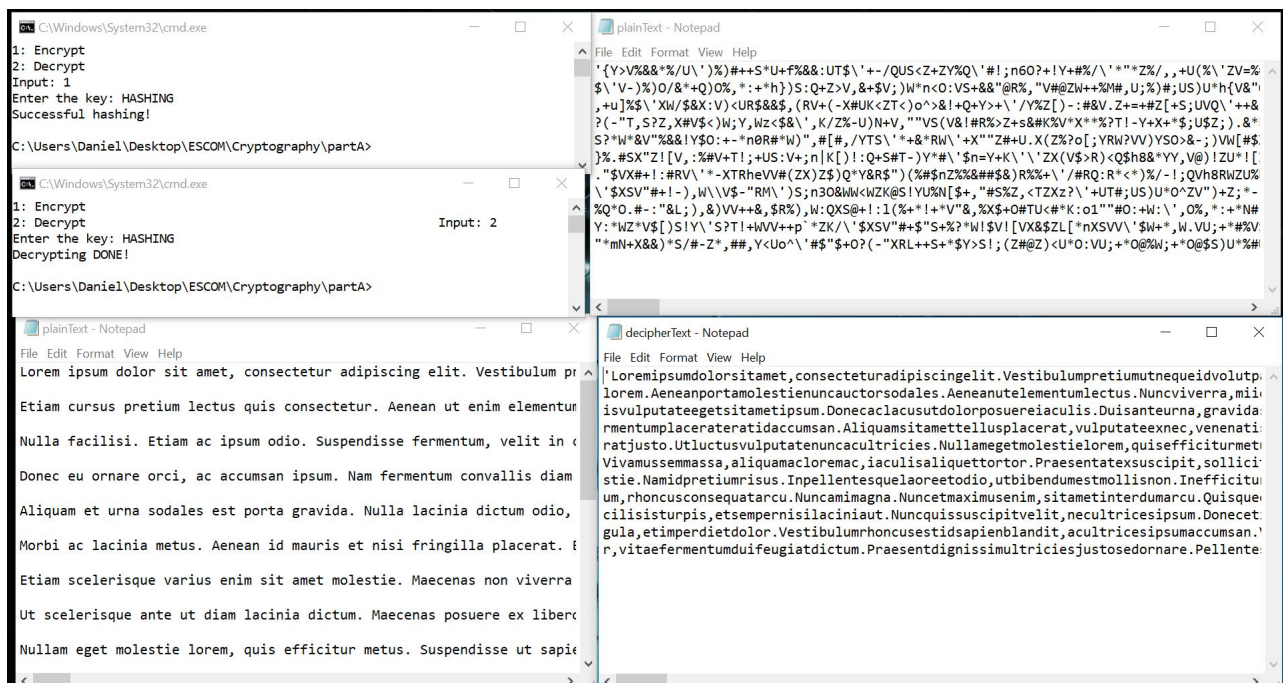


Figura 21: Captura del funcionamiento

## 4. Conclusión

El surgimiento de nuevas redes de comunicación en lo particular el internet, abrió diferentes posibilidades para el intercambio de información, con lo que de manera casi proporcional aumentaron las amenazas de la seguridad de los canales de información. Es necesario entonces, crear diferentes mecanismos que nos "garanticen" la confidencialidad y autenticidad de los datos.

En el presente documento se inicia nuestro recorrido por el mundo de la criptografía, en concreto se revisan dos mecanismos que son el cifrado afín y el cifrado de palabras clave, lo que resulta bastante interesante el conocer como funcionan.

En cuanto a las dificultades que se presentaron en la práctica, fueron casi mínimas, ya que inicialmente se planteaba que se trabajaría con el alfabeto inglés, pero al momento de cambiar al código ascii se tuvieron que realizar diferentes adecuaciones al código fuente con el que se contaba, el problema que se presentaba es que Python separa por caracteres y el problema era que caracteres como el salto de línea los interpretaba como dos caracteres lo que nos creaba un gran problema en el descifrado.

## Referencias

- [1] C. Paar and J. Pelzl, *Understanding Cryptography A Textbook For Students And Practitioners*. Springer; Edición: 2010, 2009.