

|  |  |
| --- | --- |
| C:\Users\DORA\Pictures\BATIZ\th[1].jpg | **I**nstituto **P**olitécnico **N**acional Escuela Superior de Cómputo **ESCOM** |

**CRYPTOGRAPHY.**

**“Práctica Cifradores Clásicos”**

**Abstract:**

Dentro de ésta práctica hablaremos de los cifradores de vigenère, affine y también veremos cómo funciona el algoritmo de Euclides y su variante extendida con algunos ejemplos así como sus códigos de implementación para dar un resultado óptimo en funcionamiento.

**By:**

-Chacón Inostrosa Jaime Enrique.

-Ramírez Olvera Guillermo

-Sánchez Méndez Edmundo Josué

**Date:**

X de Marzo del 2021.

**Professor**:

MSc. Nidia Asunción Cortez Duarte

**Group**:

3CM13.

**Algoritmo de Euclides:**

El algoritmo de Euclides es un método para encontrar de una manera eficaz el máximo común divisor de dos números enteros, este algoritmo se basa en tomar el problema original para hacerlo uno más pequeño hasta hacerlo uno más fácil de resolver, esta propiedad es la siguiente: si se tiene que se debe encontrar el MCD(A,B), representamos a A de la siguiente forma A=B\*Q+R, en donde podemos con el residuo, podemos llegar a lo siguiente MCD(A,B)= MCD(B,R), continuando de la misma forma hasta que el residuo llegue a 0, para que al final podamos decir que el MCD(A,B) es el número que antecede a ese 0 en la posición del residuo.

Ahora para el caso del algoritmo extendido de Euclides se necesita que A y B sean coprimos es decir que MCD(A,B)=1, ahora para la aplicación de este algoritmo se crean nuevas ecuaciones en donde el residuo va a en la parte izquierda de la igualdad y el resto en la derecha de tal forma que se cumpla la igualdad, después se tiene una formula parecida a la siguiente, 1=ax+by, los consientes van a irse reemplazando hasta llegar a A y B, de tal forma que nos dé una combinación lineal de los mismos, lo que a nosotros nos interesa es que x es el inverso multiplicativo de a módulo b, por lo que nos es de mucha utilidad cuando trabajamos con algebra modular al querer encontrar el inverso de un número particular.

**Código del algoritmo de Euclides:**

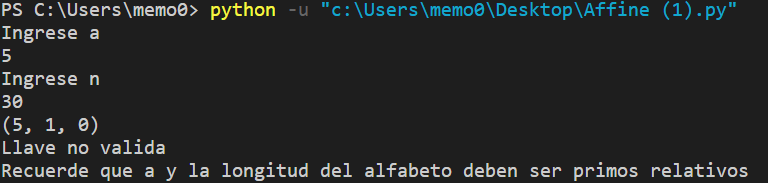
1. **def** mcd(a, b):
2. **if** b == 0:
3. **return** a
4. **return** mcd(b,a%b)

**Código del algoritmo de Euclides extendido:**

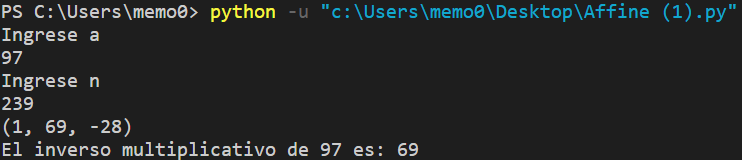
1. **def** extendidoEuclides(a, b):
2. **if** b == 0:
3. **return** 0,1,0
4. u0 = 1
5. **u1 = 0**
6. v0 = 0
7. v1 = 1
8. **while** b != 0:
9. q = a//b
10. **r = a - b \* q**
11. u = u0 - q \* u1
12. v = v0 - q \* v1
13. a = b
14. b = r
15. **u0 = u1**
16. u1 = u
17. v0 = v1
18. v1 = v
19. **return** a, u0, v0

**Capturas de pantalla:**

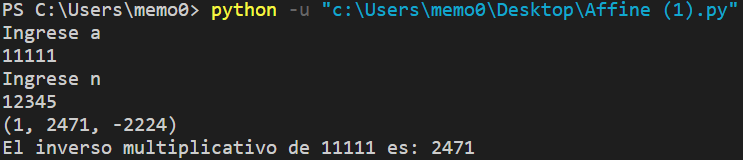
a=5, n=30



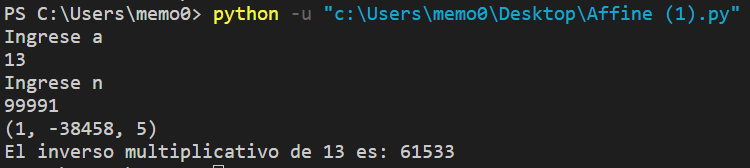
a=97, n=239



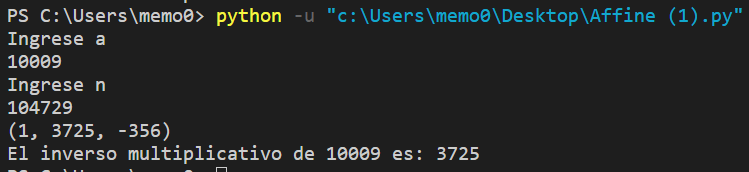
a=11111, n=12345



a=13, n=99991



a=10009, n=104729



**Conclusión:**

La implementación de la practica fue sencilla excepto por algunas circunstancias que sufrimos en la implementación del código ASCII, pero para lo demás se pudo realizar sin mayor problema, reflexionando sobre los cifradores mono alfabéticos y poli alfabéticos, podemos ver una clara mejora en la complejidad de los cifrados, ya que en la época en la que se utilizaban los poli alfabéticos eras súper complejos y seguros, en la actualidad con diversas formar de romperlo como puede ser con el método de kasiski o con un simple análisis de frecuencias, ya no puede ser considerado tan bueno, pero sin duda es un gran ejemplo de la evolución de la criptografía con el paso del tiempo.

**Cifrador Vigenère:**

El cifrado de vigenère es un cifrado simple que se basa en una tabla de “X” x “Y” casilleros, los valores de X y Y deben ser iguales, dichos valores deben de ser dependiendo de la cantidad de letras o caracteres con las que cuente el alfabeto, para este caso nosotros tomaremos el ejemplo del alfabeto inglés el cual consta de 26 letras que van de la “A” a la “Z”, por lo que dichas letras van colocadas en la tabla de 26x26 cada una de ellas tanto en filas como en columnas ,ésta tabla es la que para método demostrativos se utiliza. La primera fila de esta tabla tiene los 26 símbolos de las letras, de la “A” a la “Z”. La segunda fila se corre una posición y se empieza desde la “B” a la “Z” y al final se agrega la “A”. La tercera fila se vuelve a recorrer una posición y se empieza en la “C” para terminar en la “Z”, agregando dos símbolos a la derecha en los espacios sobrantes, en este caso la “A” y la “B”. Y así sucesivamente como se muestra en la imagen 1:

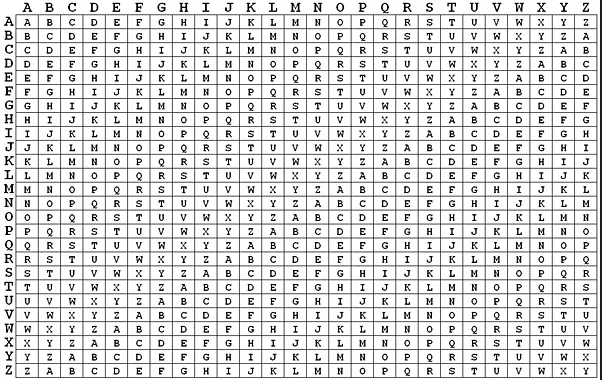


Imagen 1. Cuadro de Vigenère.

Además del texto a cifrar, el esquema Vigenère necesita de manera normal de una llave única, la cual se ha utilizado para poder cifrar o descifrar un mensaje cifrado con éste método, la llave utilizada se repite la cantidad de veces hasta llevar la longitud del texto a cifrar. Debemos de considerar que el tamaño de la llave debe de ser igual o menor al tamaño del mensaje pero de una longitud mayor a cero. Normalmente se elimina los espacios y los símbolos de puntuación. Por ejemplo, si la llave es “morsa”, y el texto a cifrar es “bienvenido al mundo de la criptografía”, entonces tendremos que cifrar:

bienvenidoalmundodelacriptografia

morsamorsamorsamorsamorsamorsamor

Considerando este particular ejemplo, se toma la primera letra del texto a cifrar, así como la primera letra de la llave. Esto nos da la coordenada de la columna y renglón respectivamente. Así encontramos la letra cifrada. Por ejemplo, de acuerdo a la tabla. Si la primera letra es la “b” (de bienvenido) –la columna, la letra de la llave es la “m” (de morsa), que es la fila y obtenemos una “n”. Este proceso lo realizamos para todas las letras y así generamos el texto cifrado.

También una forma más rápida o sencilla es recordar que hablamos de un proceso poli alfabético en el que podría verse como una implementación del algoritmo de corrimiento y la ecuación para cifrar sería obteniendo cada valor de la llave, sumándolo al valor de cada letra para el mensaje a cifrar posteriormente aplicar modulo correspondiente así mismo para descifrar encontrando el inverso aditivo de cada letra por parte de la llave y sumándolo al texto por descifrar para posteriormente aplicar el modulo correspondiente.

**Código del Cifrado:**

1. def encrypt(file,keyS):
2. try:
3. f=open(file,"r")
4. message=f.read()
5. f.close()
6. key = str(keyS)
7. returnMessage = "Cypher made successfully plese check your directory"
9. if(keyS == "777"):
10. key = generatekey(len(message))
11. returnMessage +=" the key generate is ' "+ key +" '"
12. encrypted = ""
13. split\_message = [
14. message[i : i + len(key)] for i in range(0, len(message), len(key))
15. ]
17. for each\_split in split\_message:
18. i = 0
19. for letter in each\_split:
20. number = (letter\_to\_index[letter] + letter\_to\_index[key[i]]) % len(alphabet)
21. encrypted += index\_to\_letter[number]
22. i += 1
24. f = open("encrypt.vig","w")
25. f.write(encrypted)
26. f.close
27. return returnMessage
28. except:
29. return  "Error encrypt"

**Código del Descifrado:**

1. def decrypt(file,keyS):
2. try:
3. f=open(file,"r")
4. cipher=f.read()
5. f.close()
6. key=str(keyS)
7. decrypted = ""
8. split\_encrypted = [
9. cipher[i : i + len(key)] for i in range(0, len(cipher), len(key))
10. ]
12. for each\_split in split\_encrypted:
13. i = 0
14. for letter in each\_split:
15. number = (letter\_to\_index[letter] - letter\_to\_index[key[i]]) % len(alphabet)
16. decrypted += index\_to\_letter[number]
17. i += 1
19. f = open("decrypt.vig","w")
20. f.write(decrypted)
21. f.close
22. return "Decipher made successfully, plese check your directory"
23. except:
24. return "Error Decryption"

**Código de cálculo de la llave:**

1. def generatekey(sizemsg):
2. tamañokey = random.randrange(sizemsg)
3. key = ""
4. for i in range(tamañokey):
5. while True:
6. letra = random.choice(auxkey)
7. if letra!=' ':
8. key += letra
9. i=i
10. break
11. return key

**Ejecuciones:**

Primero tenemos programa que tiene tanto el cifrado como el descifrado según sea el caso necesario para el usuario, en donde puede subir un documento así mismo como puede elegir ingresar una llave conocida o generar simplemente otra nueva, cabe destacar que solo puede hacerse esto en el descifrado como se muestra en la Imagen 2.

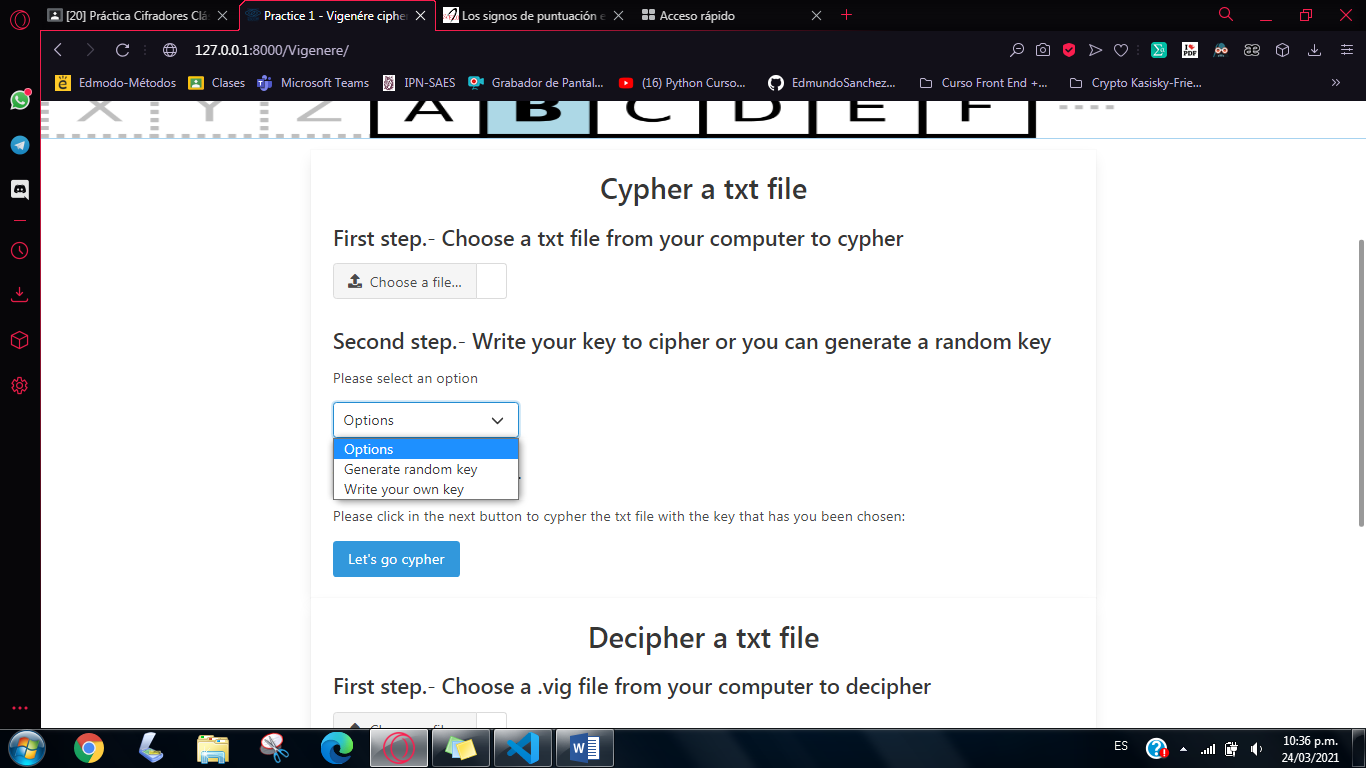


Imagen 2. Programa Vigenere.

Tomamos como muestra el texto que se tiene dentro del archivo correspondiente para poder ser cifrado con extensión .txt como se muestra en la Imagen 3.

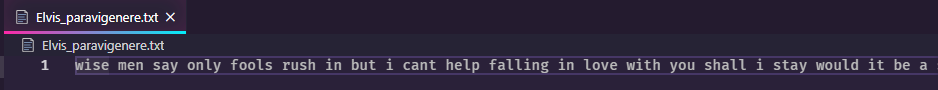


Imagen 3. Texto a Cifrar.

Ahora pasamos a colocar una llave que para el ejemplo toaremos la palabra “beautiful” como vemos en la Imagen 4.

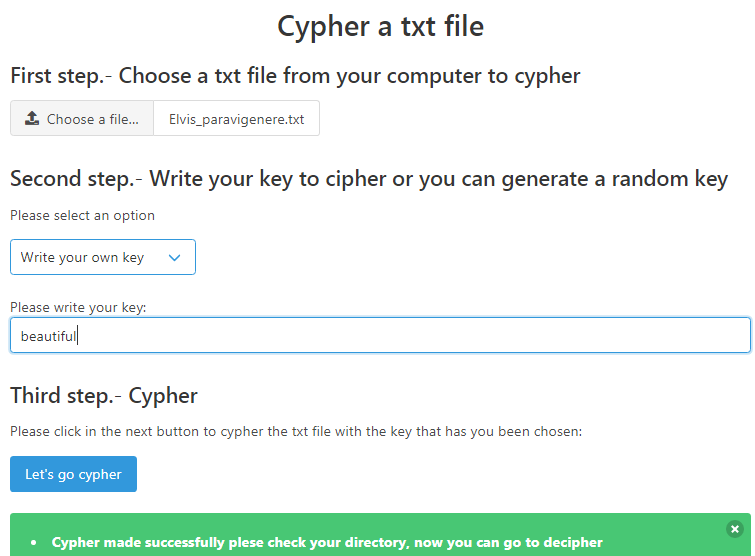


Imagen 4. Ingreso de Llave.

Ahora una vez que damos en cifrar nos genera un archivo .vig en el cual se encuentra nuestro archivo o texto cifrado como vemos en la Imagen 5.

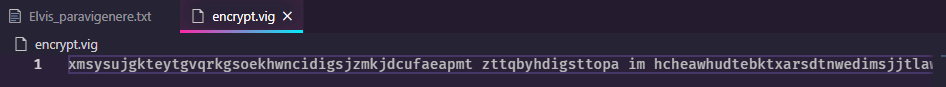


Imagen 5. Texto Cifrado.

Posteriormente vamos a descifrar el mensaje haciendo el proceso parecido al anterior y eligiendo la llave correcta como se muestra en la Imagen 6.

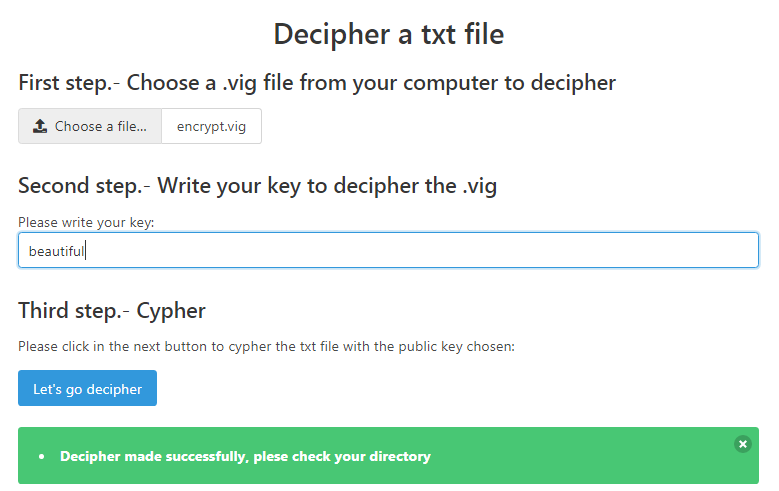


Imagen 6. Descifrado.

Finalmente nos genera otro archivo .vig para poder descifrar nuestro archivo dándonos el resultado de la Imagen 7.

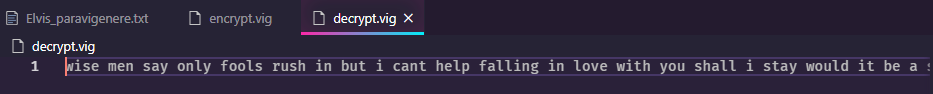


Imagen 7. Archivo descifrado.

Ahora vamos a modificar los primeros 7 caracteres del archivo cifrado anteriormente como se muestra en la Imagen 8.

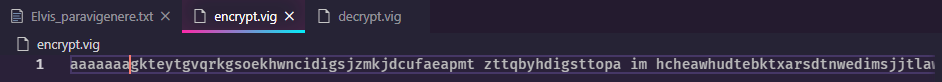


Imagen 8. Modificando archivo cifrado.

Ahora falta repetir el proceso de descifrado para el nuevo texto con las modificaciones realizadas y la misma llave usada anteriormente como se muestra en la siguiente Imagen 9.

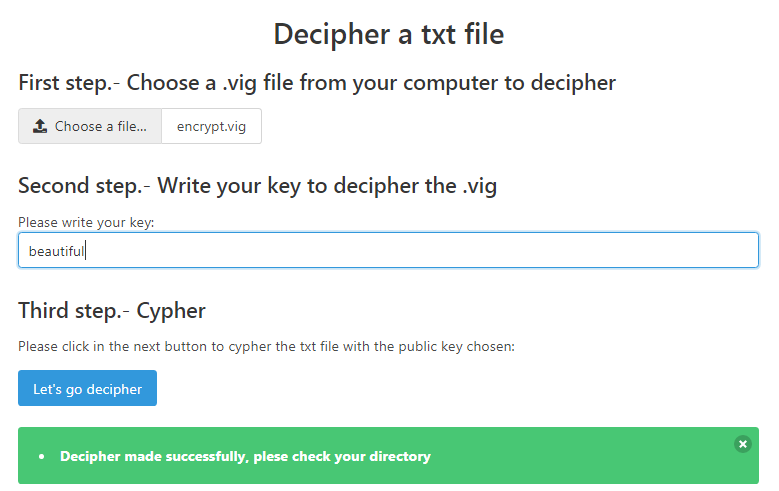


Imagen 9. Descifrado archivo alterado.

Finalmente observamos que se hace el proceso de descifrado pero al momento de ver el resultado es diferente únicamente los 7 caracteres alterados al texto correcto, por lo tanto al alterar el proceso de cifrado también tiene repercusiones al descifrar pero no en todo el texto como se muestra en la siguiente Imagen 10.

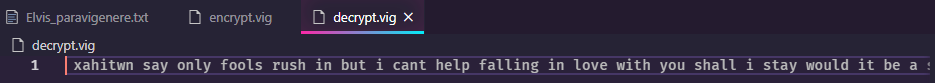


Imagen 10. Descifrado archivo alterado.

**Cifrador Affine:**

El cifrador Affine forma parte de la criptográfica clásica, el cual a ser un algoritmo de sustitución mono-alfabética es de tipo sustitución, podemos decir que es mono-alfabética ya que los caracteres que sean iguales de un texto dado se mapean a otro carácter con base en nuestro factor aditivo y multiplicativo, evidentemente también depende de si será el cifrar o descifrar. Vemos primero la forma de cifrar de Affine.

1. Verificar que nuestro factor multiplicativo α y la longitud del alfabeto sean coprimos, en caso de serlo el algoritmo puede continuar, en caso contrario se deberá buscar un α que si sea coprimo.
2. A cada valor de los caracteres del texto en claro será multiplicado por el valor α. Para poder saber el valor de los caracteres necesitamos buscarlos en nuestro alfabeto y la posición será nuestro valor.
3. A cada valor obtenido en el paso 2, se le sumara el valor del factor aditivo o β.
4. A cada valor obtenido en el paso 3, se le aplicara mod n, siendo n la longitud de nuestro alfabeto.
5. A cada valor obtenido en el paso 3 se buscará el carácter que tenga ese valor en el alfabeto, así ser ira formando nuestro mensaje cifrado.

Ahora por otro lado, para poder descifrar debemos hacer lo siguiente:

1. Calcular el inverso multiplicativo del valor α.
2. Calcular el inverso aditivo de β.
3. Por cada carácter de nuestro texto cifrado se deberá buscar el valor en el alfabeto, una vez obtenido se le suma el valor obtenido en el paso 2.
4. A cada valor del paso 3 se multiplicará por el valor obtenido en el paso 1.
5. A cada valor del paso 4 se3 le aplicara el módulo con base a la longitud del alfabeto.
6. A cada valor obtenido del paso 5 se buscará el carácter que tenga ese valor en el alfabeto, así ser ira formando nuestro mensaje descifrado.

**Código del Cifrado:**

**def** Encrypt(archivo, alphabet, alpha, beta):

f=open(archivo,"r")

texto=f.read()

f.close()

mensaje = "Cypher made successfully plese check your directory"

alfabeto = "abcdefghijklmnopqrstuvwxyz "

bandera = True

esNumero = False

**try**:

numero = int(alphabet)

esNumero = True

**except**:

esNumero = False

**if** esNumero:

alfabeto = [chr(i) **for** i **in** range(int(alphabet))]

bandera = False

**elif** ',' **in** alphabet:

alfabeto = alphabet.replace(',', '')

**elif** alphabet == "ES":

alfabeto = "abcdefghijklmnñopqrstuvwxyz "

**elif** alphabet == "D":

alfabeto= "0123456789"

**elif** alphabet == "ASCII":

alfabeto = [chr(i) **for** i **in** range(256)]

bandera = False

**if**(alpha == 777 **and** beta == 777):

alpha,beta = generateRandom(len(alfabeto))

mensaje += ", alpha value = " + str(alpha) + " beta value = " + str(beta)

**if**(extendidoEuclides(alpha,len(alfabeto))[0]==1 **and** alpha<len(alfabeto)):

textoCifrado = ""

**if** bandera:

**for** p **in** texto:

textoCifrado += alfabeto[((alpha \* alfabeto.find(p) )+ beta) % len(alfabeto)]

**else**:

**for** p **in** texto:

textoCifrado += alfabeto[((alpha \* alfabeto.index(p) )+ beta) % len(alfabeto)]

f = open("encrypt.aff","w",encoding="iso-8859-1")

f.write(textoCifrado)

f.close

**return** mensaje

**else**:

**return** "Alpha value not valid"

**Código del Descifrado:**

**def** Decrypt(archivo, alphabet, alpha, beta):

**try**:

f=open(archivo,"r",encoding="iso-8859-1")

textoCifrado=f.read()

f.close()

alfabeto = "abcdefghijklmnopqrstuvwxyz "

bandera = True

esNumero = False

**try**:

numero = int(alphabet)

esNumero = True

**except**:

esNumero = False

**if** esNumero:

alfabeto = [chr(i) **for** i **in** range(int(alphabet))]

bandera = False

**elif** ',' **in** alphabet:

alfabeto = alphabet.replace(',', '')

**elif** alphabet == "ES":

alfabeto = "abcdefghijklmnñopqrstuvwxyz "

**elif** alphabet == "D":

alfabeto= "0123456789"

**elif** alphabet == "ASCII":

alfabeto = [chr(i) **for** i **in** range(256)]

bandera = False

inversoalpha = inversoMultiplicativo(len(alfabeto),alpha)

minusbeta = InversoAditivo(beta,len(alfabeto))

textoDescifrado = ""

**if** bandera:

**for** c **in** textoCifrado:

textoDescifrado += alfabeto[ (inversoalpha \* (alfabeto.find(c) + minusbeta)) % len(alfabeto)]

**else**:

**for** c **in** textoCifrado:

textoDescifrado += alfabeto[ (inversoalpha \* (alfabeto.index(c) + minusbeta)) % len(alfabeto)]

f = open("decrypt.aff","w",encoding="iso-8859-1")

f.write(textoDescifrado)

f.close

**return** "Decipher made successfully, plese check your directory"

**except**:

**return** "Error Decryption"

**Código para generación aleatoria de valores α y β:**

**def** generateRandom(longitud):

m = []

**for** i **in** range(1,longitud):

m.append(i)

maux = []

**for** i **in** m:

**if**(extendidoEuclides(i,longitud)[0]==1):

maux.append(i)

**return** random.choice(maux),random.randint(0,longitud)

**Código para el cálculo del inverso aditivo:**

**def** InversoAditivo(beta,n):

**return** int(n-beta%n)

**Pruebas:**

Como primer paso tendremos que dirigirnos al apartado Affine cipher (ver imagen 11) de nuestra página en donde encontraremos que se nos ofrece la parte del cifrado y descifrado, con el fin de que el usuario tenga cierta comodidad al usar el programa.



Imagen 11. Inicio Affine cipher.

Empezaremos cifrando el archivo simon\_paraAffine.txt, eligiendo el archivo como en la imagen 12.

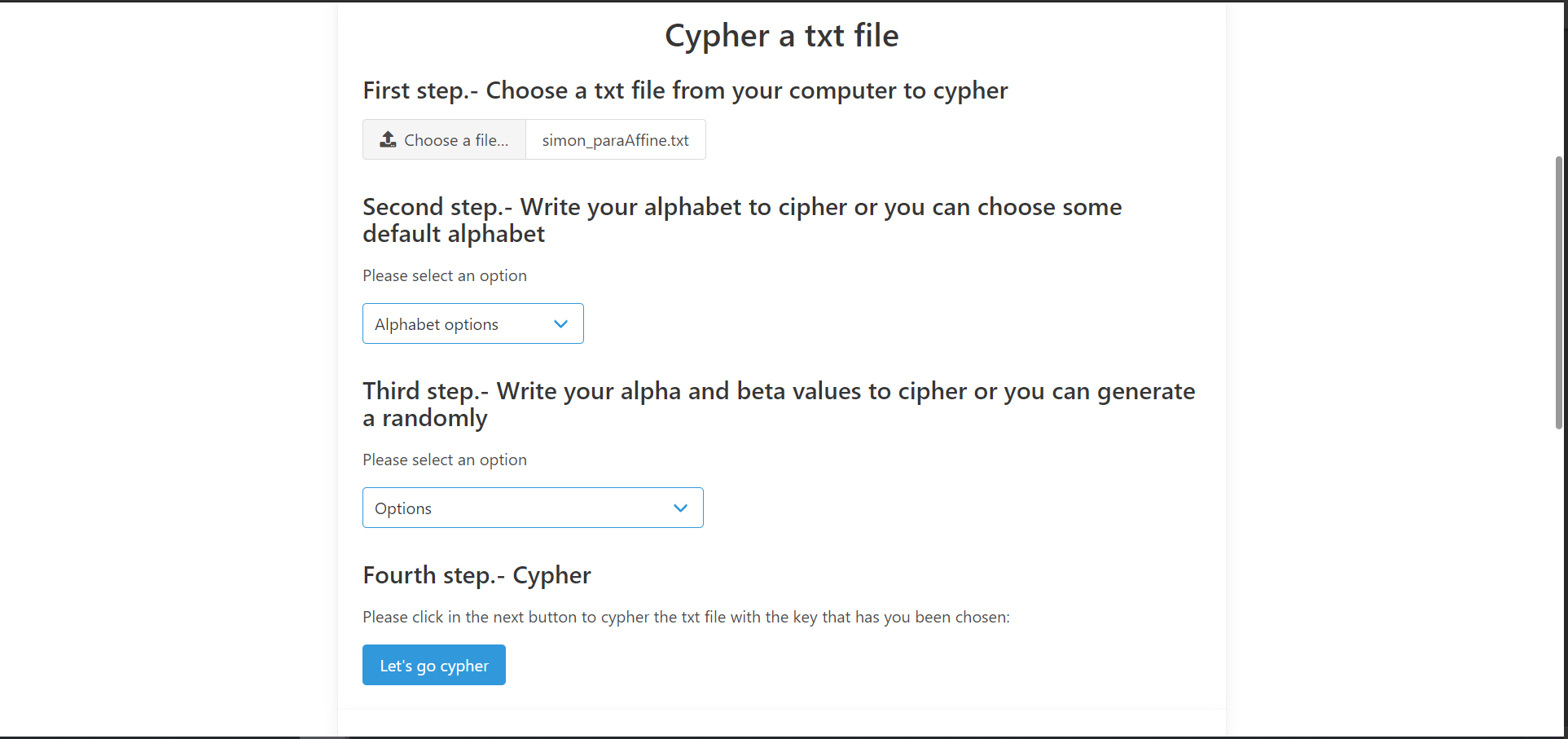


Imagen 12. Eligiendo archivo.

El programa nos da varias opciones para elegir alfabetos, por defecto tenemos la posibilidad de elegir: entre el alfabeto inglés con espacio, alfabeto en español con espacio, ASCII de 256 elementos y dígitos del 0-9 sin espacios, sin embargo, tenemos la opción de escribir nuestro propio alfabeto como se muestra en la imagen 13.

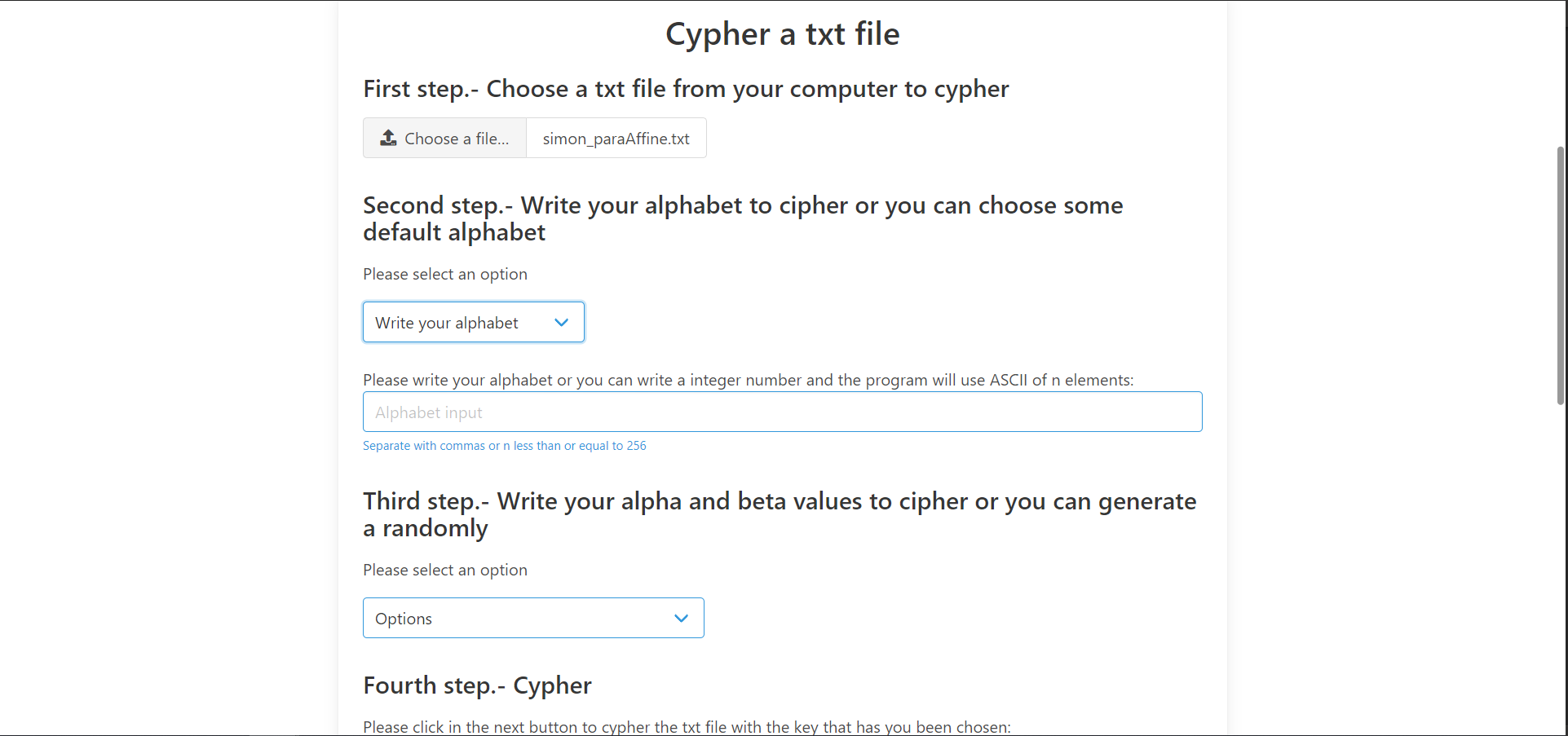


Imagen 13. Opción de escribir un alfabeto propio.

En la imagen 13 vemos que tenemos la opción de escribir una lista con el contenido de nuestro alfabeto separado con comas o podemos escribir un numero entero “n” para poder elegir la cantidad n caracteres del ASCII.

Finalmente tenemos la opción de escribir los valores de alfa y beta o por otro lado generar valores aleatorios, en caso de elegir la opción de escribir nuestros valores nos aparecerá el siguiente componente para poder escribir los valores como se muestra en la imagen 14.

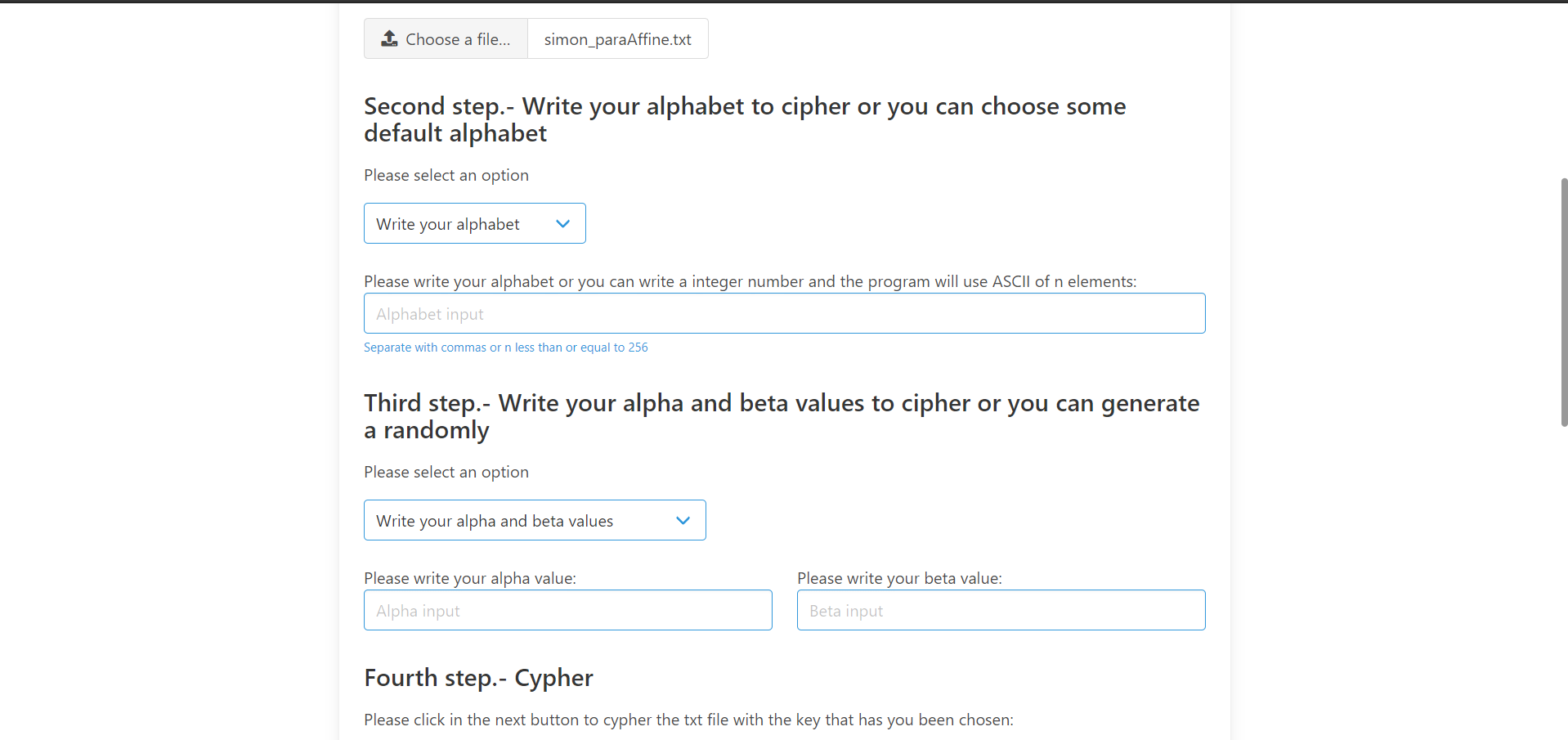


Imagen 14. Opción escribir valores de beta y alfa propios.

Viendo por encima la parte grafica del descifrado vemos que tenemos campos similares a excepción de que en la parte de ingresar los valores alfa y beta es necesario que sean proporcionados por el usuario (ver imagen 15).

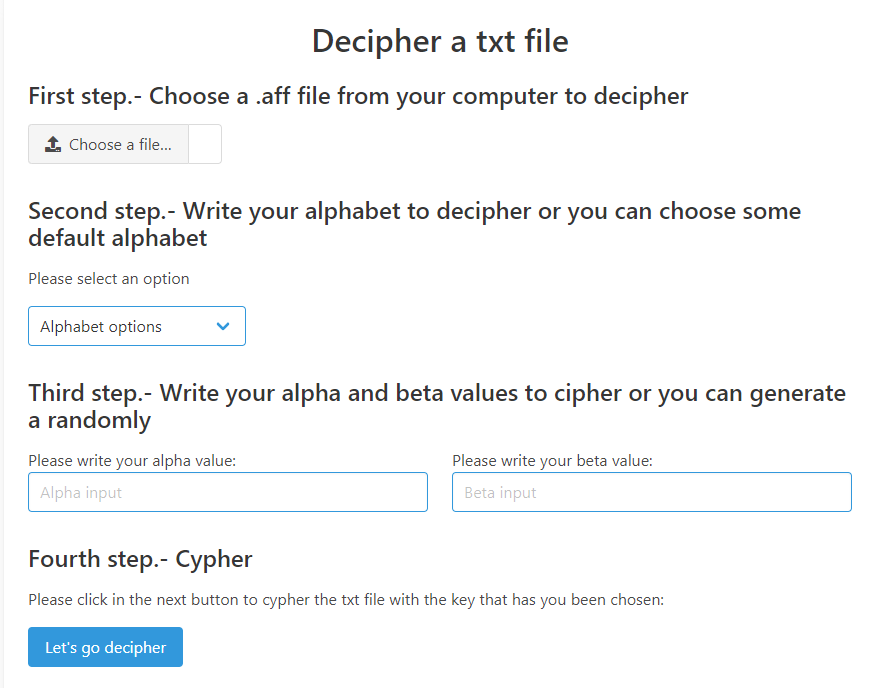


Imagen 15. Parte grafica para el descifrado.

Ahora procederemos con las pruebas correspondientes, empezando con n = 97, recordemos que esa n obtendremos la cantidad n caracteres del ASCII. Este caso obtendríamos hasta la letra a minúscula, generaremos los valores de beta y alfa de manera aleatoria. Ingresando los valores en el programa. (ver imagen 16)

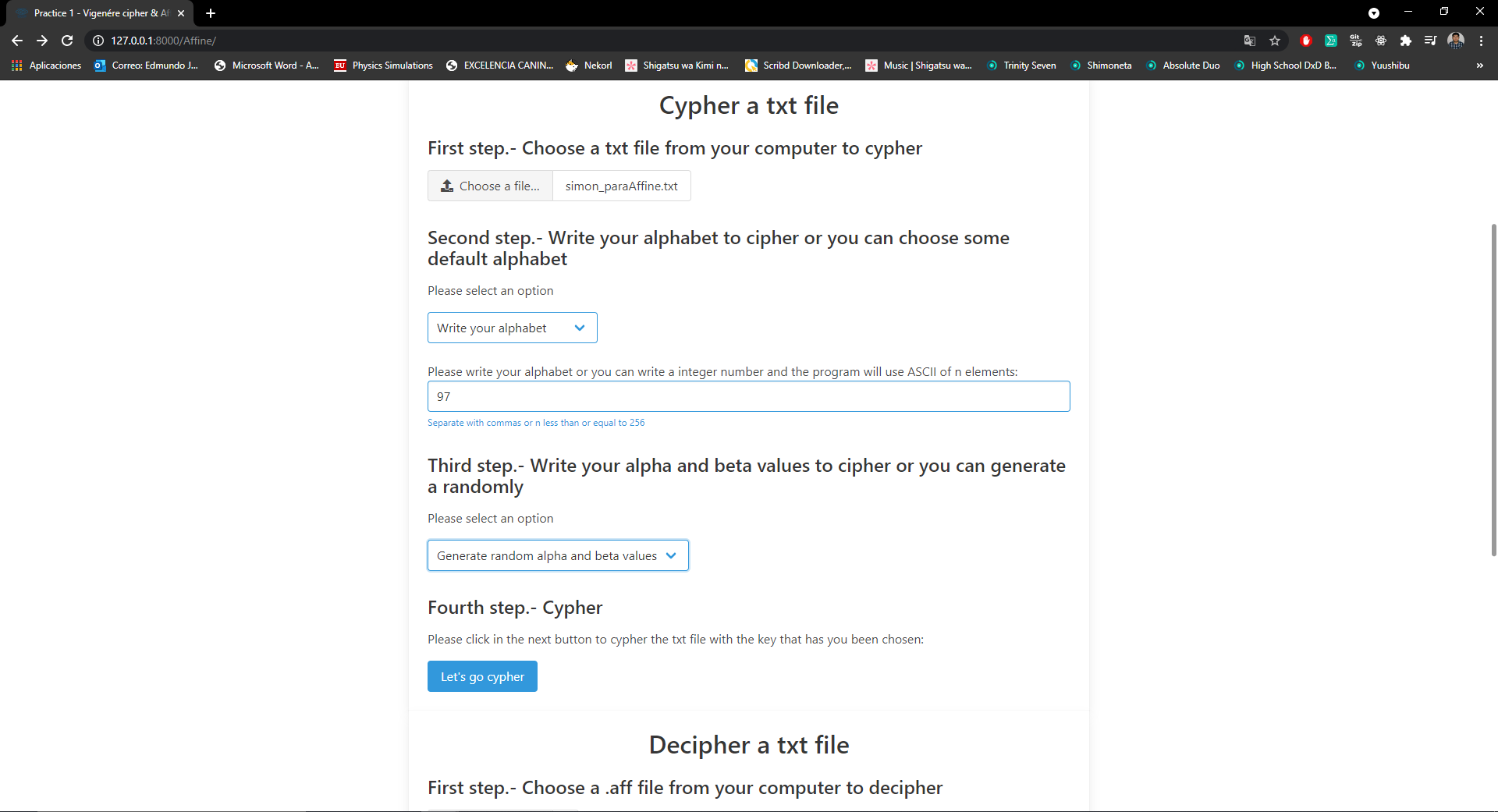


Imagen 16. Ingreso de valores al programa.

Al momento de cifrar nuestro archivo el programa nos arroja el siguiente mensaje. (ver imagen 17)

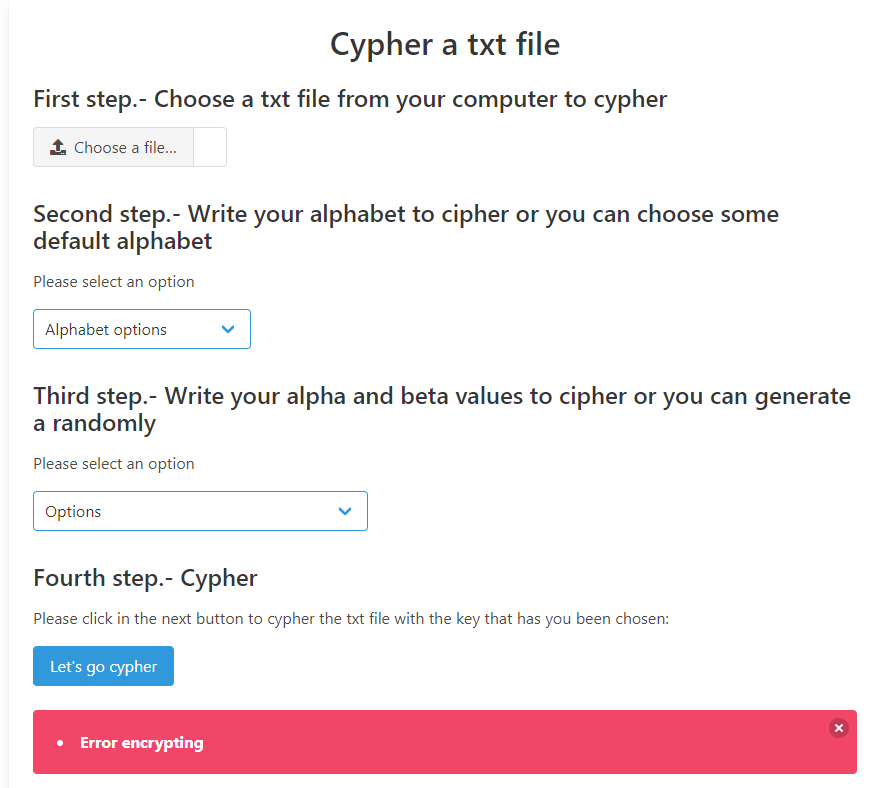


Imagen 17. Mensaje de error.

Como vemos nos manda un error de encriptación y es que era de esperarse ya que el contenido de nuestro archivo contiene letras minúsculas del abecedario en inglés, pero al tomar solo 97 elementos excluimos el abecedario ingles en minúsculas a excepción de la letra a, sin embargo, veamos la prueba con n=128, veremos que ahora si se incluye el abecedario en ingles con las letras minúsculas, pasemos ahora a la imagen 18 en donde introducimos los valores correspondientes en el programa, generando nuevamente valores beta y alfa aleatorios y ejecutamos el cifrado.

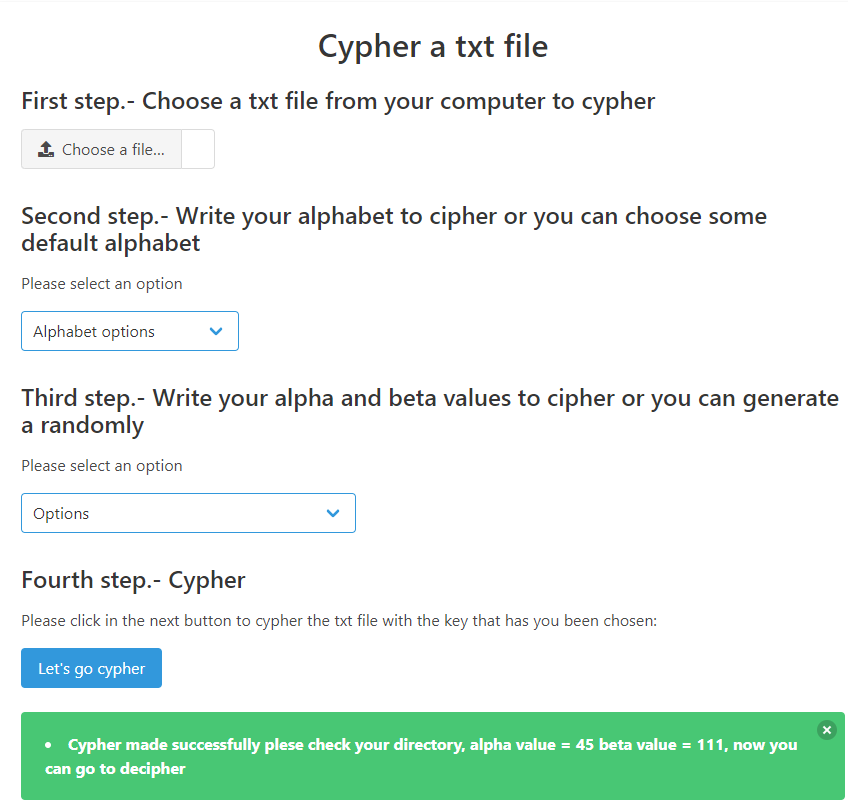


Imagen 18. Archivo cifrado y mensaje con los valores de alfa y beta correspondientes.

En la imagen 19 podemos ver la creación de un archivo con la extensión .aff con el mensaje cifrado y el contenido del mismo.

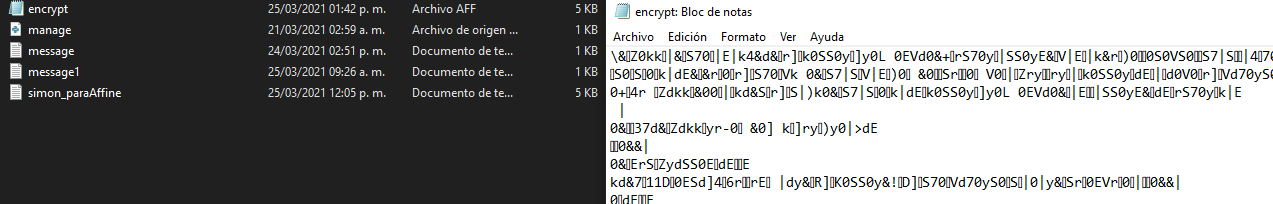


Imagen 19. Creación de archivo .aff y contenido del mismo

Como vemos en la imagen 19 ahora si se pudo realizar el cifrado de nuestro archivo, vemos un mensaje en el que se nos dice que el cifrado se realizo exitosamente y ademas de eso nos devuelve los valores de alfa y beta ya que sin ellos no podríamos realizar el siguiente el cual es el descifrado ver imagen 20.

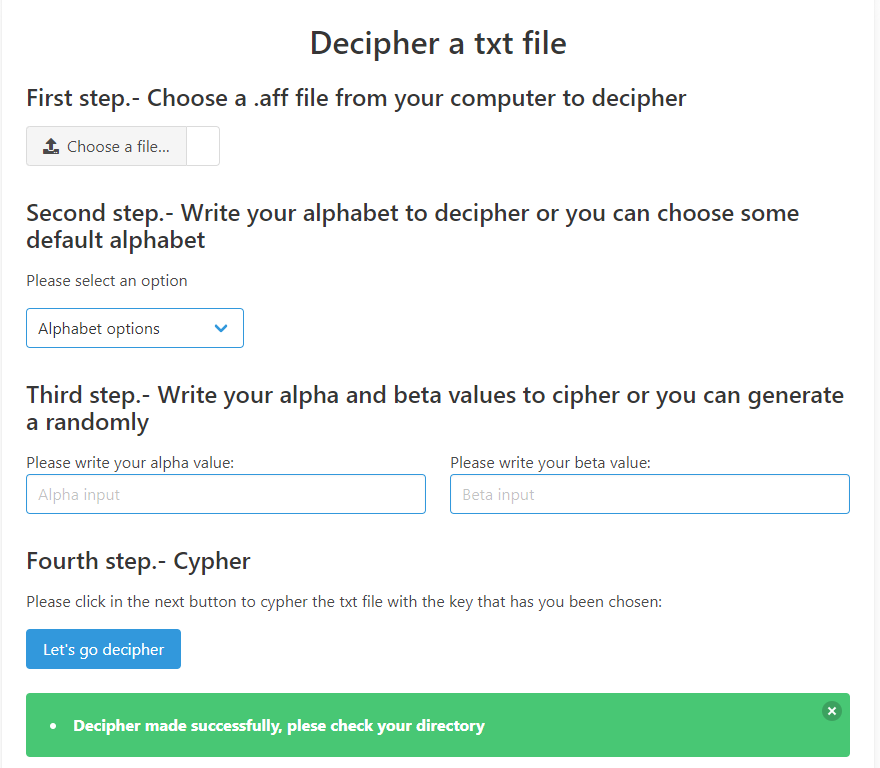


Imagen 20. Descifrado de nuestro archivo con extensión .aff

Como vemos se nos arroja un mensaje diciendo que el descifrado se realizó con éxito y que chequemos nuestro directorio, en la imagen 21 vemos la creación de un archivo con el nombre “decrypt.aff”

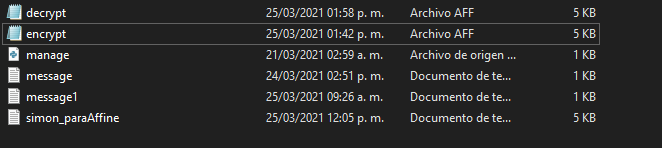


Imagen 21. Creación de archivo “decrypt.aff”

Ahora en la imagen 22 vemos una pequeña comparación entre el archivo general con el archivo descifrado.

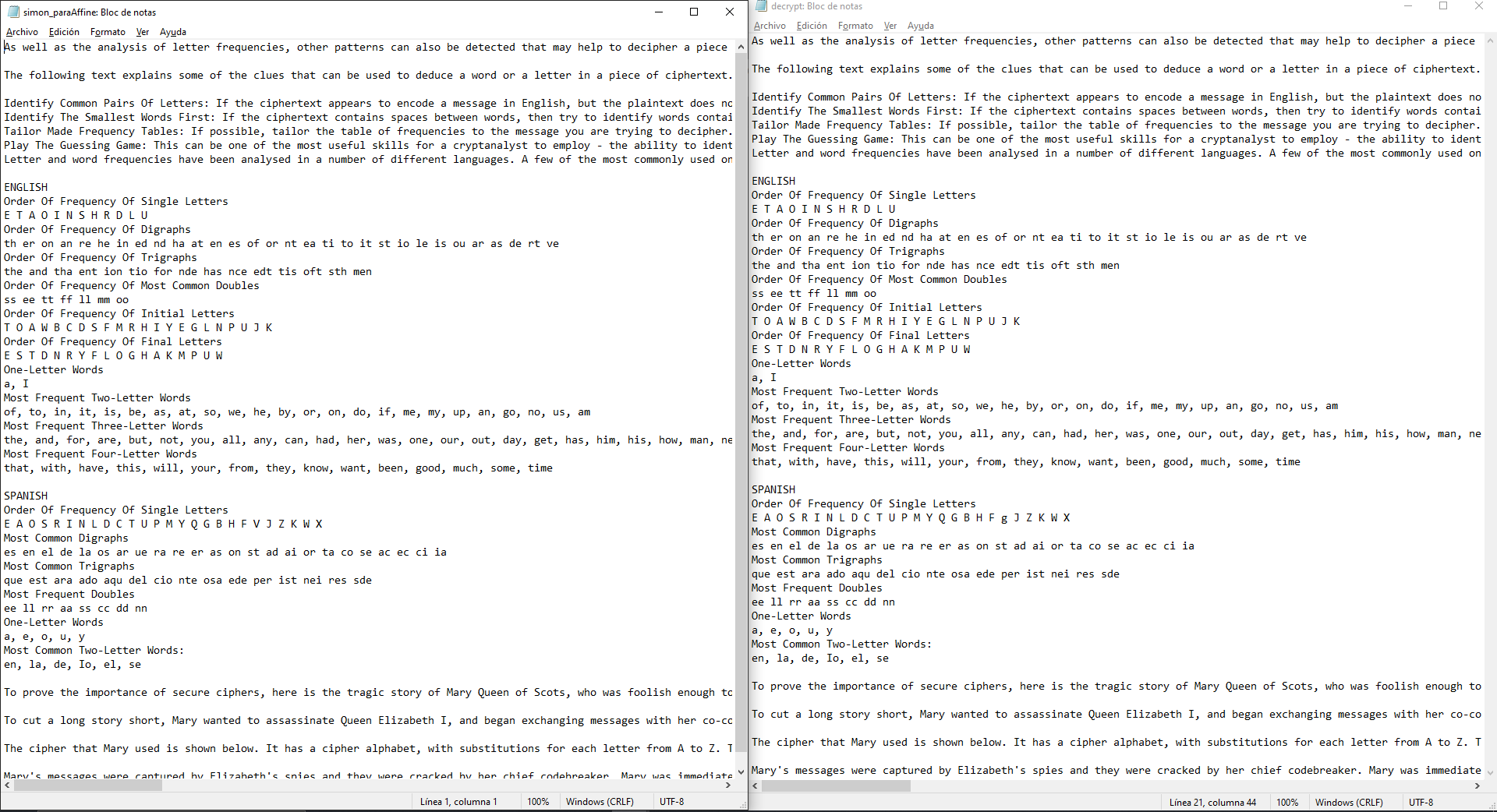


Imagen 22. Comparación de archivos, mensaje original vs imagen descifrado.

Como ultima prueba usaremos los 256 de ASCII, sin embargo, como primera prueba para este apartado usaremos un alfa par, que como sabemos las alfas posibles para el ASCII de 256 no es un alfa valida (ver imagen 23 y 24)

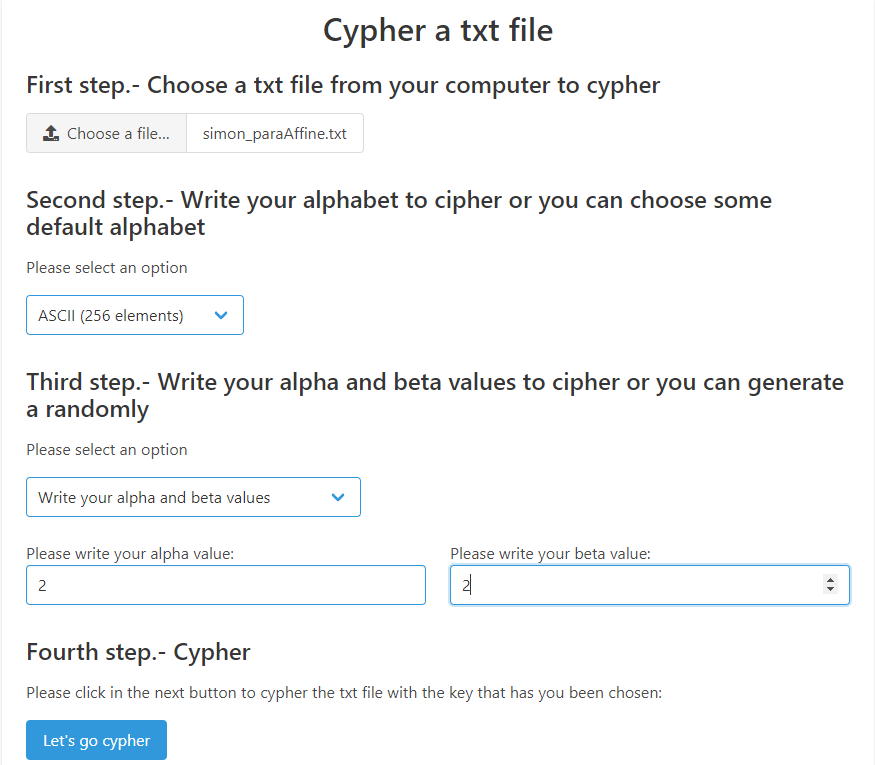


Imagen 23. Valores para usar para la última prueba.

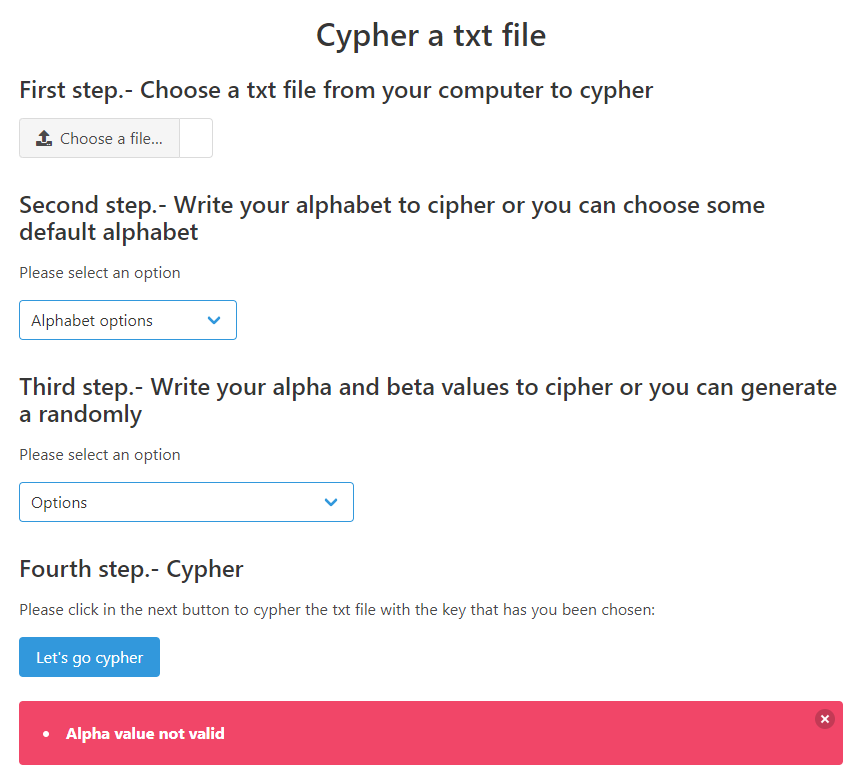


Imagen 24. Mensaje de error de alfa no valido.

Como vemos el programa nos manda un error de alfa no valido, por lo que no se cifra nuestro archivo, ahora en la imagen 25 vemos los valores alfa y beta a usar y en la 26.

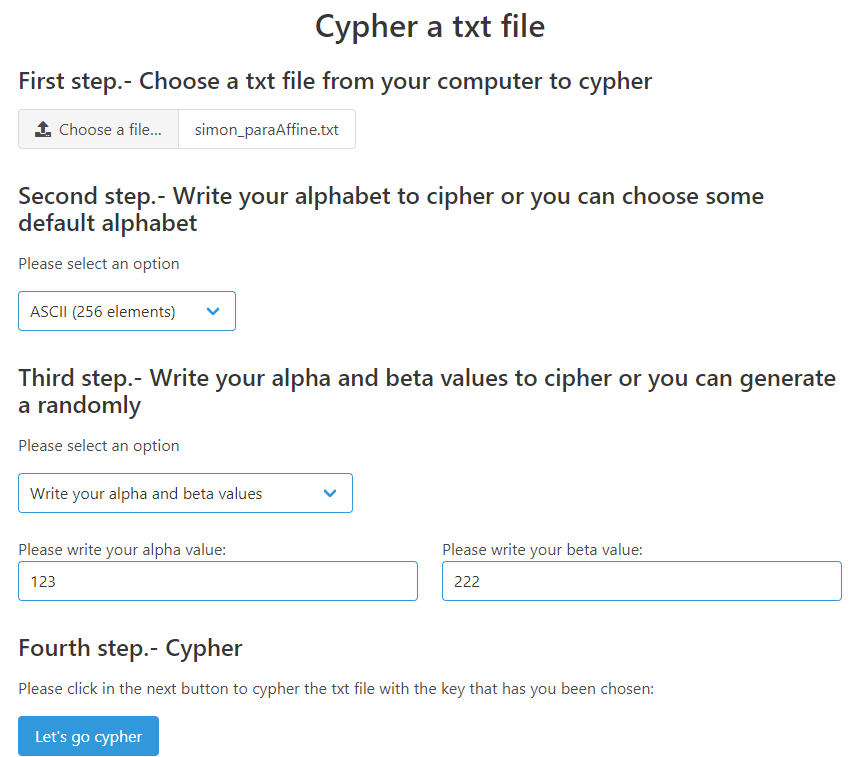


Imagen 25. Datos para usar en el cifrado.

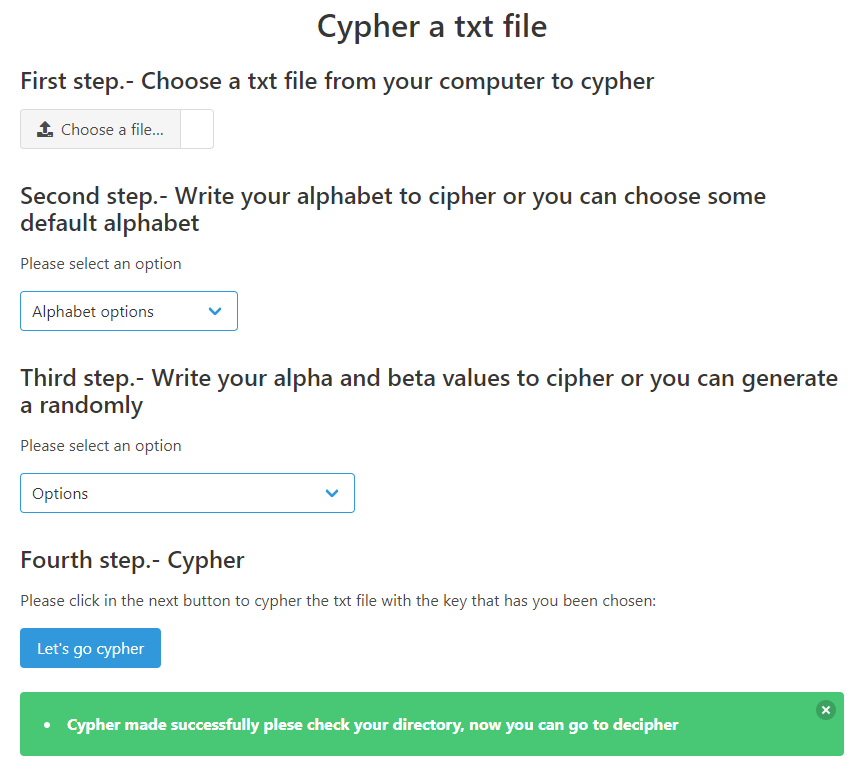


Imagen 26. Cifrado realizado exitosamente.

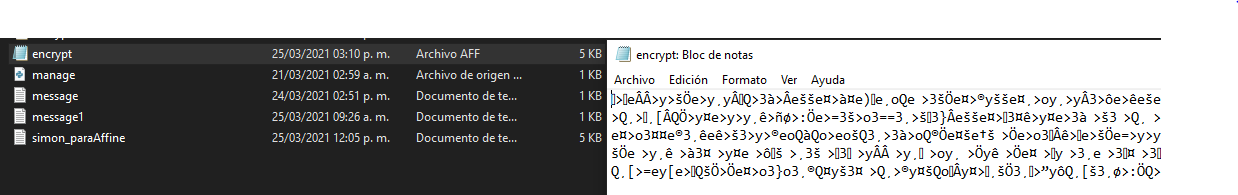


Imagen 27. Archivo cifrado de manera exitosa.

Ahora pasemos a alterar los primeros 10 caracteres de nuestro archivo, no sin antes crear una copia para posteriormente descifrarla.

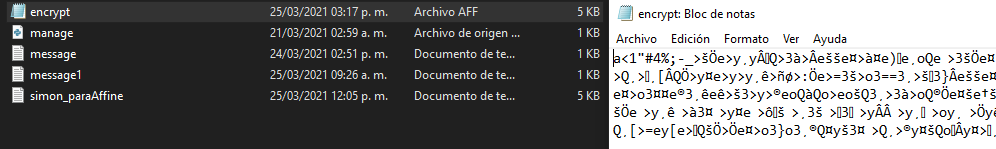


Imagen 28. Nuevo contenido del archivo.

Ahora procedamos a descifrarlo ver imagen 29.

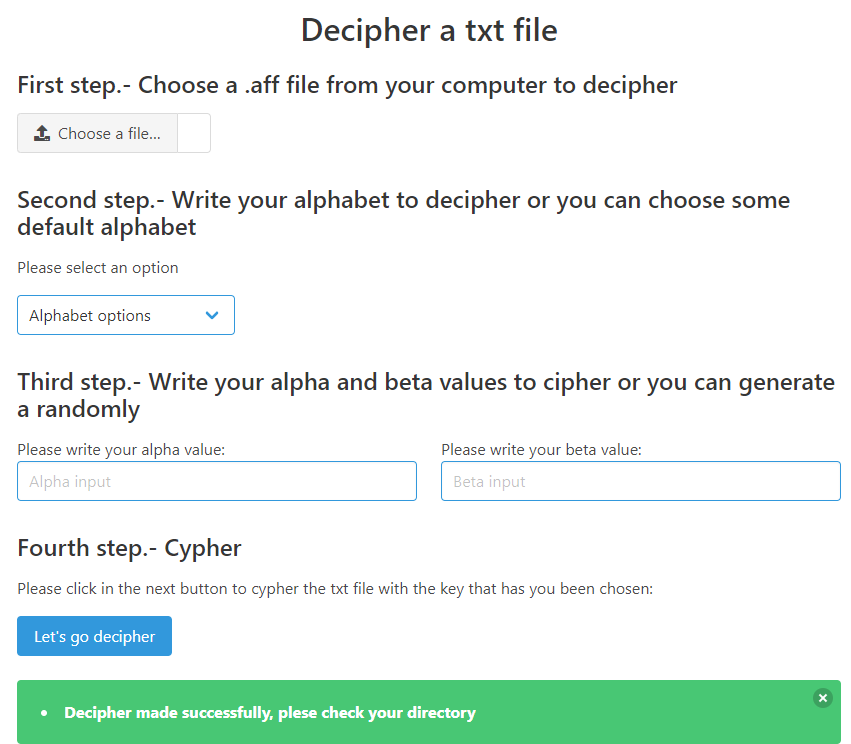


Imagen 29. Descifrado realizado de manera correcta.

Como vemos en la imagen 29 el descifrado se realizo de manera correcta, pero si nos dirigimos a ver el archivo descifrado veremos que los primeros 10 caracteres de nuestro texto fueron modificados provocando perdida de información. (ver imagen 30)

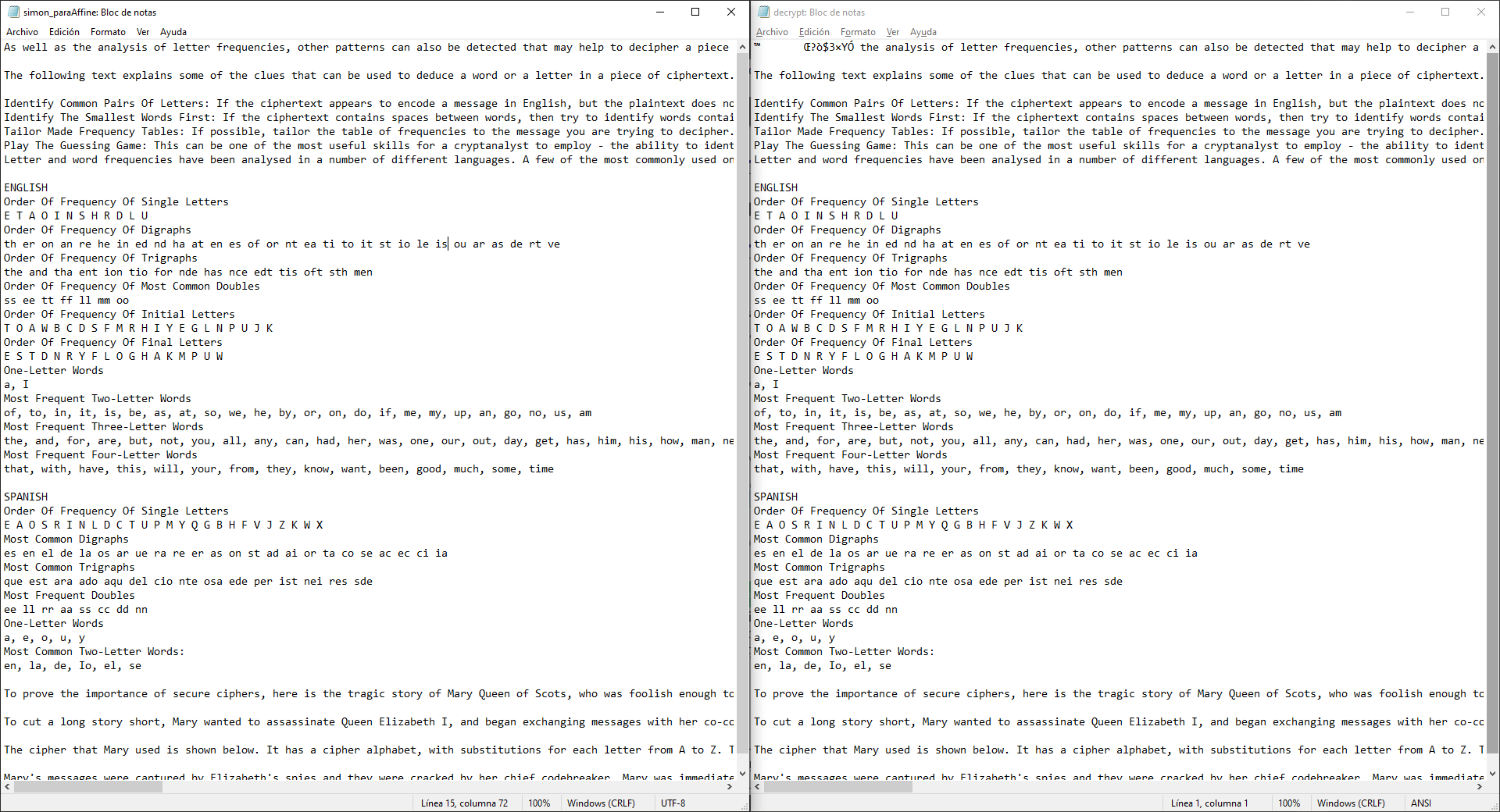


Imagen 30. Comparación del archivo original con el descifrado del archivo modificado en 10 caracteres.

INSTRUMENTO DE EVALUACIÓN

|  |  |  |
| --- | --- | --- |
| Aspectos a evaluar | Puntaje | Tú |
| Documento | | |
| La portada tiene: nombre completo, materia, nombre de profesor, fecha, logotipos, título de práctica y un resumen. (ver ejemplo anexo) | 1 |  |
| En media cuartilla con tus palabras explicar el AE y AEE, qué es y para qué sirve cada uno de ellos. | 1 |  |
| Código correspondiente a las dos funciones AE y AEE y captura de pantalla de las ejecuciones de las pruebas solicitadas. | 2 |  |
| Media cuartilla con tus palabras sobre el algoritmo Affine. | 1 |  |
| Código correspondiente a las dos funciones (cifrado y descifrado) así como el cálculo de la llave de descifrado y captura de pantalla de las ejecuciones de las pruebas solicitadas. | 2 |  |
| Media cuartilla con tus palabras sobre el algoritmo Vigenère. | 1 |  |
| Código correspondiente a las dos funciones (cifrado y descifrado) así como el cálculo de la llave de descifrado y captura de pantalla de las ejecuciones de las pruebas solicitadas. | 2 |  |
| Conclusiones en donde se exprese principalmente las dificultades de la imple mentación de la práctica (si es que las hubo) así como una reflexión sobre la diferencia entre cifradores de sustitución mono alfabética y poli alfabético | 1 |  |
| El código tiene formato (sugiero utilizar http://www.planetb.ca/syntax-highli ght-word) | 1 |  |
| Todas las imágenes en el documento tienen título y se referencian en alguna parte del mismo. (Ej. “en la imagen 1 se muestra ...”) | 1 |  |
| Programa | | |
| El programa cuenta con interfaz gráfica que le permita al usuario elegir la opción deseada: algoritmo y cifrado o descifrado y que reciba los parámetros (en caso de ser necesario) | 2 |  |
| La interfaz permite seleccionar el archivo que se va a cifrar/descifrar | 2 |  |
| Su programa genera el archivo cifrado con el mismo nombre del archivo de entrada más la extensión .vig o .aff | 2 |  |
| Las funciones AE y AEE mandan mensaje “prueba con otro valor” en caso de que alpha no sea coprimo con n | 1 |  |
| Total | 20 | ? |