



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

Práctica 03: Cifrados clásicos y contemporáneos

ALUMNOS

Gabriela López Diego - 318243485
Abraham Jiménez Reyes - 318230577
Javier Alejandro Rivera Zavala - 311288876
Juan Daniel San Martín Macias - 318181637

PROFESORA

Anayanzi Delia Martínez Hernández

AYUDANTES

Cecilia del Carmen Villatoro Ramos
Roberto Adrián Bonilla Ruíz
Ivan Daniel Galindo Perez
Roberto Adrián Bonilla Ruíz

ASIGNATURA

Criptografía y Seguridad

27 de Febrero del 2024

1. Introducción

En esta práctica exploraremos algunos de los cifrados clásicos vistos a lo largo de las clases, del mismo modo, tendremos un primer acercamiento a las herramientas de cifrado y codificación que existen hoy día. Haremos uso de uno de los tipos de cifrado más básico, tal como lo es el cifrado monoalfabético por desplazamiento y luego de ello utilizaremos la herramienta de cifrado y codificado OpenSSL.

La idea tras de esta práctica es analizar y comparar, los distintos recursos de los que nos dotan los sistemas de cifrado y codificado existentes, para así lograr una comunicación de la información más segura. Buscamos también, entender la complejidad con la que cuentan los distintos sistemas de cifrado y codificado que hemos explorado, esto tanto a la hora de usarse como a la hora de implementarse a través de un lenguaje de programación moderno, en este caso, Python.

Por último pero no menos importante, durante esta práctica se incidirá en una noción fundamental de la materia estudiada, la diferencia existente entre cifrar y decodificar, términos que aunque en el habla común suelen ser de uso intercambiable, en realidad se refieren a cosas distintas, aunque notar dichas diferencias requiera de un análisis de las sutilezas.

2. Desarrollo

(3 PUNTOS) CIFRADO DE CÉSAR

- **(0.5 puntos)** Dado el siguiente texto, realicen cifrado de César paso a paso para obtener el texto ilegible con clave de desplazamiento 10, expliquen como han logrado de la misma forma que se muestra en el ejemplo.

Todo a su tiempo

Comenzaremos realizando el cifrado de César de izquierda a derecha de nuestro texto dado. Determinaremos a C como el índice de la letra cifrada, p el índice de la letra no cifrada en el abecedario (sin considerar la letra ñ) y k la clave de desplazamiento, es decir, $k = 3$.

0	1	2	3	4	5	6	7	8	9	10	11	12
A	B	C	D	E	F	G	H	I	J	K	L	M
13	14	15	16	17	18	19	20	21	22	23	24	25
N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Con cada una de las letras que cifraremos, tomaremos su índice inicial (comenzando por 0) y le sumaremos 3 unidades (el valor de k), ya que es el valor del desplazamiento que en esta ocasión realizaremos. Con el resultado obtenido, aplicamos modulo 26 donde 26 el número total de letras en nuestro abecedario, y de no realizarlo podríamos generar errores cuando queramos cifrar las ultimas letras. Por ende, utilizaremos la siguiente formula para el cifrado de cada una de las letras del texto dado.

$$C = (p + k) \text{ modulo } 26$$

Para la letra **T** tendremos

$$C = (19 + 3) \text{ mod } 26$$

$$C = 22 \text{ mod } 26 = 22$$

De igual forma, para la letra **o**

$$C = (14 + 3) \text{ mod } 26$$

$$C = 17 \text{ mod } 26 = 17$$

Para la letra **d**

$$C = (3 + 3) \text{ mod } 26$$

$$C = 6 \text{ mod } 26 = 6$$

Nuevamente queremos cifrar la letra **o** sin embargo, ya la hemos cifrado con anterioridad. Sabemos que para **o**, $C = 17$.

Para la letra **a**

$$C = (0 + 3) \bmod 26$$

$$C = 3 \bmod 26 = 3$$

Para la letra **s**

$$C = (18 + 3) \bmod 26$$

$$C = 21 \bmod 26 = 21$$

Para la letra **u**

$$C = (20 + 3) \bmod 26$$

$$C = 23 \bmod 26 = 23$$

Nuevamente queremos cifrar la letra **t** sin embargo, ya la hemos cifrado con anterioridad. Sabemos que para **t**, $C = 22$.

Para la letra **i**

$$C = (8 + 3) \bmod 26$$

$$C = 11 \bmod 26 = 11$$

Para la letra **e**

$$C = (4 + 3) \bmod 26$$

$$C = 7 \bmod 26 = 7$$

Para la letra **m**

$$C = (12 + 3) \bmod 26$$

$$C = 15 \bmod 26 = 15$$

Para la letra **p**

$$C = (15 + 3) \bmod 26$$

$$C = 18 \bmod 26 = 18$$

Nuevamente queremos cifrar la letra **o** sin embargo, ya la hemos cifrado con anterioridad. Sabemos que para **o**, $C = 17$. Ahora, agrupando todos los índices de cada una de las letras ya cifradas obtenemos

22, 17, 6, 17, 3 21, 23 22, 11, 7, 15, 18, 17

Y por lo tanto, el texto dado cifrado empleando el cifrado de César queda de la siguiente manera

WRGR D VX WLHPSR

- **(1 punto)** Empleando el alfabeto de 26 letras de la A a la Z, creen un programa en python que reciba texto plano y el número de desplazamientos el cual muestre el nuevo alfabeto y su texto cifrado. Agreguen capturas de pantalla de ejecución de su programa, la salida deberá verse de la siguiente manera, así mismo deberán adjuntar su código en la entrega.

```
1 ENTRADA
2 Texto plano:  "De la abundancia del corazon habla la boca"
3 Numero de desplazamientos: 6
4
5 SALIDA:
6 "Jk rg ghatjgtiog jkr iuxgfut nghrg rg huig"
7
8 ALFABETO:
9 A,B,C,D,E,F, ... ,Y,Z  ## Alfabeto original
10 G,H,I,J,K,L, ... ,E,F  ## Alfabeto modificado segun el numero de desplazamiento
11
```

Es importante mostrar como se vería el alfabeto modificado (con las primeras 6 letras basta)

Primero hicimos el código en un collab.

```

1 def cesar(texto, desplazamientos):
2     alfabeto = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
3     texto_cifrado = ''
4
5     for letra in texto:
6         if letra.upper() in alfabeto:
7             indice = (alfabeto.index(letra.upper()) + desplazamientos) % 26
8             if letra.isupper():
9                 texto_cifrado += alfabeto[indice]
10            else:
11                texto_cifrado += alfabeto[indice].lower()
12        else:
13            texto_cifrado += letra
14
15    return texto_cifrado
16
17
18 nombre_archivo = input("Ingrese el nombre del archivo de texto (incluyendo la extension .
19                        txt): ")
20
21 try:
22     with open(nombre_archivo, 'r') as archivo:
23         texto_original = archivo.read()
24 except FileNotFoundError:
25     print("El archivo especificado no fue encontrado.")
26     exit()
27
28 desplazamientos = int(input("Numero de desplazamientos: "))
29
30 print("\nALFABETO ORIGINAL:")
31 print("A,B,C,D,E,F,...Y,Z")
32
33
34 alfabeto_desplazado = ''.join([chr(((ord(letra) - 65 + desplazamientos) % 26) + 65) for
35                                letra in 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'])
36 print("\nALFABETO CON DESPLAZAMIENTO DE", desplazamientos, "POSICIONES:")
37 print(alfabeto_desplazado)
38
39 texto_cifrado = cesar(texto_original, desplazamientos)
40
41
42 print("\nTexto cifrado:")
43 print(texto_cifrado)
44

```

El archivo que utilizamos como entrada se llama mes.txt en el tenemos los meses del año, el ejemplo al que debemos cifrar y el texto del primer ejercicio

```
# Mostrar el texto cifrado
print("\nTexto cifrado:")
print(texto_cifrado)

Ingrese el nombre del archivo de texto (incluyendo la extensión .txt): mes.txt
Número de desplazamientos: 6

ALFABETO ORIGINAL:
A,B,C,D,E,F,...Y,Z

ALFABETO CON DESPLAZAMIENTO DE 6 POSICIONES:
GHIJKLMNOPQRSTUVWXYZABCDEF

Texto cifrado:
Ktkxu
Lkhxxku
Sgxfu
Ghxor
Sgeu
Patou
Parou
Gmuyzu
Ykvzokshxk
Tubokshxk
Joiockshxk

zuju g ya zoksuvu
Jk rg ghatjgtiog jkr iuxgfót nghrg rg huig
```

- (0.5 puntos) Validen el texto obtenido en el ejercicio 1 utilizando tu programa ¿obtuvieron el mismo resultado? Adjuten una captura de su ejecución.

```
Ingrese el nombre del arch
Número de desplazamientos:

ALFABETO ORIGINAL:
A,B,C,D,E,F,...Y,Z

ALFABETO CON DESPLAZAMIENTO
DEFGHIJKLMNOPQRSTUVWXYZABC

Texto cifrado:
Wrgr d vx wlhpsr
```

Si llegamos al mismo resultado

- **(1 punto)** Dado el siguiente texto codificado por el cifrado de César (se desconoce de cuánto es el desplazamiento). Muestren el texto descifrado e indica cuál es el número de desplazamiento que se ha utilizado para cifrar. Respondan ¿cómo obtuvieron el texto? ¿qué herramientas ó recursos utilizaron? ¿cómo obtuvieron la clave de desplazamiento? Argumenten sus procedimientos detalladamente.

```
1 Ls htvy lz whjplual, lz ivukhkvzv. Ls htvy uv lz lucpkpvzv up wylzbtphv up vynbssvzv. Uv
  zl jvtwvyah jvu ybklgh, uv lz lnpvzah, uv zl luvqh mhjpstlual, uv nbhykh ylujvy. Ls
  htvy uv zl klslpah lu sh thskhk, zpuv xbl zl ylnvjpqh jvu sh clykhh. Avkv sv kpzjbswh
  , avkv sv jyll, avkv sv lzwlyh, avkv sv zvwvyah.
```

Muestren el texto descifrado e indica cuál es el número de desplazamiento que se ha utilizado para cifrar.

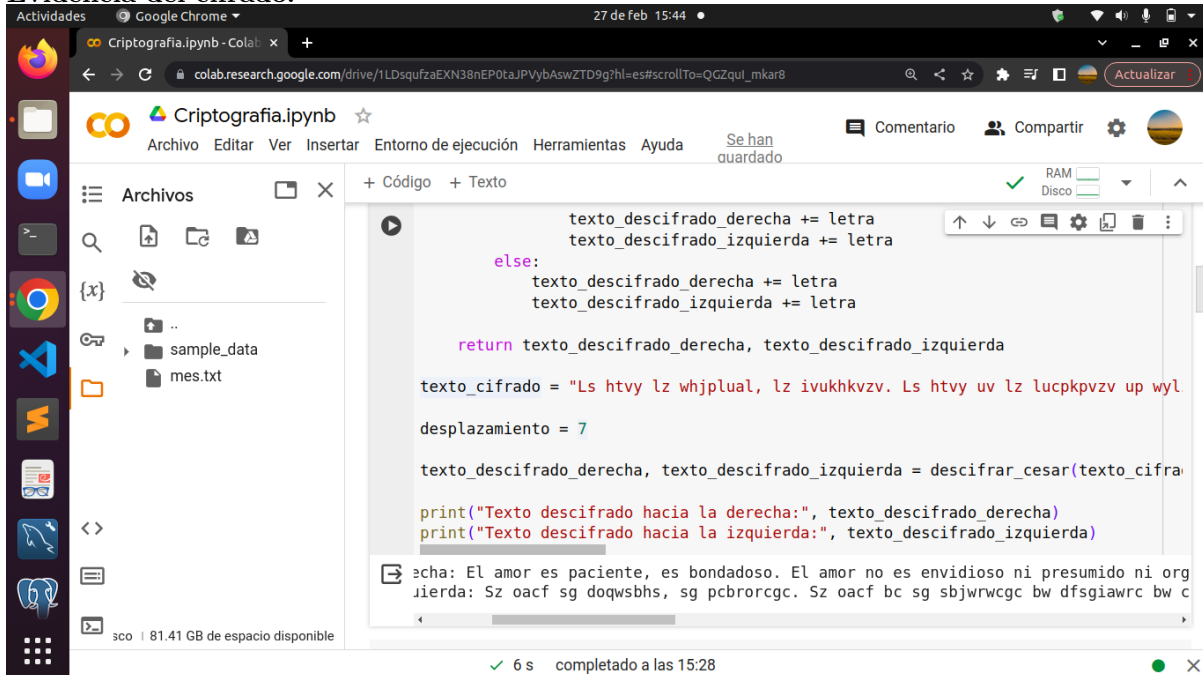
El texto es: El amor es paciente, es bondadoso. El amor no es envidioso ni presumido ni orgulloso. No se comporta con rudeza, no es egoista, no se enoja facilmente, no guarda rencor. El amor no se deleita en la maldad, sino que se regocija con la verdad. Todo lo disculpa, todo lo cree, todo lo espera, todo lo soporta.

El desplazamiento fue de 7.

Respondan ¿cómo obtuvieron el texto? ¿qué herramientas ó recursos utilizaron? Utilizamos el código que se utilizo en el ejercicio posterior a este llamado *descifrar_cesar*

¿Cómo obtuvieron la clave de desplazamiento? Colocamos al texto a decifrar y para el numero de desplazamientos con fuerza bruta, esto quiere decir que intentamos los 25 desplazamientos que existen y afortunadamente con el numero 7 nos dio un texto legible.

Evidencia del cifrado.



```

texto_descifrado_derecha += letra
texto_descifrado_izquierda += letra

else:
    texto_descifrado_derecha += letra
    texto_descifrado_izquierda += letra

return texto_descifrado_derecha, texto_descifrado_izquierda

texto_cifrado = "Ls htvy lz whjplual, lz ivukhkvzv. Ls htvy uv lz lucpkpvzv up wylzbtphv up vynbssvzv. Uv
zl jvtwvyah jvu ybklgh, uv lz lnpvzah, uv zl luvqh mhjpstlual, uv nbhykh ylujvy. Ls
htvy uv zl klslpah lu sh thskhk, zpuv xbl zl ylnvjpqh jvu sh clykhh. Avkv sv kpzjbswh
, avkv sv jyll, avkv sv lzwlyh, avkv sv zvwvyah."

desplazamiento = 7

texto_descifrado_derecha, texto_descifrado_izquierda = descifrar_cesar(texto_cifrado, desplazamiento)

print("Texto descifrado hacia la derecha:", texto_descifrado_derecha)
print("Texto descifrado hacia la izquierda:", texto_descifrado_izquierda)

```

acha: El amor es paciente, es bondadoso. El amor no es envidioso ni presumido ni orgulloso. No se comporta con rudeza, no es egoista, no se enoja facilmente, no guarda rencor. El amor no se deleita en la maldad, sino que se regocija con la verdad. Todo lo disculpa, todo lo cree, todo lo espera, todo lo soporta.

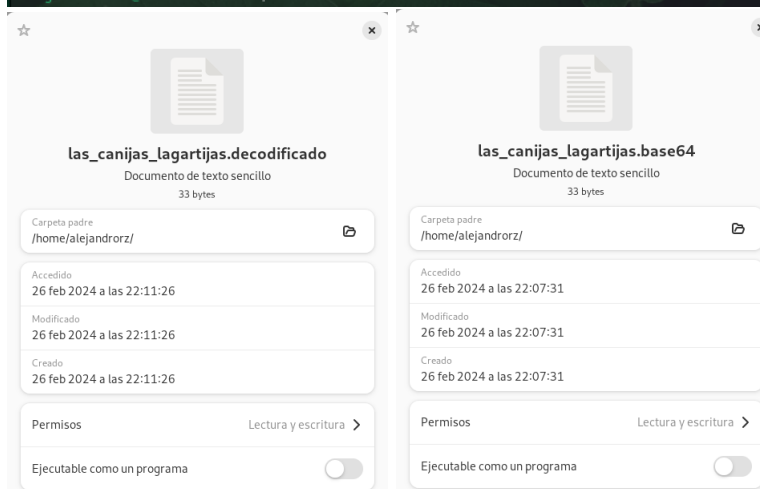
6 s completado a las 15:28

(7 PUNTOS) CIFRADO SIMÉTRICO

■ (3 puntos)

1. Creen un archivo txt con el cual contenga escrito el nombre de su equipo como se muestra en el ejemplo a continuación:
(Si son equipo 1 y se llaman “mazapan”)
Mazapan.txt → (texto dentro del archivo) : mazapan
Únicamente agregar al archivo txt, el nombre del equipo
2. Codificar su archivo a base64: `*openssl enc -base64 -in Equipo1.txt -out Equipo1.base64*`
Aclaración: Codificar no significa cifrar, solo es un paso previo
3. Cifren el archivo base64 con el algoritmo simétrico aes256
`*openssl enc -aes-256-cbc -in equipo1.base64 -out equipo1.cifrado*`
¿Cómo afectaría cambiar `*cbc*` por `*ecb*` en los parámetros? ¿Qué significan dichas letras?
cbc significa “cipher block chaining”, al emplear dicho parámetro se garantiza que entonces el cifrado sea “acumulativo”, es decir, cada bloque de texto plano toma en consideración al bloque inmediato anterior, haciendo una combinación de ambos mediante el operador XOR, para así producir el cifrado del bloque en turno. Por otro lado, ecb significa “electronic codebook”, a diferencia de cbc, en esta modalidad el cifrado se hace bloque por bloque de manera independiente, de tal suerte que bloques de texto iguales terminarán siendo cifrados exactamente de la misma forma. Obviamente, al cambiar el parámetro cbc por ecb, hacemos que sea más sencillo reconocer patrones durante el análisis, lo cual podría suponer una vulnerabilidad frente a un posible atacante.
4. Procedan a decodificar el mensaje
`*openssl enc -aes-256-cbc -d -in equipo1.cifrado -out equipo1.decodificado*`
Expliquen las diferencias que hay entre los archivos .base64 y .decodificado ¿Qué se puede argumentar?
No se aprecia ninguna diferencia notable entre ambos archivos salvo por su extensión, incluso cuando verificamos las propiedades de dichos archivos, estas aparecen iguales:

```
alejandrorz@immerwahr:~$ ls
Descargas  las_canijas_lagartijas.base64  llave.txt  Videos
Documentos las_canijas_lagartijas.cifrado  Música
Escritorio las_canijas_lagartijas.decodificado  Plantillas
Imágenes   las_canijas_lagartijas.txt        Público
alejandrorz@immerwahr:~$ cat las_canijas_lagartijas.base64
bGFzIGNhbm1qYXNjbGFnYXJ0aWphcwo=
alejandrorz@immerwahr:~$ cat las_canijas_lagartijas.decodificado
bGFzIGNhbm1qYXNjbGFnYXJ0aWphcwo=
alejandrorz@immerwahr:~$
```



Podemos apreciar que el contenido del archivo `las_canijas_lagartijas.decodificado` es el resultado de descifrar a través de `openssl` el texto cifrado contenido en el archivo `las_canijas_lagartijas.cifrado`, pues aunque el texto nos dice que decodificamos, en realidad se emplea un comando para descifrar.

Por lo anterior es lógico que las `canijas_lagartijas`.`decodificado` luzca como en el archivo `codificado` en base 64 que generamos en un principio.

5. 5. Descifren el archivo:

`*openssl enc -base64 -d -in equipo1.decodificado -out equipo1.descifrado*`

6. Adjunten en un zip `Equipo<N>.zip` los 5 archivos generados adicional a un sexto archivo llamado `"llave.txt"` en el cual contendrá la llave elegida usada para cifrar sus archivos.

7. Respondan ¿qué utilidad ven a este tipo de cifrado? ¿en qué situaciones sería un buen recurso a utilizar?

Al haberse hecho uso de un cifrado simétrico, consideramos que sería más útil cuando se trata de intercambiar información con personas que ya cuentan con la clave para descifrar el texto. No es tan confiable si tenemos que enviar (como en este caso) la clave a través de un canal potencialmente inseguro. Podría servir también en escenarios donde sólo nosotros queremos acceder a la información cifrada, un ejemplo sería un pequeño respaldo de nuestras contraseñas. Podemos guardar nuestras contraseñas en un archivo cifrado que administraremos a través de una clave maestra, siempre que no compartamos la clave con nadie más, nuestras contraseñas permanecerán relativamente seguras.

- (3 puntos) La siguiente imagen muestra sinónimos de palabras utilizando el cifrado de César, con la pequeña modificación que el desplazamiento puede ser tanto a la izquierda como a la derecha. Descifren las palabras y únanlas con su sinónimo del otro lado, notarán una intersección en alguna de las 12 imágenes, muestren las palabras decodificadas y en cual imagen se denota la intersección, detallen el proceso que emplearon para llegar a ella. Finalmente respondan ¿que número de clave obtuvieron para obtener las palabras?



El proceso que se utilizó para descifrar las palabras fue con el desarrollo de un algoritmo que puede descifrar una cadena cifrada con un desplazamiento hacia la izquierda y hacia la derecha.

```
1 def descifrar_cesar(texto_cifrado, desplazamiento):
2     alfabeto = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
3     texto_descifrado_derecha = ''
4     texto_descifrado_izquierda = ''
5
6     for letra in texto_cifrado:
7         if letra.isalpha():
8             mayuscula = letra.isupper()
9             letra = letra.upper()
10            if letra in alfabeto:
11                indice_derecha = (alfabeto.index(letra) - desplazamiento) % 26
12                indice_izquierda = (alfabeto.index(letra) + desplazamiento) % 26
```



```

13
14         if not mayuscula:
15             texto_descifrado_derecha += alfabeto[indice_derecha].lower()
16             texto_descifrado_izquierda += alfabeto[indice_izquierda].lower()
17         else:
18             texto_descifrado_derecha += alfabeto[indice_derecha]
19             texto_descifrado_izquierda += alfabeto[indice_izquierda]
20     else:
21         texto_descifrado_derecha += letra
22         texto_descifrado_izquierda += letra
23     else:
24         texto_descifrado_derecha += letra
25         texto_descifrado_izquierda += letra
26
27     return texto_descifrado_derecha, texto_descifrado_izquierda
28
29 texto_cifrado = "ivultzuf"
30 desplazamiento = 9
31
32 texto_descifrado_derecha, texto_descifrado_izquierda = descifrar_cesar(texto_cifrado,
33                                     desplazamiento)
34
35 print("Texto descifrado hacia la derecha:", texto_descifrado_derecha)
36 print("Texto descifrado hacia la izquierda:", texto_descifrado_izquierda)

```

En la función *descifrar_cesar* que hemos desarrollado, se recibe como entrada el texto cifrado y el número de desplazamientos. Luego, se recorre cada letra del texto cifrado y se calcula la nueva letra correspondiente al realizar el desplazamiento hacia la izquierda y hacia la derecha en el alfabeto. Finalmente, se construye el texto descifrado con las letras correspondientes.

Ya con la implementación de éste algoritmo, se procedió a descifrar las palabras con prueba y error, es decir, ir asignando un valor distinto a la variable *desplazamiento* (línea 30), hasta encontrar una palabra coherente ya sea en el descifrado hacia la izquierda o hacia la derecha.

Ejemplo de ejecución:

```

1 Texto descifrado hacia la derecha: zmlckqlw
2 Texto descifrado hacia la izquierda: reducido
3

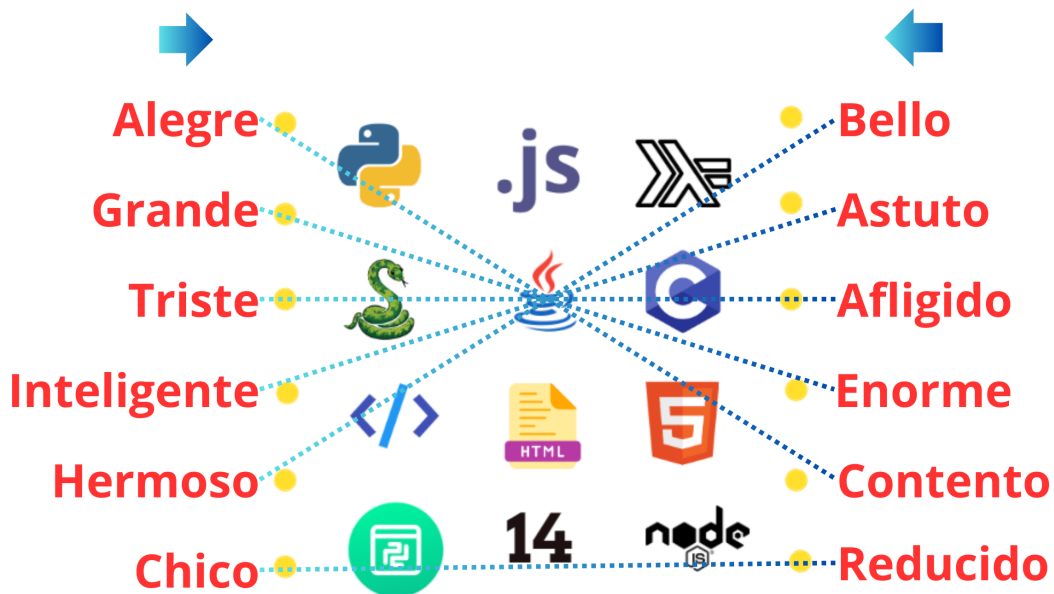
```

Al encontrar que el número de clave de desplazamiento es 9, el valor de la variable queda fijo y sólo se cambió el valor de la variable *texto_cifrado* con los textos que se muestran en la imagen.

Con éste procedimiento obtuvimos éstos resultados:



Donde La flecha de color azul que se encuentra encima indica la dirección del desplazamiento en el cuál se descifró. Por último, se relacionaron las palabras descifradas con sus respectivos sinónimos.



Teniendo que 5 de las 6 líneas se sobreponen sobre el icono del lenguaje de programación **Java**.

- **(1 punto)** Realiza el proceso de cifrado con openssl para una imagen y documenten paso a paso el proceso ¿Funciona igual? ¿Qué se requeriría para obtener la imagen inicialmente de vuelta? ¿Qué pasa si abro el archivo “.descifrado” de una imagen?

Nuestra imagen a cifrar es la siguiente



Primero, crearemos una llave aleatoria

```
1 openssl rand -base64 32 > llave.txt
```

Lo anterior nos generará una llave aleatoria de 32 bytes que se guardará en un archivo .txt llamado *llave*. Trabajaremos sobre 32 bytes ya que usaremos AES-256. Luego, usaremos el siguiente comando para poder encriptar la imagen de la ranita con nuestra llave anterior

```
1 openssl enc -aes-256-cbc -in ranita.jpeg -out ranitaCifrada.cifrado -pass file:llave.txt -pbkdf2
```

Observemos que invocamos openssl junto al algoritmo de cifrado que utilizaremos, en este caso, AES-256. Luego, nos encargamos de asignar el archivo de entrada, la imagen **ranita.jpeg**. Cuando la imagen se encuentre ya cifrada se generará un nuevo archivo que tendrá por nombre **ranitaCifrada.cifrado**. Finalmente, la cifraremos usando la llave que habíamos generado con anterioridad. Usamos **-pbkdf2** por un warning generado en terminal **Derivación de clave obsoleta** por lo cual, openssl nos recomienda usar dicho comando para brindarnos una llave mucho más segura y asegurar nuestro cifrado.

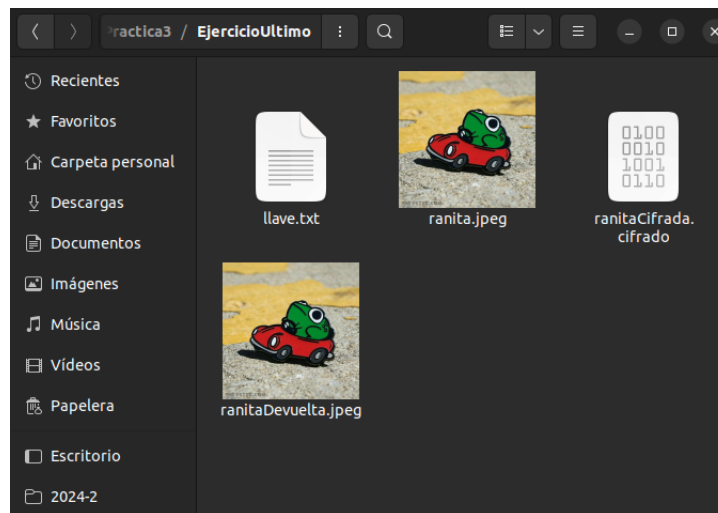
Vemos que el proceso de cifrado de una imagen funciona exactamente igual que el de un archivo. Usamos los mismos comandos y el uso de una llave.

Por otro lado, si queremos la imagen de vuelta, es decir, descifrar la imagen solamente necesitamos realizar lo siguiente en terminal

```
1 openssl enc -d -aes-256-cbc -in ranitaCifrada.cifrado -out ranitaDevuelta.jpeg -pass file:llave.txt -pbkdf2
```

Con lo anterior, llamamos a openssl indicando que queremos descifrar con **-d** y que será usando el algoritmo AES-256. Luego, pasamos como parámetro la imagen cifrada que queremos de vuelta es decir, **-in ranitaCifrada.jpeg** y el nombre del archivo junto con la extensión que queremos se guarde la imagen descifrada **-out ranitaDevuelta.jpeg**. De igual forma, utilizaremos la misma llave que ocupamos para cifrar pero ahora la usaremos para descifrar la imagen. También ocupamos **-pbkdf2** para evitar el warning generado por openssl.

Cuando queramos abrir el archivo generado al descifrar la imagen, tendremos sencillamente de vuelta exactamente la misma imagen que teníamos antes de cifrar.



3. Conclusión

En conclusión, la práctica exploró cifrados clásicos y contemporáneos, como el cifrado César y el cifrado simétrico. Demostró el proceso paso a paso del cifrado de cifrado César y proporcionó un programa Python para el cifrado de texto utilizando un alfabeto de 26 letras. De esto nos llamó la atención lo mucho que ha avanzado el cifrado en éstos cientos de años, si bien la idea en general es la misma (proteger información), los cifrados contemporáneos tienen un alcance mucho mayor, ya que cifran algo más que cadenas de texto. Por otro lado, se realizó un ejercicio de descifrado utilizando el cifrado César, mostrando el uso de herramientas como OpenSSL para el cifrado y descifrado de archivos utilizando el algoritmo simétrico AES256. Se destacó la utilidad práctica de este tipo de cifrado, destacando escenarios en los que puede mejorar las medidas de seguridad.

4. Referencias

- Tobias, Eric. (8 de septiembre de 2021). ECB vs. CBC – Pros and Cons of These Block Cipher Modes. Ubiq. Recuperado el 26 de febrero de 2024. <https://www.ubiqsecurity.com/ecb-vs-cbc-block-cipher-mode-differences/>
- Wikimedia foundation (s.f.). Block cipher mode of operation. Wikipedia. Recuperado el 26 de febrero de 2024. https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation
- Stackoverflow. Recuperado el 26 de Febrero de 2024 <https://stackoverflow.com/questions/26791977/python-caesar-cipher>