

Encriptación y Desencriptación de contraseñas e información

¿De qué manera se puede programar un protocolo de seguridad para una aplicación de notas, usando estructuras de datos, el cifrado de Elgamal y OpenCV?

Asignatura: Informática Aplicada

Código: knn768

N° de Palabras: 3996

Índice

1. Introducción.....	3
2. Planteamiento del problema.....	3
3. Justificación.....	4
4. Objetivo.....	4
5. Marco teórico.....	4
6. Metodología.....	6
7. Discusión y Análisis.....	6
8. Conclusiones.....	17
9. Bibliografía.....	18
10. Anexos.....	20

Introducción

La encriptación ha sido un método usado desde tiempos antiguos para esconder información que haya sido interceptada por un tercero y para lograrlo se utilizan diferentes métodos, pero en este siglo XXI se usan los famosos algoritmos de encriptación y protocolos de seguridad y en esta monografía se creará uno de ellos, usando estructura de datos, el cifrado Elgamal y OpenCV.

Pero ¿Por qué hacer un protocolo de seguridad?, Siempre me ha intrigado como una red social, plataforma, etc., cada día se vuelve una parte más de nosotros y para interactuar con ella, requerimos una contraseña, que es almacenada en una base de datos, pero el inconveniente ocurre cuando esa base de datos es copiada, hackeada o vista por el propio creador, etc., y es en esos momentos cuando nuestra contraseña queda expuesta a la vista de alguien que puede usarla para acceder a nuestra cuenta.

Para solucionar este problema, un algoritmo de cifrado de contraseña en la base de datos es lo idóneo, pero a veces esto no es suficiente y se necesitan implementar más protocolos de verificación, por lo que el uso de estructura de datos, el cifrado de Elgamal y OpenCV son idóneos para este problema.

Planteamiento del problema

Un problema clave es el time complexity y el space memory de el cifrado de Elgamal y la demás ejecución de todo el código, por lo que el uso de estructuras de datos como Linkedlist, permitiría bajar en algo la ejecución total. No obstante no sirve encriptar una contraseña si hay keyloggers indetectables capaces de copiar los movimientos de tu teclado, para solucionar esto se utilizara los módulos de OpenCV python para hacer un registro de la cara del usuario y hacer uso del registro para cuando inicie sesión, de esta forma no se permitirá entrar si la cara que registra la contraseña no es la misma a la registrada en el sistema. De esta forma, se plantea la siguiente pregunta de investigación:

¿De qué manera se puede programar un protocolo de seguridad para una aplicación, usando estructuras de datos, el cifrado de Elgamal y OpenCV?

Justificación

La web, un gran mundo que tiene como objetivo guardar y compartir la información de los usuarios, esto respetando que son conscientes de lo que hacen, pero qué pasa, cuando la información que se almacena es delicada, como puede el usuario estar seguro de lo que guarda en la red y no quiere compartir, en verdad está siendo vista por alguien más. En principio gran parte de los usuarios creen que la red es algo totalmente seguro y que no tienen que preocuparse, pero por mi parte, yo si me preocupo al leer del ataque del Bundestag alemán, donde se filtró información financiera, chats, etc., o conocer a John the Ripper, un software programado para romper con los algoritmos de encriptación, keyloggers, capaces de registrar tus movimientos de teclado. Con todo esto, anexado por mi pasión por la programación, se decide crear un protocolo de seguridad, que sea capaz de evitar la descriptación por parte del administrador o de algún tercero que haya interceptado el mensaje.

Objetivo

- Programar un protocolo de seguridad para una aplicación, usando estructuras de datos, el cifrado de Elgamal y OpenCV

Marco teórico

Entrando en tecnicismos, para programar un algoritmo, primero se tiene que hablar de varios conceptos relacionados a su rendimiento.

El primer concepto es el Time Complexity de un algoritmo, el cual define (Pandey, 2022), como la cantidad de tiempo en el que un algoritmo se tarda en ejecutar una función de la longitud de entrada, sin tener en cuenta la ejecución real de la máquina. En pocas palabras, es el tiempo en el cual los datos se demoran en pasar por todo el algoritmo, ignorando la capacidad de ejecución de la computadora.

Otro concepto importante es el Space Complexity el cual representa el espacio de memoria RAM que usa un algoritmo para ser ejecutado con respecto al tamaño de entrada y espacio auxiliar si es necesario, explica (GeeksforGeeks, 2022).

El tercer concepto es el Big O Notation que representa qué tan rápido es un algoritmo para procesar todos sus datos, para hacer la clasificación utiliza la estructura de $O(n)$, donde n es el número de operaciones que tiene que hacer el algoritmo en su peor caso (Salton, 2019).

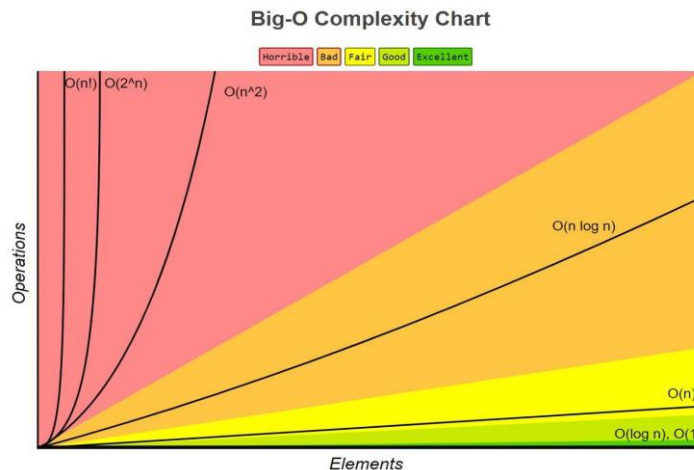


Figura 1: Comparación de Big O Functions.
Imagen de (Salton, 2019)

Estructuras de Datos

Un término bastante importante y que está ligado a la pregunta de investigación son las estructuras de datos y se definen como algoritmos especializados en almacenar y organizar grandes cantidades de datos, con el propósito de realizar operaciones más eficaces al momento de buscar, insertar, actualizar y eliminar de valores, expone (Mallawaarachchi, 2020).

La estructura de datos que se va usar es linked list, definido por (Programiz, 2021), como una estructura de datos lineal¹, donde no se almacena en ubicaciones de memoria contiguas, en cambio usa nodos que almacenan datos y la dirección del siguiente nodo.

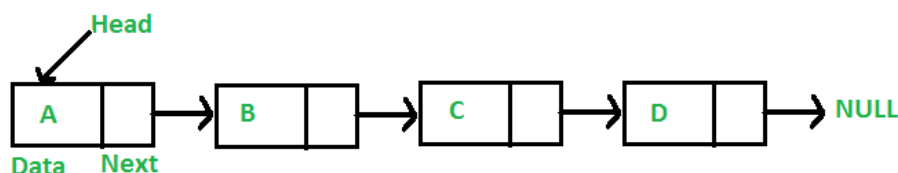


Figura 2: Representación de Linked List.
Imagen de (GeeksforGeeks, 2022)

¹ Estructura de datos lineal: "Una estructura de datos lineal tiene elementos de datos conectados entre sí, de modo que los elementos se organizan de manera secuencial y cada elemento está conectado al elemento que está delante y detrás de él.", explica (Samad, 2022).

Para empezar con el código de encriptación de elaboración propia, se le pide al usuario que suba 20 fotos de su rostro en diferentes posiciones, y las guarde en una carpeta con su nombre en el archivo images, para qué el script de faces-train.py, tomado de (Coding For Entrepreneurs, 2018), haga el entrenamiento de reconocimiento facial, para saber más sobre cómo funciona este script **ver anexo 6**, posteriormente se ejecuta el archivo main.py, que empieza llamando al código faces.py tomado de (Coding For Entrepreneurs, 2018), para saber más sobre cómo funciona este script **ver anexo 7**, el cual consigue el índice de similitud entre el rostro en cámara y el rostro registrado. Posteriormente se crea una matriz de $A * A$, siendo A, el índice de similitud. Pero para completar los valores que están dentro de la matriz, se hace uso de la variable b y c, adicionalmente se guarda las ubicaciones X y Y de la matriz con los valores a encriptar, para ser enviadas a b.py, donde se hace su encriptación con el cifrado de Elgamal, tomado de (Panda, 2021), **ver anexo 2** para entender su funcionamiento, y los valores retornados, se convierten en un linkedlist que rotara k posiciones, siendo k el índice de similitud.

Para la desencriptación, se hará todo un proceso inverso, pero con una autenticación de ± 2 puntos de error en el índice de similitud.

Metodología

Para el desarrollo de este trabajo monográfico, se implementará una metodología con enfoque Triangular o mixto siguiendo una ruta específica para la creación del código.

Discusión y Análisis

Partiendo con el código del protocolo de seguridad, el proceso de encriptación, se realiza con el archivo main.py, donde primero se importa la librería Pickle, con la finalidad de serializar y deserializar la estructura de objetos de la variable conf, la cual después de haber llamado al script faces.py, contiene el índice de similitud entre el rostro de la persona registrada y el rostro que aparece en cámara, la idea de usar esta librería, fue para contener el objeto y que este fuera llamado en otro script para reconstruir el objeto en la desencriptación, sin incurrir en cambios frecuentes de la variable conf, que se producen cada vez que se llama al archivo de desencriptación. En las siguientes líneas se esquematiza lo mencionado anteriormente:

```

with open('{path_file}/cache/conf.conf', 'wb') as data:
    pickle.dump(conf, data)

with open('{path_file}/cache/conf.conf', 'rb') as data:
    conf = pickle.load(data)
    conf = conf.decode('ascii')
    conf = int(conf)

```

Para los siguientes pasos de encriptación, se creó la variable pasw, que vendría a ser la unión de la contraseña con el mensaje a encriptar, y su creación se justifica para la autenticación en la desencriptación más adelante y para su almacenamiento en la variable b, la cual va a pasar todo el formato string a integer para su posterior encriptación. Continuando con las variables, la variable power, guarda la longitud del input del usuario elevado la longitud del input de la contraseña, esta acción se realizó a fin de tener un valor constante con el cual multiplicar los ítems del ciclo for en pasw, de esta forma en el primer condicional, se usa la función .isdigit() para identificar si el ítem es un dígito, y la razón de su uso, está en que la función atraviesa cada ítem usando el valor ASCII, permitiendo que su iteración tenga un time complexity de $O(n)$, explica (Saxena, 2022). Posteriormente se revisa que si el ítem es igual que cero, ya que si lo es, se sobrescribirá como la longitud del usuario, debido a que el almacenamiento en la variable de tipo array llamada b, se va a guardar la multiplicación entre el carácter tipo int multiplicado por power, aunque en el else statement, en caso de que el carácter no sea un dígito, se usa el método ord(), el cual convierte un ítem de tipo string, en un carácter de tipo int, a partir del estándar global Unicode, siempre y cuando el ítem sea de 8 bits, expone (Przywóski, 2015), por lo que en la variable b, se guardará el Unicode del ítem multiplicado por power. Con todo esto, ya se tiene almacenado una primera distorsión de la contraseña, la cual posteriormente va a ser guardada en una matriz con más valores.

```

usr = input("Ingrese nombre de usuario: ")
pass1 = input("Ingrese Contraseña: ")
msg1 = input("Ingrese Mensaje: ")

pasw = pass1 + ": " + msg1
power = len(usr) ** len(pasw)

b = np.array([], dtype = int)
c = np.array([], dtype = int)

```

```

for i in pasw:
    if i.isdigit():
        a = int(i)
        if a == 0:
            a += len(usr)
        b = np.append(b, (a*power))
    else:
        b = np.append(b, (ord(i) * power))

```

Ahora bien en la variable curr_time se almacena el tiempo en milisegundos multiplicando por mil, redondeado al valor entero más cercano, con el propósito de tener un rango aleatorio, ya que en el ciclo for loop, el rango de repetición, va ir desde (conf *conf), representando al tamaño de la matriz, menos len(pasw), debido a que se necesita valores que complementen lo que falta para llenar la matriz, sabiendo que la longitud de pasw, ya hace parte de la matriz, por lo que continuando con curr_time, éste será puesto en la variable a, el cual escoge de manera aleatoria un número entero entre 1 y curr_time, de esta forma se asegura que cada vez que el código es ejecutado, se obtenga valores totalmente aleatorios que se guardan en la variable c de tipo array. Ahora en la variable array, se van a concatenar las dos variables b y c, puesto que de no solo tener ya los datos que van a hacer parte de la matriz, se hace uso del método shuffle() para reorganizar los elementos del array de manera aleatoria, explica (W3schools, 2022), con el fin de que a la hora de hacer uso del método reshape en la variable array y pasados los parámetros, se crea una matriz de tamaño conf por conf, con el objetivo de contener de forma aleatoria, las distorsiones de los valores de la variable pasw, junto con los valores aleatorios de la variable c, un ejemplo del resultado se puede ver en el **anexo 1**.

```

curr_time = round(time.time()*1000)

for i in range((conf*conf)- len(pasw)):
    a = random.randint(1, curr_time)
    c = np.append(c,a)

array = np.concatenate((b,c))

random.shuffle(array)
array = np.reshape(array, (round(conf), round(conf)))

```


Terminando con la variable array, para que quede guardada para su posterior descriptación, se hace uso del módulo pickle para serializar el array en un documento con la dirección path_file, con el nombre times1, que refiere a un variable que contiene la fecha de ejecución del programa, con una extensión .data para hacer una diferenciación con otro archivo más adelante que va a tener el mismo nombre pero con otra función.

```
with open(f'{path_file}/cache/{times1}.data', 'wb') as data:
    pickle.dump(array, data)
```

Una vez que se tenga la matriz con los datos de la contraseña y valores aleatorios, se procederá con un for loop en la variable b, donde se usa el método where() en cada ítem iterado, para guardar en la variable index_x e index_y, la position que estas tienen en la matriz, de tal forma que se encripta la dirección de X y Y de los ítems de la contraseña en la matriz, con el propósito de descriptarlas posteriormente y facilitar su ubicación. Aunque para evitar combinaciones entre estos dos elementos, se usa la variable str1, para unir ambos elementos con un espacio intermedio entre ambos, de esta forma se garantiza que no importa si la posición del elemento X sea 1 y la del elemento Y sea 13, y que al unirlos quede 1 13, que por lo que vemos no se puede distinguir si la posición en X es 1 o es 11, por lo que en la separación con espacio en la tupla, quedaria '1 13', haciendo distinción entre ambos valores. Continuado la encriptación, se va a hacer uso de main(), que es una función importada del archivo b.py, la cual retorna cuatro parámetros después de la encriptación con el algoritmo Elgamal, finalmente, estos elementos son guardados en la variable d de tipo array.

```
c = ''
d = []
for i in b:
    index_x, index_y = np.where(array == i)
    c = str(index_x[0]),str(index_y[0])
    str1 = ' '.join(c)
    m = main(str1)
    d.append(m)
```

Con la variable *d* y *conf*, son mandados como parámetros en a la función linked que es importada de c.py, con la finalidad de convertir una lista tipo array en una tipo linkedlist, con el cual se pueda mover *conf* posiciones adelante, es decir que si la lista es [1,2,3,4,5,6] con *conf* 3, ahora será 4,(5,(6,(1,(2,(3,None))))).

```
linked1 = linked(d, conf)
```

Para lograr el cometido, primero se inicia con la función `linked`, la cual al recibir dos parámetros, el primero es el array a convertir y el segundo el número `conf` que moverá el `linkedlist`. Pero para hacer la conversión, primero se pensó en hacer uso del un ciclo `while`, el cual al iterar por cada elemento, mandaría a la función `Node()` todo lo iterado, pero el problema iba en que en vez de hacer que cada `Nodo` fuera un valor diferente por cada ítem del primer parámetro, se obtiene un solo `Nodo` con toda la información, haciendo que fuera imposible hacer una iteración independiente a cada elemento, por lo que esta solución fue descartada. Pero la solución llegó con un truco que usa (Joma Class, 2020) en su solución al problema `Intersection of Two Linked List`, el cual hace uso de dos `pointers` como una forma para iterar de manera correcta un `linked list`. De esta manera el primer `pointer` que es `l1`, va a ser el `Nodo` cabeza que tiene como valor el primer ítem del primer parámetro, así mismo el segundo `pointer` es `head` y este es igual a `l1`, por lo que en el ciclo `for`, el rango va a ser dentro de 1 (porque ya tenemos el primer valor en un `Nodo`), hasta la longitud del primer parámetro, por consiguiente como el valor `.next` del primer `nodo` está vacío, entonces se asigna un nuevo `Nodo` con el valor de la iteración del primer parámetro, en otras palabras el `Nodo` pasa de `(1,(None))` a `(1,(Node(2,(None))))`, pero para no hacer que el `Nodo` reescriba el `.next` y en vez de eso itere, se asigna que `head` sea igual a su siguiente valor y que se continúe con el ciclo `for`, pasando así de `(1,(Node(2,(None))))` a `(2,(Node(3,(None))))` y finaliza retornando la lista tipo `linkedlist` y el segundo parámetro a la función `rotate` de (Joma Class, 2020), la cual va a ser que el `linkedlist` rote `K` posiciones hacia adelante. Ver **anexo 3**, para ver cómo funciona la función `rotate` y ver el **anexo 4** para mirar un ejemplo del resultado obtenido del `linked list` después de la función `rotate`.

```
class Node:
    def __init__(self, value1=None, next=None):
        self.value1 = value1
        self.next = next

    def __repr__(self):
        return f"{self.value1}, {self.next}"

def rotate(list1, k):
    length = 0
    current = list1
```

```

while current != None:
    length += 1
    current = current.next
k = k % length
faster, slower = list1, list1
for _ in range(k):
    faster = faster.next
while faster.next != None:
    faster = faster.next
    slower = slower.next
faster.next = list1
head = slower.next
slower.next = None
return head

def linked(a,b):
    llist = Node(a[0])
    head = llist
    for i in range(1,len(a)):
        head.next = Node(a[i])
        head = head.next
    return rotate(llist, b)

```

Posteriormente con el uso de la librería pickle, se serializa el linkedlist rotado, que fue almacenado en la variable linked1, con la extensión .node y nombre de times1, con la finalidad de tener dos archivos con el mismo nombre, pero con diferente extensión, que hacen referencia a una misma ejecución de código.

```

with open(f'{path_file}/cache/{times1}.node', 'wb') as node:
    pickle.dump(linked1, node)

```

Para terminar por completo con la encriptación, se decidió hacer uso de la base de datos Sqlite3, debido a que (Koulianos, 2020), explica que se ejecuta en un archivo ligero, el cual no necesita de servidor o instalación RDBMS, adicionalmente su lectura y escritura, se llevan directamente en el mismo archivo con extensión .db, y al ser un archivo autónomo, este puede ser movido o copiado y ser compatible con todas los sistemas operativos. Pero hubo un problema y fue que

Sqlite3, no puede guardar clases, incluso si estas son declaradas tipo Blob², imposibilitando así almacenar linked1 y array, por lo que se decidió hacer uso de algo común que tienen ambas clases, y es su nombre, ocasionando que la base de datos solo guarde el usuario y el nombre de las clases.

```
connex = sqlite3.connect('user_encrypted_data.db')
try:
    connex.execute('''
        CREATE TABLE DATA
            (ID TEXT          NOT NULL,
             NODO             BLOB    NOT NULL);''')
    print("Tabla creada exitosamente")
except sqlite3.OperationalError:
    print("La tabla DATA ya existe")
connex.execute("INSERT INTO DATA (ID,NODO) VALUES (?,?)", (usr, times1))
connex.commit()
```

Empezando con la descryptación, se encontró con un problema y fue en el llamado a la función send() de faces.py para obtener el valor de la variable conf, ya que cada vez que se hacía el llamado, se ejecutaba todo el código, provocando que se reescribiera el archivo .conf y que no se pudiera hacer la comparación de la similitud entre el rostro guardado con el rostro que aparece en cámara, por lo que para solucionar el problema, se decidió hacer una copia del archivo faces.py y llamarlo faces1.py, de esta forma cuando se ejecutaba el archivo, se obtiene el índice de similitud y se guardaba en la variable a, mientras que en la variable conf, se deserializa el índice con el que se hizo la encriptación, adicionalmente se crea la variable continuel con el valor False, con el motivo de hacer la verificación entre ambos índices de similitud.

```
from faces1 import send2

a = send2()

continuel = False

with open('{path_file}/cache/conf.conf', 'rb') as data:
```

² Blob: Hace parte de los tipos de datos de SQLite y almacena datos exactamente como fueron ingresados, sin especificar el tipo de dato; explica (SQLite, 2022)

```

conf = pickle.load(data)
conf = conf.decode('ascii')
conf = int(conf)
print("B: ",str(conf))

```

Para hacer el proceso de comparar ambos índices de similitud, se va a tener un rango, donde la variable b, la cual contiene el inicio del rango con a-2, va a pasar por un ciclo while que tiene como límite máximo a+2, dejando así un rango de tres números, los cuales si alguno coincide con el valor de conf, se asigna True a continuel, dando así seguridad de verificación de similitud y continuando con la descryptación. Y la razón de porqué asignar un rango, fue debido a que el índice de similitud en la variable a, puede cambiar debido a factores como la iluminación y posición del rostro que aparece en cámara, por lo que después de 20 pruebas, con diferentes iluminaciones y posiciones, se hizo un cálculo de incertidumbres que dio como resultado ± 2 unidades de similitud (**ver anexo 5**), ocasionando que si el valor de conf en la similitud de la encriptación, era de 54, en la descryptación, el índice de similitud con el rostro que aparece en pantalla debe estar dentro del rango de 52, 53, 54, 55 y 56.

```

b = a-2

while b < a+2:
    print(b)
    if b == conf:
        continuel = True
        break
    else:
        b += 1

```

El uso del while continue, sirve para verificar si se hizo la asignación True en la verificación de la similitud de rostros, en caso de no ser así, el while no se inicializa y finaliza la ejecución de código. Por otra parte se le va a pedir al ejecutor que inserte el nombre del usuario que hizo la encriptación, los datos a descryptar y la contraseña. De esta forma se almacena en la variable matrix, la deserialización del archivo .data con el nombre seleccionado, obteniendo la matriz a descryptar, así mismo la variable linked contiene la deserialización del archivo .node del nombre seleccionado, para adquirir las posiciones encriptadas de los ítems contenedores del mensaje. Por último está la variable linked2, la cual pasa linked y conf como parámetros a la función rotate_list_inv del archivo c.py.

```

while continue1:
    conn = sqlite3.connect('user_encrypted_data.db')
    cursor1 = conn.execute("select ID, NODO from DATA")
    for fila in cursor1:
        print("Usuario: ", f'{fila[0]}', "      Data: ", fila[1])
    conn.close()

    usr1 = input("Usuario a usar: ")
    slec = input("Datos a Desbloquear: ")
    pasw1 = input("Contraseña a Desbloquear: ")

    with open(f'{path_file}/cache/{slec}.data', 'rb') as data:
        matrix = pickle.load(data)

    with open(f'{path_file}/cache/{slec}.node', 'rb') as node:
        linked = pickle.load(node)

    linked2 = rotate_list_inv(linked, conf)

```

Como la variable linked tiene items movidos conf posiciones, se necesita devolverlos a su posición original para descryptarlos y obtenerlos de forma correcta las posiciones con las que se van a buscar los items en la variable matrix, por lo que para lograr el cometido, se buscó una forma en la que los elementos se muevan otra vez k posiciones para que queden en la posición original, pero para saberlo, primero se tiene que saber la longitud del linkedlist que fue pasado, así que se hace uso la variable length para que cada vez que en el ciclo while se este iterando, la variable incrementa por uno hasta que el siguiente ítem del nodo sea None. Una vez con la longitud, podemos ver que para sacar el número de veces que se mueven los nodos en la función rotate, se hace uso de la variable k para contener la operación $k \% ^3 \text{ length}$, donde k es el número de veces a mover y length, la longitud del linkedlist, obteniendo de esta forma que $5 \% 10$ es igual 5 y que $6 \% 10$ es igual a 6, y así sucesivamente para todos los números enteros que sean menores a length, hasta que $10 \% 10$ es igual a 0 y $11 \% 10$ es igual a 1 y sigue de la misma forma cuando k es mayor que length, de esta forma se puede saber que tantos puestos hay que moverse para llegar a la posición original, ya que si sacamos la diferencia entre length y k, en la variable k, podemos ver dos casos, donde $2 \% 7$ es igual a 2, entonces significa que los items se movieron 2 puestos, pero obteniendo la diferencia, se puede ver que se necesitan 4 puestos

³ %: También conocido como módulo operador, devuelve el residuo de la división de dos números; aclara (Van Schooneveld, 2020)

para volver a su posición original, es decir que si el linked list 6->7->1->2->3->4->5, que se movio 2 puestos, ahora necesita obtener su diferencia para saber que se tiene que mover 5 veces para estar en el origen, aunque ocurre algo similar si k es mayor que length, ya que la diferencia al ser negativa y pasar por $k \% \text{length}$, el resultado es igual a la diferencia faltante para que k sea igual a length, por lo que si 6->7->1->2->3->4->5 que ahora se movio 9 veces y su diferencia es -2, entonces $-2 \% 7$ es igual a 5, y es ese resultado el número de veces a mover para estar en el origen, terminando así la función con el retorno del linkedlist que fue movido k veces en la función rotate que tuvo como parámetros k y el linkedlist a mover.

```
def rotate_list_inv(l1, k):
    length = 0
    current = l1
    while current != None:
        length += 1
        current = current.next
    k = length - k
    return rotate(l1, k)
```

Una vez que tenemos el linkedlist en el orden correcto, se tiene que descifrar con el cifrado Elgamal, las posiciones que necesitamos para encontrar los datos en la variable matrix, para ello se manda los cuatro parámetros a la función main1 de g.py; con los valores de posicionamiento, se separan para no tener errores de búsqueda en matrix, por lo que para lograr el objetivo, se hace un ciclo for loop, el cual al detectar un espacio en blanco, guarda en la variable a2 los datos que son siguientes al índice del espacio y en la variable a1, los valores que están por detrás del índice.

```
matrix_b = np.array([], dtype = int)
current = linked2
while current != None:
    wa = main1(current.data[0], current.data[1], current.data[2],
current.data[3])
    a1 = ''
    a2 = ''
    for id in range(len(wa)):
        if wa[id] == ' ':
            a1 = int(wa[:id])
            a2 = int(wa[id+1:])
```

```

matrix_b = np.append(matrix_b, matrix[a1][a2])
wa = ''
current = current.next
power1 = len(usr1) ** len(matrix_b)

```

Para terminar de descryptar, se necesita primero cambiar en la variable pasw1 cualquier valor 0 por la longitud del usuario, debido a que se hizo lo mismo en la encriptación y para no tener discordancias en los datos descryptados. Finalmente se usa la variable power1, la cual contiene la longitud del usuario elevado a la longitud de la matrix_b, para hacer la división con el ítem a descryptar, de esta forma se pueden obtener dos posibles valores, uno donde el valor es menor a 9, y significa que el valor a descryptar es un número, y otro cuyo valor sea mayor a 9, siendo este un valor que debe pasar por la función chr() para ser convertirlo en su valor de string, por último como el string descryptado contiene la contraseña y el mensaje, se hace una verificación donde el input de la contraseña tiene que ser igual a la descryptada, para proceder con mostrar el mensaje.

```

def final_msg(a,b):
    final_msg = ''
    for i in a:
        ay = round( i / power1)
        if ay <= 9:
            final_msg += ''.join(str(ay))
        else:
            final_msg += ''.join(chr(ay))
    if final_msg[:len(b)] == b:
        print("[Mensaje]: ", final_msg[len(b)+1:])
    else:
        print("Contraseña Incorrecta!")

for i in pasw1:
    if i == '0':
        pasw1 = pasw1.replace('0', str(len(usr1)))
    final_msg(matrix_b, pasw1)
continuel = False

```

Finalmente se puede observar todo el código completo en el **anexo 8**, con su respectiva organización de archivos.

Conclusiones

Como consideraciones finales y dando respuesta a la pregunta de esta monografía, la manera en la que se puede programar un protocolo de seguridad para una aplicación, usando estructuras de datos, el cifrado de Elgamal y OpenCV, es en hacer como primer paso una verificación de similitud entre rostro en cámara y el rostro de encriptación, haciendo uso de la librería de OpenCV, posteriormente usar una matriz que contenga de manera distorsionada los datos a encriptar, así mismo las direcciones de los datos en la matriz, son encriptados con el cifrado de Elgamal, y finalmente se hace uso de la estructura de datos linkedlist, para mover k posiciones dependiendo del índice de similitud del rostro; esta fue una propuesta realizada por fuente propia, haciendo uso de los tres elementos anteriormente descritos más códigos de autores ya mencionados; con la finalidad de ofrecer una mayor seguridad con los datos del usuario.

Bibliografía

- Coding For Entrepreneurs. (2018, Diciembre 20). *OpenCV-Python-Series*. GitHub.
<https://github.com/codingforentrepreneurs/OpenCV-Python-Series/blob/master/src/faces.py>
- Coding For Entrepreneurs. (2018, Abril 17). *OpenCV-Python-Series*. GitHub.
<https://github.com/codingforentrepreneurs/OpenCV-Python-Series/blob/master/src/faces-train.py>
- GeeksforGeeks. (2022, Septiembre 7). *What does 'Space Complexity' mean?*
GeeksforGeeks. <https://www.geeksforgeeks.org/g-fact-86/>
- GeeksforGeeks. (2022, Octubre 25). *Linked List Data Structure*. GeeksforGeeks.
<https://www.geeksforgeeks.org/data-structures/linked-list/>
- Joma Class. (2020, Agosto 1). *Intersection of two linked list*. Joma Class.
<https://www.jomaclass.com/blog/intersection-of-two-linked-list>
- Joma Class. (2020, Agosto 12). *Rotate Linked List*. Joma Class.
<https://www.jomaclass.com/blog/rotate-linked-list>
- Koulianos, P. (2020, Agosto 7). *5 Reasons to Use SQLite the Tiny GIANT for Your Next Project*. Medium. <https://medium.com/swlh/5-reasons-to-use-sqlite-the-tiny-giant-for-your-next-project-a6bc384b2df4>
- Mallawaarachchi, V. (2020, Febrero 27). *8 Common Data Structures every Programmer must know | by Vijini Mallawaarachchi*. Towards Data Science.
<https://towardsdatascience.com/8-common-data-structures-every-programmer-must-know-171acf6a1a42>
- Panda, S. (2021, Octubre 20). *ElGamal Encryption Algorithm*. GeeksforGeeks.
<https://www.geeksforgeeks.org/elgamal-encryption-algorithm/>
- Pandey, U. (2022, Julio 15). *Time Complexity and Space Complexity*. GeeksforGeeks.
<https://www.geeksforgeeks.org/time-complexity-and-space-complexity/>
- Programiz. (2021). *Linked List Data Structure*. Programiz.
<https://www.programiz.com/dsa/linked-list>

Przywóski, J. (2015). *ord — Python Reference (The Right Way) 0.1 documentation*. Python Reference (The Right Way). <https://python-reference.readthedocs.io/en/latest/docs/functions/ord.html>

Salton, K. (2019, Marzo 4). *Understanding time complexity with Python examples | by Kelvin Salton do Prado*. Towards Data Science. <https://towardsdatascience.com/understanding-time-complexity-with-python-examples-2bda6e8158a7>

Samad, A. (2022). *What are linear data structures?* Educative.io. <https://www.educative.io/answers/what-are-linear-data-structures>

Saxena, A. (2022, Agosto 18). *Python String isdigit() Method*. GeeksforGeeks. <https://www.geeksforgeeks.org/python-string-isdigit-method/>

SQLite. (2022, Abril 12). *1. Datatypes In SQLite*. SQLite. <https://www.sqlite.org/datatype3.html>

Van Schooneveld, J. (2020). *Python Modulo in Practice: How to Use the % Operator – Real Python*. Real Python. <https://realpython.com/python-modulo-operator/>

W3schools. (2022). *Python Random shuffle() Method*. W3Schools. https://www.w3schools.com/python/ref_random_shuffle.asp

Anexos

Anexo 1: Matriz de la variable array con parámetros de conf = 29, usr = Dh, pass1 = Dh205@, msq1 = Hola

[1636186696092	414323622083	245183036323	155845459489	761797838084	801222637673	237044772689	213763001550	1477549662953
	1584794490071	1557402874383	1613636851308	1120190218900	674594293497	1422480457936	1205908160668	941597593614	1256818835948
	1611977542897	496133002388	1019242759847	77887498548	487113551030	491337875576	1416616797088	57339873693	834037759639
	62802071455	249806302345	643877826908	207079346207	1504059784125	1639889723752	1171845666694	247687638080	1340125959423
	613657157224	166580746836	425984	17581453893	201833739909	878292439554	1346313382439	294912	1047879967804
	148122707306	1454600522793	1086225165714	1611900909881		117431818185	1389237789296	1511482419980	1534909085629
[118517041521	29494604235	1046679788281	282920086158	271289004795	1116514467877	768453934325	1151134908221	723882077072
	8192	106594500965	1128444591685	1012021949968	1398559173870	1441806552392	1375783552866	1397769416248	1336729383727
	992033771953	11285669322	354843205824	996555482496	1662637992389	74445871076	306331813464	361933695013	683120415424
	1476261860989	526124845616	507079048162	1555633798161	294911551510	1457527601536	731214811074	1328606197379	506800805398
	332330028820	129280555197	67838191567	1243791132547	469832429820	1333112982078	1557633615691	1473287508192	1621342784325
	1633482914650	1288109406288	510344153418	399163698996		1276007217601	65234204161	38802371034	717031157395
	721217790354	1207838691516	823590327735	38592733522	816200553022	294587555845	939181749422	758420341499	447237834916
	1657775096395	1643304305642	681552676264	55959674521	1174457473080	1040658108293	1313879709680	163618590314	1627404956313
	914747283522	1204161138460	397312	892467805423	518174771145	1175535611935	363554828486	1540856194878	442368
	820240474759	635198188536	458808498625	881148238036	1515778542250	1399261481766	780844895492	203795396061	847035332517
	576594521114	633275929057	1618671221137	752386167300	912976556392	266005590133	1630794654848	1041530129733	1664047289673
	675832671624	855170663026	893828429999	454656		983745318054	367688132650	272802630459	830655269644
	313884425866	1580845800557	9171244476	689963355486	569521575942	49675638503	1384415183671	108291387410	18659256566
	825860997909	511022015370	1453174126231	517792008377	1351154315639	[851781714954	1570484621497	105524840406
	493448699298	150645929566	938554063831	29127895379	1228698488742	1235234065098	34534574657	631994259962	1581261303250
	777238294380	1105825564382	559359765599	45869155859	257284616844	1617212496498	473790102995	399293695813	489174799284
	1515388440333	1458993928908	45127658036	433746296993	16354868958	1362590931425	1543171181717	1003306640588	1073082661606
	514141235029	440442451385	122488092246	1189121347439		133896180076	86051435398	15433909619	23625467334
[1162081237267	1250401730623	573434045722	1452955618217	610090589895	717934245178	886681472136	1478564693587	1386791584507
	480696019883	823964396707	148986944786	1379040177752	269974572671	885567746712	360493736512	718131212555	94104745981
	1079320685574	12207217964	1149899466535	498578003256	1324187558851	1309470110495	364326724174	53587508595	731784228872
	1452567699014	1268848951194	1608615225154	191937726751	808985999662	1652001241899	262144	108757039967	805163893004
	1623462323002	488448976417	23556479066	609210113163	464059255336	1493742085518	14088284550	1235707743377	84453619585
	8192	1309247923469	889435768605	471989434465		1529540791008	1650516485369	1538235664831	282364667921
	248530264778	1234037559728	31352069129	66695132422	560993748690	332353939811	1220688740361	1604104503068	1385941668548
	830527840864	1118649594780	1188483622786	1389757234593	1102632078898	1440145547612	65945140478	1323804312404	1535229176077
	1154219739570	340597853055	432136940077	999790842193	1191780367184	1320168204291	127699780626	204269708385	1315964314882
	729406291660	1480589649196	876758687729	1060394648558	1365973705745	1234503911127	453814227731	1087464980491	545871504335
	93003796578	858032521307	128693848214	1511674261957	17683163478	367597527435	169903844476	560662686938	266512006925
	1237701538123	1237715634590	1301325407029	1329505302451		4239295615379	304808646494	221310031606	141309047778
	385302132671	1494836767284	686614176160	130068681169	649110406343	289891460698	1393351703674	779550073000	1264815527542
	1331749677672	20088268533	2809909327821	1212547653733	640500201709	[912136468978	1592115630018	1474690645290
	1526946284149	1009585552368	188139620935	464317181846	938876801217	767356540434	1401148852918	243760029248	402634588591
	1135405618369	1461428992714	333917350033	18558122264	902008417147	95987869922	564997844051	1343977157861	1179757399112
	441110076459	713037623244	965399558215	621578283543	1405229387597	366645863526	1244382240003	42590799517	78561874619
	801222637673	237044772689	213763001550	1477549662953		250322521835	283885741107	1480115340806	61029353939
	250322521835	283885741107	1480115340806	61029353939	846534244236	1564655459032	415443012428	740691092859	52193711675
	238565555438	466747245727	638097963907	480482227875		388535724803	163665085059	796578669296	1226419132106
	1187218294808	1408645141641	977569656435	1585997412341	633052469948	1554347721089	573115371003	578337141216	1507490467214
	873475923894	1420053661176	428408480794	957501181516	1198861130749	[75780684887	1350494795594	473965221948
	1507658325601	873899246240	774678699539	1442220843099	161348061573	778016575602	1546849731651	235497115706	1549831349081
	357143622369	303740447517	1220320072433	1086760241784	133236142558	1037724985320	1025091464429	858404512600	935131751260
	1140805162930	589726875105	207244075727	1179368144323		1617328622306	1418143757199	440935442532	1542970583278
	718145196863	1592689824746	619703484072	255380042343	559652529328	1007753446167	577708024208	703495694158	278528
	802672699185	1239011025776	1268973698750	361899357754	501673892530	1523081293537	1456964640831	372449015677	901175178826
	110248186405	857178423046	478011859724	259524185906	1387128981076	949190310560	1210610791434	624420028666	1371501264160
	1640648504529	18080481438	1595199406451	1108471112595	442247366775	1593707366797	1162568989825	162955773831	138709741918
	1773309586237	193482044959	985046067309	1282658137698	1027336673540	416252164057	1102889839180	1498555430638	692408243883
	1615837069280	1340860924774	774115777257	361812594772		163099368123	1192752968982	526448175483	151290723937
[1359817515395	277113571814	1459911060774	1623146404735	1122003567526	733539256169	220738456246	452609045676	841150258267
	158910443037	1502218380460	1521463461761	568461628294	1320765526497	1619237951089	790403264909	1290111475365	1269182742351
	1090826619306	1206311785441	647369062842	887453866839	151527744815	[1600932984571	1226427634126	183507569012
	108990149069	82821053952	362696847653	1269087396245	1280206650236	1525072894455	146914791905	778905626820	912121087754
	128985934884	212367445552	184410417546	1596830456044	1432005033601	503705484512	873202570100	569431940336	582050897390
	1449446701273	596913880442	1549393528794	660757019531		416868777921	1027666032214	571702798044	1233904073599
[1365220025473	1238756902786	1195215275875	255556058286	1147572123553	895271372580	1385624351037	338262100263	1450065184065
	65079393851	20480	1412327206745	476640499053	293136982459	1214061010911	1021046372644	281283245806	1093085217051
	14204088360817	1478796535170	928660832664	387288806573	474161729016	[619448046013	1118726373072	708829999500
	222208839498	1020406660026	1184191686967	390163441158	1460357344754	205467170159	897963740191	127270200505	1283830678553
	912816101473	1339363538652	571901324122	94010528086	37532347588	1401733832630	471477059890	628018522223	1043928031231
	363331216274	316806053271	450085228332	307168610993		1284048702222	295438087827	391469641760	1063833053722
	75337992726	791001737266	643189423130	992631723550	439232097974	1406348561349	1068092492215	341343373823	1647431156278
	1676603127965	1021907916618	1172598093275	1075216094108	95428138412	81664909685	529088180452	1613597633959	905822049922
	1429335165588	125055984426	1346361502748	1501489616165	1104742486418	[4421304022	182991407784	1650988812197
	1066389541598	1003653401838	1191420757558	1486796773620	980318044922	356701917528	1002519773491	1068208364144	92148661354
	1081041066475	1645934366835	57692426421	370901005261	298525036548	642513512134	680692384274	494915914585	960205269939
	1518965380596	1349987759640	828953149265	494815737308		10075765223	450572578081	1587206816227	504415965266
	897364400594	715906156659	999074718734	31925498711	1459829472263	115257477317	737543393891	1314555874102	309361882797
	432171186681	1420154646564	271837640000	1359903847885	668775651166	[1510265309631	1279097668438	640890827200
	1224280543733	1215099251103	1291642297290	1431576446021	1614867944096	1663115190275	1580409739721	350883303822	119377612614
	1569092704489	1357443247000	437472792259	1060722369237</					

113103391049	791584779203	1469072749032	1010850870189	1053351576020
957432110191	174822528916	378287299875	990877666187	1286508098499
1524925737847	1582909917553	1513286342393	1145263625160	1056442197746
1254169649080	748576900447	1486802650731	1222782196748]	
[1226154862638	728211467572	1244574744059	882392971519	69595700372
726581113645	17268635617	909731057124	601678042703	475053194168
894074088318	742781768387	439313690531	889833488034	515336001035
1257276390351	282177110066	696784913722	1037753512437	1314780623919
129533748339	1406877795409	398239235950	131072	24338261400
473835563609	488499027444	343765446662	1317654157553]]	

Anexo 2: Funcionamiento del algoritmo Elgamal tomado de (Panda, 2021)

El algoritmo Elgamal, es un criptosistema de clave pública, que usa la clave simétrica para la dificultad el cálculo de logaritmos discretos en un grupo cíclico, siendo así difícil calcular el valor de g^{ak} , incluso si se sabe g^a y g^k , explica (Panda, 2021).

La idea del funcionamiento del el algoritmo Elgamal, dada por (Panda, 2021) es:

1. Bob genera claves públicas y privadas:
 - Bob elige un número q muy grande y un grupo cíclico F_q .
 - Del grupo cíclico F_q , elige cualquier elemento g y un elemento a tal que $\gcd(a, q) = 1$.
 - Luego calcula $h = g^a$.
 - Bob publica F , $h = g^a$, q y g como su clave pública y retiene a como clave privada.
2. Alice cifra los datos usando la clave pública de Bob:
 - Alice selecciona un elemento k del grupo cíclico F tal que $\gcd(k, q) = 1$.
 - Luego calcula $p = g^k$ y $s = h^k = g^{ak}$.
 - Ella multiplica s con M . Luego envía $(p, M*s) = (g^k, M*s)$.
3. Bob descifra el mensaje:
 - Bob calcula $s' = p^a = g^{ak}$.
 - Divide $M*s$ por s' para obtener M como $s = s'$.

Anexo 3: Funcionamiento de la función rotate tomado de (Joma Class, 2020)

Para rotar los ítems del un linkedlist, primero hay que conocer la longitud de este, para ello se atraviesa por todo el linkedlist hasta que el valor de `.next` sea igual a `None`, posteriormente se hace uso de la variable k , que con operador `%` calcula en número de posiciones x a mover dependiendo de la longitud y el número deseado a mover, con estos datos se crean dos pointers el primero avanza hasta la longitud de k , después el primero pointer se mueve hasta el final del linkedlist mientras que el segundo pointer hace lo mismo, una vez termina, el valor `.next` del primer pointer es igual a la lista original, la cabeza del linkedlist va a ser el valor `.next` del segundo pointer y al mismo tiempo va a cortar el linkedlist con un `None`, explica (Joma Class, 2020).

```
def rotate(list1, k):
    length = 0
    current = list1
    while current != None:
        length += 1
        current = current.next
    k = k % length
    fast, slow = list1, list1
    for _ in range(k):
        fast = fast.next
    while fast.next != None:
        fast = fast.next
        slow = slow.next
    fast.next = list1
    head = slow.next
    slow.next = None
    return head
```

Anexo 4: Resultado del linkedlist después de la función rotate con parámetros de conf = 29, usr = Dh, pass1 = Dh205@, msg1 = Hola

```
-----
([2777923688302642262343572241201589327686397275493200, 3111274530898959333824800910145780047008764948552384, 1777871160513691047899886234369017169719294256315648,
2777923688302642262343572241201589327686397275493200, 2833482162068695107590443686025621114240125221003064], 19685382089010263168297364182996923793411910273456, 6
1751199751652516651135981004639180949276765036171, 81827936696054159733143143704740493579233744015256), ([1436992710903444451959836836686110168765520551108855, 836
06848634382226594814159526464098190848320645152, 1280229869713977784473309181774898150354736490987889, 1436992710903444451959836836686110168765520551108855], 1766
9366178910227789039503678462098960661316867571, 2169635222214531581158008996177755292942840861614, 55105476125661969140582006362215547899149905272121), ([18015875
19358446030864442964277852995008388796577250, 1153016012389405459753243497137825916805368829809440, 1801587519358446030864442964277852995008388796577250, 201777802
1681459554568176119991195354409395452166520], 73857596691709335270094175891690322523912276326517, 31431120909118926094528752848275282747260360448201, 9154434255279
7553857192056966845090994508207106524), ([1496952918889057144285923412441608834971941893495264, 840394621130698747669290336809324258229862115646464, 14706905869787
22808421258089416317451902258702381312], 1200117321366632114317679684860820168037539860434, 41334948634724633420598868836602653113021674672545, 466540028539964573
61108288191817455651869612266714), ([3702061797907018945156187696946621236852470934353600, 2369319550660492124899960126045837591585581397986304, 362802056194887856
625306394300768881211542151666528, 3702061797907018945156187696946621236852470934353600], 6350482341323496661718899836101655423319380128072, 929675358147302673188
1611966636209304598992823833, 97749138672226449361261865955823185090735263802980), ([2206331106424529775815754340791429100535409063665450, 216220448429603918029943
9253975600518524700882392141, 1412051908111699056522082778106514624342661800745888, 2206331106424529775815754340791429100535409063665450, 2250457728553020371332069
427607257682586117244938759], 28860115056332842650685128819081976592149880893885, 8087056549426002332479841626728583307278055968327, 482981451653043822544967448526
1467822201864208196), ([573618059427889929096256268108363317675189084704576, 382412039618593286064170845405575545116792723136384, 59751881190405200947526694594621
17892449862990600, 5975188119040520094752669459462117892449862990600], 1994830515638578788202563542854281724755706560132, 889454413594017431125921986987648820
3989042941463, 26947979757402950873012347135708002047721371249060), ([963348710119003494513607559661241969559660346284276, 6291256882409818739680702430440763882838
59817981568, 1041989421149126228759616340041751518095142823531972], 20174437197370692078055350326077656810902107993960, 1731059816419684536468061781353432399954051
6101095, 23939957097604014722241593147819504823808839071708), ([188672505537358447165621313642322569596217401031961, 123214697494031572059795677625621065769548891
4959648, 204074342744897912240365910674348901808153515401917], 5508137934544414883885971195799608270119764048839, 443231337259763291343216888550168286787409698863
3, 51747029909798425506172386903231923462450422987830), ([1249265259899969511584649288239178773059839237244204, 140223651621425149259501450720724147996512567445778
0, 81584670034283723055281167829667770161527665139072, 137674130682853782903286970712564362147577934922184], 53611304248635336325106963704627644711415849013176,
26420410451662924135811994793630134342205051421713, 87517406108617083363035859826076909093024814226545), ([1477242999783012224809706629128753231419251015751359, 14
77242999783012224809706629128753231419251015751359, 964730122307273289671645145553471498069714949062112, 1477242999783012224809706629128753231419251015751359, 1477
242999783012224809706629128753231419251015751359], 46881086024971953024759580789587867315329063345911, 10274809726158815409843867000653494284047031125879, 86170156
565373833857716512508260264424097705113960), ([917367106809664598786135175369312795164786665451550, 917367106809664598786135175369312795164786665451550, 5871149483
58185343223126512236360188905463465888992, 8990819764673471306810412471861926539261490932142519, 954061791082051182737580582384085306971378132069612], 5127980897709
0460897562403825301044689487924889721, 62971997424163940687378028758576785500785485960278, 73885312516260688218592502930161167125598426574745), None
-----
```

Anexo 5: Incertidumbre en el índice de similitud

Tabla 1: Tabla con los resultados del índice de similitud

Prueba 1	34.28274124989236
Prueba 2	33.732949481936195
Prueba 3	33.16145798520633
Prueba 4	30.97051184927079
Prueba 5	34.38371571409794
Prueba 6	35.06895281437445
Prueba 7	32.05323101320842
Prueba 8	31.96332601704973
Prueba 9	31.654678820591368
Prueba 10	34.15722239811593
Prueba 11	32.60107970681687
Prueba 12	31.39216977623138
Prueba 13	30.55597569930113
Prueba 14	30.767472906324052
Prueba 15	32.56845912913467
Prueba 16	38.836783621119174
Prueba 17	32.388311957777105
Prueba 18	30.580305251837522
Prueba 19	32.11849312353868
Prueba 20	32.061164211613715

Fuente: Elaboración propia (2022)

$n =$ número de índices de similitud

$\Delta s =$ Incertidumbre en el índice de similitud

$$\Delta s = \sqrt{\frac{\sum_{i=1}^n (|prueba_i - \underline{prueba}|)^2}{n - 1}}$$

$$\Delta s = \sqrt{\frac{72.38142420990395}{20 - 1}}$$

$$\Delta s = \sqrt{3.809548642626524}$$

$$\Delta s = \pm 1.9518065074762212$$

$$\Delta s = \pm 2$$

Anexo 6: Explicación sobre el funcionamiento del script faces-train.py tomado de (Coding For Entrepreneurs, 2018)

El objetivo general de este código es hacer uso de la librería de cv2 para un archivo .yaml que contiene todo el entrenamiento de similitud de todas las caras guardadas en el archivo images, para ello hace uso de la función CascadeClassifier() para cargar una inteligencia artificial entrenada que puede reconocer cuando un objeto es una cara frontal y cuando no lo es, posteriormente revisa si cada archivo en images termina en jpg o png, puesto que estos son las únicas extensiones con las que funciona el entrenamiento, en caso de serlos, son puesto en la variable label, para asignar un id que es guardado en labels_id y con el cual se hace uso de detectMultiScale(), finalmente se identifican las posiciones del rostro en la imagen y se guarda en el id del rostro que está siendo identificado, aclara (Coding For Entrepreneurs, 2018).

```
import os
import cv2
import numpy as np
from PIL import Image
import pickle

BASE_DIR = os.path.dirname(os.path.abspath(__file__))

image_dir = os.path.join(BASE_DIR, "images")

face_cascade = cv2.CascadeClassifier('cascades/data/haarcascade_frontalface_alt2.xml')

recognizer = cv2.face.LBPHFaceRecognizer_create()

current_id = 0

label_ids = {}
y_labels = []
x_train = []

for root, dirs, files in os.walk(image_dir):
    for file in files:
        if file.endswith("png") or file.endswith("jpg"):
            path = os.path.join(root, file)
            label = os.path.basename(os.path.dirname(path)).replace(" ", "-").lower()
            print(label, path)

            if not label in label_ids:
                label_ids[label] = current_id
                current_id += 1
```

```
id_ = label_ids[label]
pil_image = Image.open(path).convert("L")
image_array = np.array(pil_image, "uint8")
faces = face_cascade.detectMultiScale(image_array, scaleFactor=1.5, minNeighbors=1)

for (x,y,w,h) in faces:
    roi = image_array[y:y+h, x:x+w]
    x_train.append(roi)
    y_labels.append(id_)

with open("labels.pickle", "wb") as f:
    pickle.dump(label_ids, f)

recognizer.train(x_train, np.array(y_labels))
recognizer.save("trainer.yml")
```

Figura 3: Código de faces-train.py.
Imagen de (Coding For Entrepreneurs, 2018)

Anexo 7: Explicación sobre el funcionamiento del script faces.py tomado de (Coding For Entrepreneurs, 2018)

Para poder crear el índice de similitud entre el rostro en cámara y en rostro registrado por faces-train.py, se hace una carga de la inteligencia artificial entrenada para reconocer rostros en la variable faces_cascade, para hacer que reconozca los rostros entrenados en el archivo .yaml, posteriormente se recorren todos los niveles de reconocimiento guardados en labels.pickle. Adicionalmente se convierte la imagen proyectada en cámara a color BGR2GRAY, con la cual se detectan multiescalas que pasan por un for loop para reconocer el rostro en cámara y compararlo con el registrado; y dejando el índice de similitud en la variable conf, finalmente se pone un recuadro con el nombre alrededor del rostro identificado, explica (Coding For Entrepreneurs, 2018).

```
import numpy as np
import cv2
import pickle

face_cascade = cv2.CascadeClassifier('cascades/data/haarcascade_frontalface_alt2.xml')
recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read("trainer.yml")

labels = {}
with open("labels.pickle", 'rb') as f:
    og_labels = pickle.load(f)
    labels = {v:k for k,v in og_labels.items()}

cap = cv2.VideoCapture(0)
while True:
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.5, minNeighbors=1)
    for (x,y,w,h) in faces:
        roi_gray = gray[y:y+h, x:x+w]
        id_, conf = recognizer.predict(roi_gray)
        print(conf)
        def send():
            return round(conf)
        if conf >=29 and conf <= 39 :
            font = cv2.FONT_HERSHEY_SIMPLEX
            name = labels[id_]
            color = (255,255,255)
            stroke = 2
            cv2.putText(frame,name, (x,y), font, 1, color, stroke, cv2.LINE_AA)
```

```
img_item = "my-image1.png"
cv2.imwrite(img_item, roi_gray)
color = (255,0,0)
stroke = 2
end_cord_x = x+w
end_cord_y = y+h
cv2.rectangle(frame, (x,y),(end_cord_x,end_cord_y), color, stroke)
cv2.imshow('frame', frame)
if cv2.waitKey(20) & 0xFF == ord('q'):
    break
cap.release()
cv2.destroyAllWindows()
```

Figura 4: Código de faces.py.
Imagen de (Coding For Entrepreneurs, 2018)

Anexo 8: Código completo de forma lineal con su respectiva organización de archivos

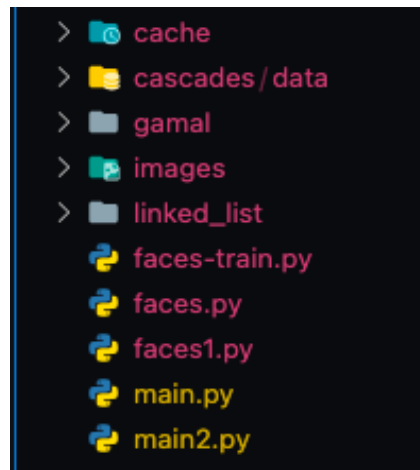


Figura 5: Nombre y ubicación de todos los archivos del código
Imagen de Elaboración propia(2022)

- La carpeta cache es el lugar donde se almacenan los archivos de tipo .data y .node
- La carpeta cascades/data almacena la inteligencia artificial entrenada con la que faces-train.py hace el reconocimiento de cara frontal; es obtenido de (Coding For Entrepreneurs, 2018)
- La carpeta gamal contiene los archivos b.py y g.py

```
import random
from math import pow

a = random.randint(2, 10)

def gcd(a, b):
    if a < b:
        return gcd(b, a)
    elif a % b == 0:
        return b;
    else:
        return gcd(b, a % b)

# Generación de grandes números aleatorios
def gen_key(q):

    key = random.randint(pow(10, 20), q)
    while gcd(q, key) != 1:
        key = random.randint(pow(10, 20), q)

    return key

# Exponenciación modular
def power(a, b, c):
    x = 1
    y = a

    while b > 0:
        if b % 2 != 0:
            x = (x * y) % c;
        y = (y * y) % c
        b = int(b / 2)
```

```

        return x % c
# Cifrado asimétrico
def encrypt(msg, q, h, g):

    en_msg = []

    k = gen_key(q) # Clave privada para el remitente
    s = power(h, k, q)
    p = power(g, k, q)

    for i in range(0, len(msg)):
        en_msg.append(msg[i])

    for i in range(0, len(en_msg)):
        en_msg[i] = s * ord(en_msg[i])

    return en_msg, p

def main(pasw1):

    msg = pasw1
    q = random.randint(pow(10, 20), pow(10, 50))
    g = random.randint(2, q)

    key = gen_key(q)
    h = power(g, key, q)

    en_msg, p = encrypt(msg, q, h, g)
    return en_msg, p, key, q

```

Figura 6: b.py

Imagen de (Panda, 2021)

```

import random
from math import pow

def power(a, b, c):
    x = 1
    y = a

    while b > 0:
        if b % 2 != 0:
            x = (x * y) % c;
            y = (y * y) % c
            b = int(b / 2)

    return x % c

def decrypt(en_msg, p, key, q):

    dr_msg = []
    h = power(p, key, q)
    for i in range(0, len(en_msg)):
        dr_msg.append(chr(int(en_msg[i]/h)))

    return dr_msg

def main1(en_msg, p, key, q):
    dr_msg = decrypt(en_msg, p, key, q)
    dmsg = ''.join(dr_msg)
    return dmsg

```

Figura 7: g.py

Imagen de (Panda, 2021)

- La carpeta images contiene las 20 imágenes del usuario que usa su rostro para encriptar y desencriptar.
- La carpeta linked_list se encuentra c.py.

```
class Node:
    def __init__(self,value1=None,next=None):
        self.value1 = value1
        self.next = next

    def __repr__(self):
        return f"{self.value1}, {self.next}"

def rotate(list1, k):
    length = 0
    current = list1
    while current != None:
        length += 1
        current = current.next

    k = k % length
    faster, slower = list1, list1

    for _ in range(k):
        faster = faster.next

    while faster.next != None:
        faster = faster.next
        slower = slower.next
    faster.next = list1
    head = slower.next
    slower.next = None
    return head

def linked(a,b):
    llist = Node(a[0])
    head = llist
    for i in range(1,len(a)):
        head.next = Node(a[i])
        head = head.next
    return rotate(llist, b)
```

```
def rotate_list_inv(l1, k):
    length = 0
    current = l1
    while current != None:
        length += 1
        current = current.next
    k = length - k
    return rotate(l1, k)
```

- El primero archivo en ejecutarse es faces-train.py, ya que empieza entrenando la inteligencia artificial para ser usada en faces.py.
- main.py es el archivo que empieza con la encriptación, donde {path_file} es el lugar donde están ubicados los archivos del protocolo de seguridad

```
import sys
import pickle
sys.path.insert(0, '{path_file}')

from faces import send

conf = str(send())
conf = conf.encode('ascii')
print(conf)

with open('{path_file}/cache/conf.conf', 'wb') as data:
    pickle.dump(conf, data)

with open('{path_file}/cache/conf.conf', 'rb') as data:
    conf = pickle.load(data)
    conf = conf.decode('ascii')
    conf = int(conf)

print(conf)
print(type(conf))

import random
import numpy as np
import getpass
import time
import datetime as dev
import sqlite3
```



```

sys.path.insert(1, '{path_file}/gamal')
sys.path.insert(2, '{path_file}/linked_list')

from b import main
from c import linked

usr = input("Ingrese nombre de usuario: ")
pass1 = input("Ingrese Contraseña: ")
msg1 = input("Ingrese Mensaje: ")

pasw = pass1 + ": " + msg1
power = len(usr) ** len(pasw)

b = np.array([], dtype = int)
c = np.array([], dtype = int)

for i in pasw:
    if i.isdigit():
        a = int(i)
        if a == 0:
            a += len(usr)
        b = np.append(b, (a*power))
    else:
        b = np.append(b, (ord(i) * power))

times1 = dev.datetime.now()
times1 = times1.strftime("%Y-%m-%d-%H:%M:%S")

curr_time = round(time.time()*1000)

for i in range((conf*conf)- len(pasw)):
    a = random.randint(1, curr_time)
    c = np.append(c,a)

array = np.concatenate((b,c))

random.shuffle(array)
array = np.reshape(array, (round(conf), round(conf)))
print(array)

```

```

with open(f'{path_file}/cache/{times1}.data', 'wb') as data:
    pickle.dump(array, data)

c = ''
d = []
for i in b:
    index_x, index_y = np.where(array == i)
    c = str(index_x[0]),str(index_y[0])
    str1 = ' '.join(c)
    m = main(str1)
    d.append(m)

linked1 = linked(d, conf)
print('[Encrypted Message]: ')

print('-----')

print(linked1)

print('-----')

with open(f'{path_file}/cache/{times1}.node', 'wb') as node:
    pickle.dump(linked1, node)

connex = sqlite3.connect('user_encrypted_data.db')
try:
    connex.execute('''
        CREATE TABLE DATA
            (ID TEXT          NOT NULL,
            NODO              BLOB      NOT NULL);''')
    print("Tabla creada exitosamente")
except sqlite3.OperationalError:
    print("La tabla DATA ya existe")
connex.execute("INSERT INTO DATA (ID,NODO) VALUES (?,?)", (usr, times1))
connex.commit()

cursor = connex.execute("select ID, NODO from DATA")

for fila in cursor:
    print(fila[1])

```

```
connex.close()
```

- main2.py es el archivo a ejecutar para hacer el proceso de descryptación del archivo seleccionado.

```
import pickle

import sys

sys.path.insert(0, '{path_file}')

from faces1 import send2

a = send2()

continuel = False

with open('{path_file}/cache/conf.conf', 'rb') as data:
    conf = pickle.load(data)
    conf = conf.decode('ascii')
    conf = int(conf)
print("B: ",str(conf))

b = a-2

while b < a+2:
    print(b)
    if b == conf:
        continuel = True
        break
    else:
        b += 1

while continuel:

    import sqlite3
    import pickle
    import sys
    import numpy as np
    import getpass
```

```

sys.path.insert(1, '{path_file}/linked_list')
sys.path.insert(2, '{path_file}/gamal')

from c import linked
from c import rotate_list_inv
from g import main1

conn = sqlite3.connect('user_encrypted_data.db')
cursor1 = conn.execute("select ID, NODO from DATA")

for fila in cursor1:
    print("User: ", f'{fila[0]}', "    Data: ", fila[1])

conn.close()

usr1 = input("Usuario a usar: ")
slec = input("Datos a Desbloquear: ")
pasw1 = input("Contraseña a Desbloquear: ")

with open(f'{path_file}/cache/{slec}.data', 'rb') as data:
    matrix = pickle.load(data)

with open(f'{path_file}/cache/{slec}.node', 'rb') as node:
    linked = pickle.load(node)

linked2 = rotate_list_inv(linked, conf)
print('[Mensaje a Desencriptar]: ')

print('-----')

print(linked2)

print('-----')

matrix_b = np.array([], dtype = int)
current = linked2
while current != None:
    wa = main1(current.data[0], current.data[1], current.data[2],
current.data[3])
    a1 = ''

```

```

a2 = ''
for id in range(len(wa)):
    if wa[id] == ' ':
        a1 = int(wa[:id])
        a2 = int(wa[id+1:])
    matrix_b = np.append(matrix_b, matrix[a1][a2])
wa = ''
current = current.next
power1 = len(usr1) ** len(matrix_b)

def final_msg(a,b):
    final_msg = ''
    for i in a:
        ay = round( i / power1)
        if ay <= 9:
            final_msg += ''.join(str(ay))
        else:
            final_msg += ''.join(chr(ay))
    if final_msg[:len(b)] == b:
        print("[Mensaje]: ", final_msg[len(b)+1:])
    else:
        print("Contraseña Incorrecta!")

for i in pasw1:
    if i == '0':
        pasw1 = pasw1.replace('0', str(len(usr1)))
final_msg(matrix_b, pasw1)
continuel = False

```