SGSSS-23.Laboratorio 5.Informe (Markel Ramiro)

Actividad 0.

Sólo si no se completó en la entrega/laboratorio anterior

1. Realizar un programa que obtenga el resumen SHA-256 de un fichero de texto

Presentar las partes más significativas del código y pruebas de su correcto funcionamiento.

2. Realizar un programa que, tomando como entrada un fichero de texto, obtenga como salida otro fichero con los mismos contenidos que el de entrada más una línea adicional

Presentar las partes más significativas del código y pruebas de su correcto funcionamiento.

Lenguaje de programación utilizado:

Actividad 1.

*[Si se ha llegado a conseguir un programa funcional, presentar las partes más significativas del código y pruebas de su correcto funcionamiento. **Incluir el resumen SHA-256 del fichero de salida**.*

```
import hashlib
import random

def generar_linea_adicional(identificador_estudiante):
    """Genera una línea adicional según las instrucciones"""
    secuencia_hexadecimal = ''.join(random.choices('abcdef', k=8))
    return f"{secuencia_hexadecimal}\t{identificador_estudiante}\t100"

def cumple_requisito_SHA256(hash_val):
    """Verifica si el hash SHA-256 comienza con el carácter "0""""
    return hash_val[0] == "0"

def get_sha256_of_file(file_path):
    hash_obj = hashlib.sha256()
    with open(file_path, 'rb') as f:
        while True:
            data = f.read(65536)  # leer en bloques de 64k
            if not data:
                break
            hash_obj.update(data)
    return hash_obj.hexdigest()
```

```
def main():
    identificador_estudiante = input("Introduce el identificador público del estudiante (dos caracteres hexadecimales en minúscula): ")

    with open('SGSSI-23.CB.02.txt', 'r') as f:
        contenido_original = f.read()

    archivo_modificado_path = 'SGSSI-23.CB.02_modificado.txt'

    while True:
        # Generar la línea adicional y añadirla al contenido modificado
        linea_adicional = generar_linea_adicional(identificador_estudiante)
         contenido_modificado = contenido_original + '\n' + linea_adicional # Añado el '\n' antes de la línea adicional para separarla del contenido original

        with open(archivo_modificado_path, 'w') as f:
            f.write(contenido_modificado)

        # Calcula el hash SHA-256 del archivo recién escrito
        hash_del_archivo = get_sha256_of_file(archivo_modificado_path)

        # Verificar si cumple el requisito
        if cumple_requisito_SHA256(hash_del_archivo):
            break

                    print(f"Archivo modificado guardado como 'SGSSI-23.CB.02_modificado.txt'. SHA-256: {hash_del_archivo}")

if __name__ == "__main__":
    main()


    main()
```

“código de ChatGPT”

- **Importaciónes:** Las importaciones permiten usar funciones de hash y generar valores aleatorios
- **Función generar_linea_adicional:** Esta función genera una línea con una secuencia hexadecimal aleatoria, seguida del identificador del estudiante y el número "100".
- **Función cumple_requisito_SHA256:** Esta función verifica si el hash SHA-256 proporcionado comienza con el carácter hexadecimal "0".
- **Función get_sha256_of_file:** Esta función calcula el hash SHA-256 de un archivo.
- **Función main:** Aquí es donde ocurre la lógica principal del programa. Se le pide al usuario un identificador de estudiante. Luego, se lee el contenido de un archivo SGSSI-23.CB.02.txt. Se genera y añade una línea adicional al contenido original del archivo. Después, se escribe el contenido modificado en un nuevo archivo SGSSI-23.CB.02_modificado.txt. A continuación, se calcula el hash SHA-256 del archivo modificado. Si el hash resultante comienza con "0", el proceso finaliza; de lo contrario, se repite el proceso con una nueva línea adicional hasta que se cumpla el requisito. El propósito de este bucle es encontrar una línea adicional que, cuando se añada al archivo, haga que su hash SHA-256 comience con "0".

```
Windows PowerShell
PS C:\Users\gerni\OneDrive\Escritorio\IA\4. Curso\SGSSI\Lab5> python .\codigo.py
Introduce el identificador público del estudiante (dos caracteres hexadecimales en minúscula): 7b
Archivo modificado guardado como 'SGSSI-23.CB.02_modificado.txt'. SHA-256: 0673c55ff92cbfb5e2224d09
2ea36599962655070a162e46eb952e0666366f42
```

SHA-256: 0673c55ff92cbfb5e2224d092ea36599962655070a162e46eb952e0666366f42

Actividad 2

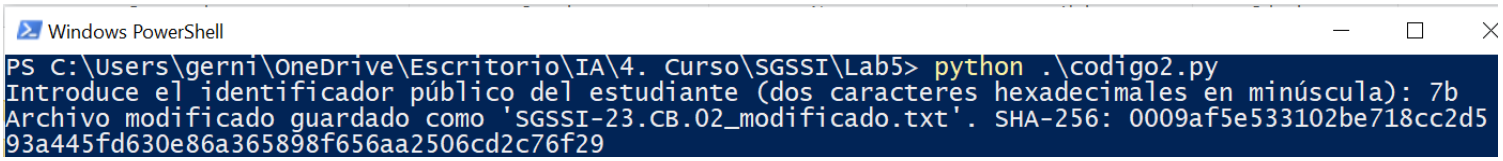
*[Si se ha llegado a conseguir un programa funcional, presentar las partes más significativas del código y pruebas de su correcto funcionamiento. **Incluir el resumen SHA-256 del fichero de salida**.*

```python
import hashlib
import random
import time

def generar_linea_adicional(identificador_estudiante):
    secuencia_hexadecimal = ''.join(random.choices('abcdef', k=8))
    return f"{secuencia_hexadecimal}\t{identificador_estudiante}\t100"

def get_sha256_of_file(file_path):
    hash_obj = hashlib.sha256()
    with open(file_path, 'rb') as f:
        while True:
            data = f.read(65536)  # leer en bloques de 64k
            if not data:
                break
            hash_obj.update(data)
    return hash_obj.hexdigest()
```

```
def cantidad_ceros_al_inicio(s):
    """Retorna la cantidad de ceros al inicio de una cadena"""
    return len(s) - len(s.lstrip('0'))

def main():
        identificador_estudiante = input("Introduce el identificador público del
estudiante (dos caracteres hexadecimales en minúscula): ")

    with open('SGSSI-23.CB.02.txt', 'r') as f:
        contenido_original = f.read()

    archivo_modificado_path = 'SGSSI-23.CB.02_modificado.txt'
    mejor_cantidad_de_ceros = 0
    mejor_hash = ''
      fin_tiempo = time.time() + 60  # Establecer un límite de 1 minuto para la
ejecución

    while time.time() < fin_tiempo:
        # Generar la línea adicional y añadirla al contenido modificado
        linea_adicional = generar_linea_adicional(identificador_estudiante)
         contenido_modificado = contenido_original + '\n' + linea_adicional  # Añado
el '\n' antes de la línea adicional

        with open(archivo_modificado_path, 'w') as f:
            f.write(contenido_modificado)

        # Calcula el hash SHA-256 del archivo recién escrito
        hash_del_archivo = get_sha256_of_file(archivo_modificado_path)
        ceros_actuales = cantidad_ceros_al_inicio(hash_del_archivo)

            # Si el hash actual tiene más ceros al principio que el mejor hash
encontrado hasta ahora, actualizar el mejor hash
        if ceros_actuales > mejor_cantidad_de_ceros:
            mejor_cantidad_de_ceros = ceros_actuales
            mejor_hash = hash_del_archivo

        print(f"Archivo modificado guardado como 'SGSSI-23.CB.02_modificado.txt'.
SHA-256: {mejor_hash}")

if __name__ == "__main__":
    main()
```

"código ChatGPT"

- **Importaciónes:** Las importaciones permiten usar funciones de hash y generar valores aleatorios
- **Función generar_linea_adicional y get_sha256_of_file:** Estas funciones son similares a las del código anterior. La primera genera una línea aleatoria y la segunda calcula el hash SHA-256 de un archivo.
- **Función cantidad_ceros_al_inicio:** Esta función devuelve la cantidad de ceros al inicio de una cadena
- **Función main**: La lógica principal ha cambiado con respecto al código anterior. En lugar de simplemente buscar un hash que comienza con "0", ahora se busca el hash que tiene la mayor cantidad de ceros al comienzo dentro de un límite de tiempo de 1 minuto.
    - Se establece un límite de tiempo de 1 minuto para la ejecución.
    - En cada iteración, se genera una línea adicional y se añade al contenido original. Luego, se calcula el hash SHA-256 del archivo modificado.
    - Se verifica la cantidad de ceros al principio del hash. Si es mayor que el "mejor" hash encontrado hasta ahora, se actualiza el mejor hash.
    - Finalmente, después de que se agota el tiempo o se encuentra el hash óptimo, se muestra el mejor hash encontrado.



SHA-256: 0009af5e533102be718cc2d593a445fd630e86a365898f656aa2506cd2c76f29

Actividad 3 (opcional) Realizar un programa que, tomando como entrada dos ficheros de texto…

[Si se ha llegado a conseguir un programa funcional, presentar las partes más significativas del código y pruebas de su correcto funcionamiento.
Si no se ha llegado a conseguir un programa funcional, explicar lo conseguido durante el laboratorio.]

Actividad 4. Opcional.

Enlace al código: