INSTITUTO POLITÉCNICO NACIONAL

Escuela Superior de Cómputo

Laboratory Session #7 “Hash function and key generation for RSA”

ALUMNOS:

- González Barrientos Geovanni Daniel
- Maldonado Flores Marco de Jesús

GRUPO: 3CM14

PROF: Diaz Santiago Sandra

MATERIA: Criptografía

FECHA: 22 de Diciembre del 2022

DESARROLLO

Para esta actividad se requirió la creación de un programa capaz de utilizar cualquiera de las funciones hash disponibles en bibliotecas criptográficas, con el objetivo de obtener el digesto de distintos tipos de ficheros según se indican en las especificaciones. De igual forma, con ayuda de dichas bibliotecas criptográficas, se solicitó la generación de un par de llaves necesarias para RSA, que posteriormente deberán ser almacenados en ficheros por separado.

Para este programa se utilizó el lenguaje de programación Python, además de la biblioteca criptográfica “pycryptodome”. Para ejecutar este programa, es necesario ejecutar el archivo “main.py” desde la terminal y en el mismo directorio donde se encuentran los distintos ficheros necesarios para las pruebas con la función hash.

El proyecto está conformado de los siguientes archivos:

- Main.py: Código fuente del programa que realiza todas las operaciones necesarias para generar los hash y el par de llaves.
- Original_text.txt : Archivo de texto plano de 1mb que servirá para realizar las pruebas hash.
- Understanding-cryptography.pdf: Archivo grande en formato pdf que contiene texto e imágenes, cuyo objetivo es verificar el comportamiento de la función hash.
- Img.jpg: Archivo de imagen que en su interior contiene un mapa de bits, el cual será procesado por el hash.
- Song.mp3: Archivo de audio en formato mp3 que se utilizará para generar el digesto con la función hash

Una vez listando el código a utilizar y todos los ficheros que se utilizaran en las pruebas, se procederá a explicar la estructura del programa.

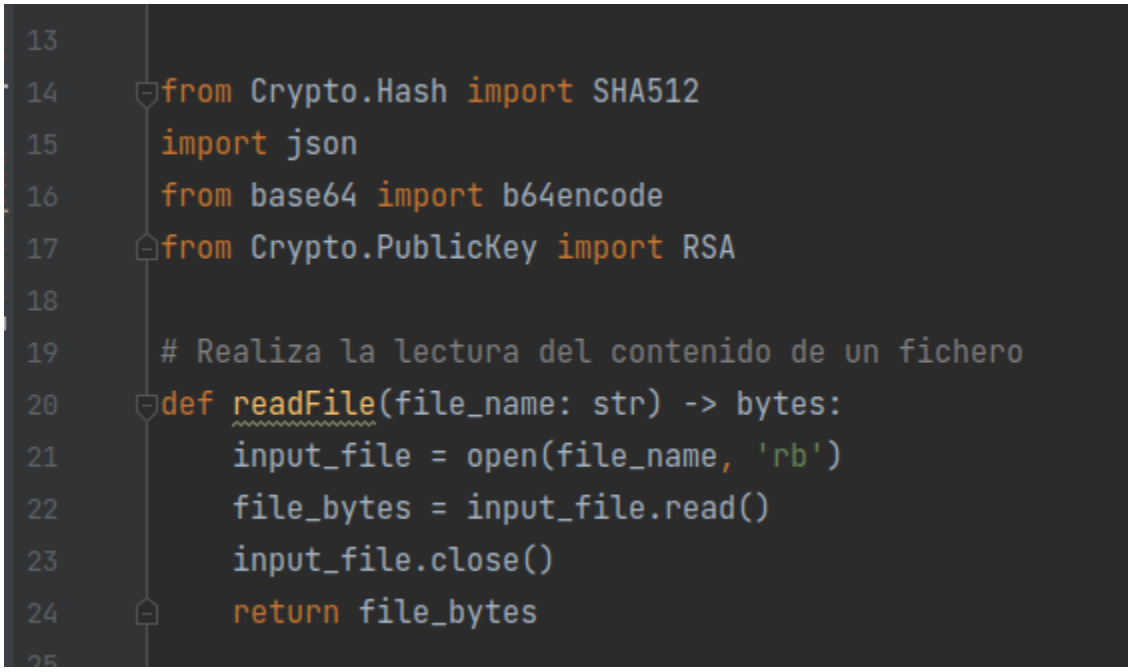
Estructura de main.py

Este programa consta de 4 funciones especiales para realizar las operaciones con el hash y RSA, además del procedimiento principal del programa.

Funcion readFile()

Como su nombre lo dice, esta función es la encargada de realizar la lectura de ficheros, para recuperar y retornar el contenido de estos en cadenas de bytes. Esta función, además de las bibliotecas utilizadas se muestran en la siguiente imagen.

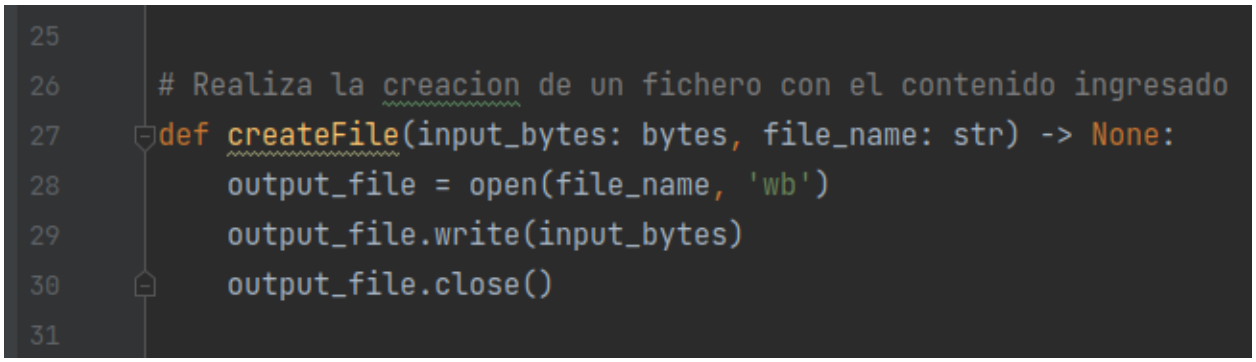
.



```
13
14  from Crypto.Hash import SHA512
15  import json
16  from base64 import b64encode
17  from Crypto.PublicKey import RSA
18
19  # Realiza la lectura del contenido de un fichero
20  def readFile(file_name: str) -> bytes:
21      input_file = open(file_name, 'rb')
22      file_bytes = input_file.read()
23      input_file.close()
24      return file_bytes
```

Funcion createFile()

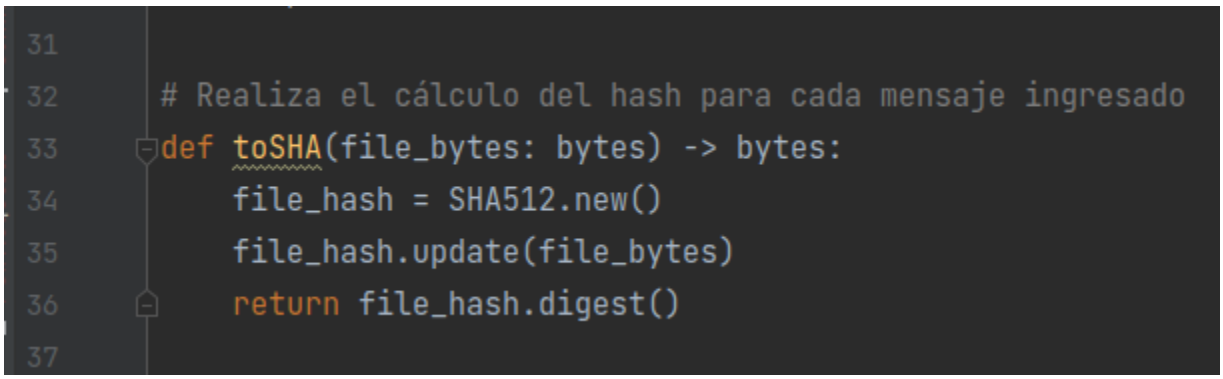
Siendo muy útil para mostrar los resultados de realizar la función hash y la creación de las llaves para RSA, esta función se encarga de almacenar los datos en un fichero. La estructura de esta función se muestra a continuación:



```
25
26  # Realiza la creacion de un fichero con el contenido ingresado
27  def createFile(input_bytes: bytes, file_name: str) -> None:
28      output_file = open(file_name, 'wb')
29      output_file.write(input_bytes)
30      output_file.close()
31
```

Funcion toSHA()

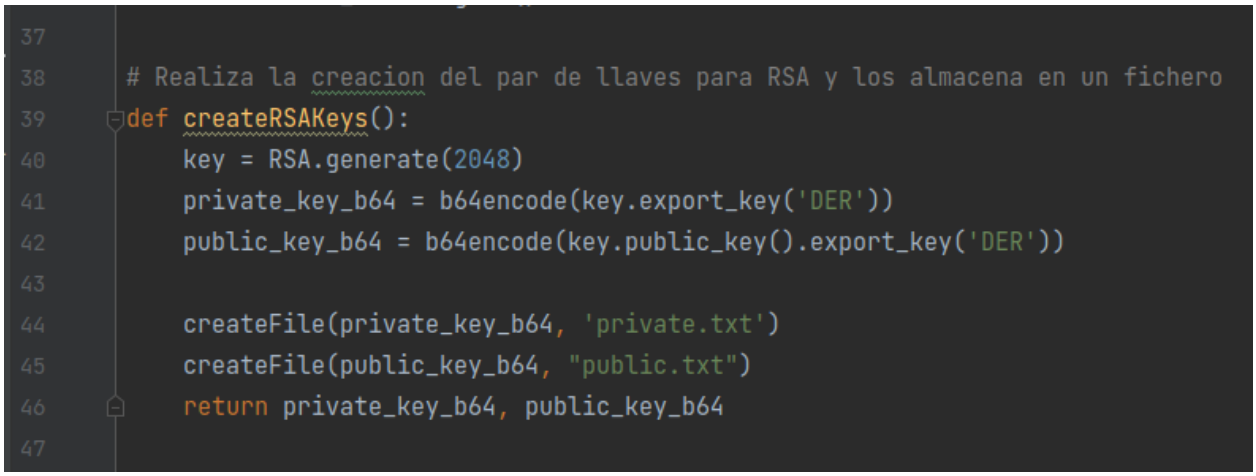
Siendo una de las funciones más importantes de este programa, en esta parte se realiza la creación de una instancia de SHA versión 2 en su formato SHA512. Una vez creada esa instancia, se ingresa el mensaje para generar el hash y retornarlo en un formato imprimible, tal como se aprecia en la siguiente imagen:



```
31
32  # Realiza el cálculo del hash para cada mensaje ingresado
33  def toSHA(file_bytes: bytes) -> bytes:
34      file_hash = SHA512.new()
35      file_hash.update(file_bytes)
36      return file_hash.digest()
37
```

Funcion createRSAKeys()

Para finalizar con las funciones especiales, se tiene al que es el encargado de generar las llaves para RSA con un nivel de seguridad de 2048 bits, además de ver como ambas llaves son codificadas en base64 y son guardadas cada uno en su propio fichero por medio de la función createFile. Esta parte del código que contiene la función descrita en este párrafo se muestra en la siguiente figura:



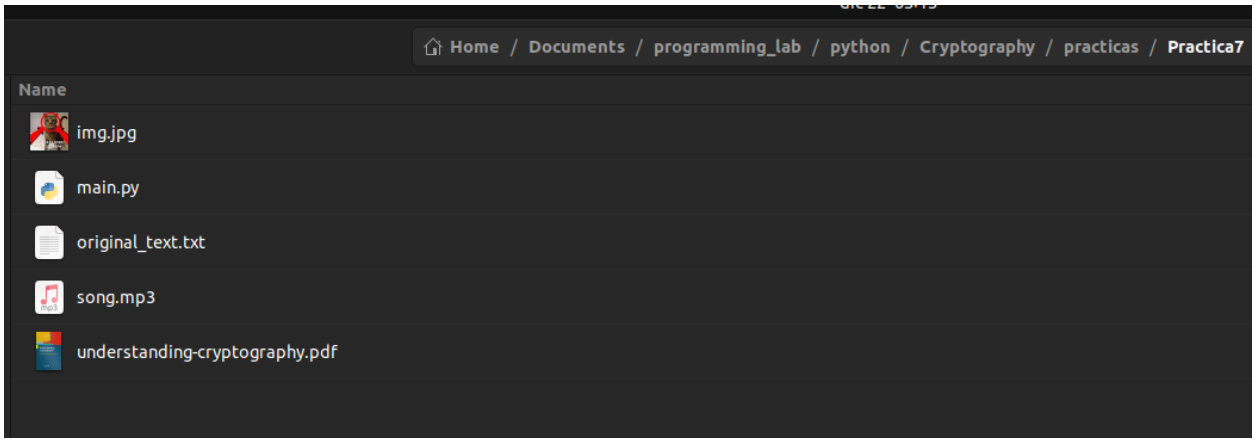
Funcionalidad principal del programa

Por último, se tienen las instrucciones que guían el comportamiento general del programa, donde se destacan las llamadas a las funciones para generar los hash, la creación de un archivo JSON para almacenar los digestos obtenidos que serán posteriormente guardados en un fichero 'output.txt', además de la creación de las llaves para RSA.

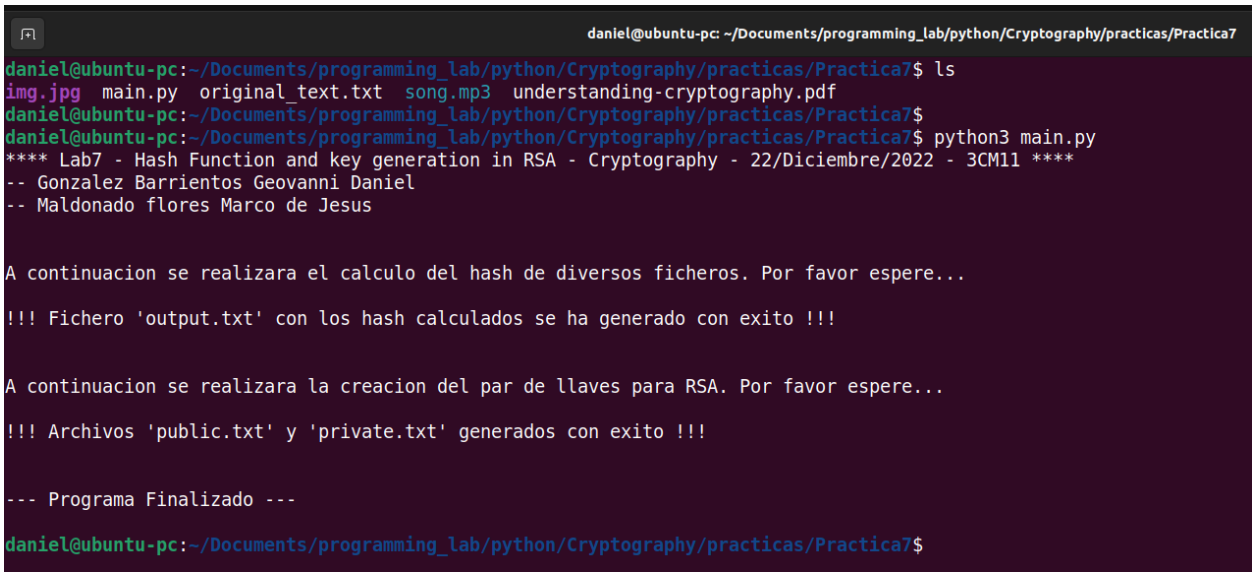
```
47
48  # Función principal
49  print("**** Lab7 - Hash Function and key generation in RSA - Cryptography - 22/Diciembre/2022 - 3CM11 ****")
50  print("-- Gonzalez Barrientos Geovanni Daniel\n-- Maldonado flores Marco de Jesus\n")
51  print("\nA continuacion se realizara el calculo del hash de diversos ficheros. Por favor espere... \n")
52
53  # Cadena de texto plano
54  plain_password = "Password"
55
56  # Calcula el hash de diversos ficheros
57  sha_password = b64encode(toSHA(plain_password.encode('utf-8'))) # Cadena de 8 caracteres
58  sha_text = b64encode(toSHA(readFile("original_text.txt"))) # Archivo de texto plano
59  sha_pdf = b64encode(toSHA(readFile("understanding-cryptography.pdf"))) # Archivo pdf
60  sha_audioFile = b64encode(toSHA(readFile("song.mp3"))) # Archivo de audio
61  sha_img = b64encode(toSHA(readFile("img.jpg"))) # Imagen
62
63  # Genera un fichero con los resultados de calcular el hash a cada archivo
64  output = json.dumps({'sha_password': sha_password.decode('ascii'),
65                       'sha_text': sha_text.decode('ascii'),
66                       'sha_pdf': sha_pdf.decode('ascii'),
67                       'sha_audioFile': sha_audioFile.decode('ascii'),
68                       'sha_img': sha_img.decode('ascii')})
69
70  createFile(output.encode('ascii'), "output.txt")
71  print("!!! Fichero 'output.txt' con los hash calculados se ha generado con exito !!!\n")
72  print("\nA continuacion se realizara la creacion del par de llaves para RSA. Por favor espere... \n")
73
74  # Genera el par de llaves para RSA
75  private, public = createRSAKeys()
76  print("!!! Archivos 'public.txt' y 'private.txt' generados con exito !!!\n")
77  print("\n--- Programa Finalizado ---\n")
```

Pruebas de funcionamiento

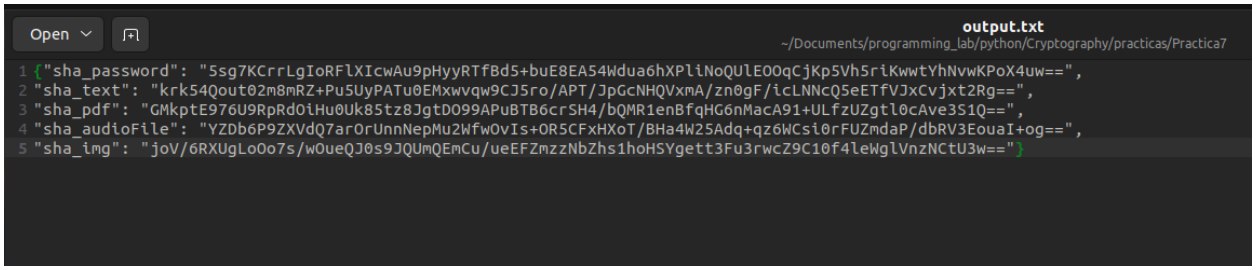
A continuación, se mostraran imágenes de las pruebas de funcionamiento del programa.



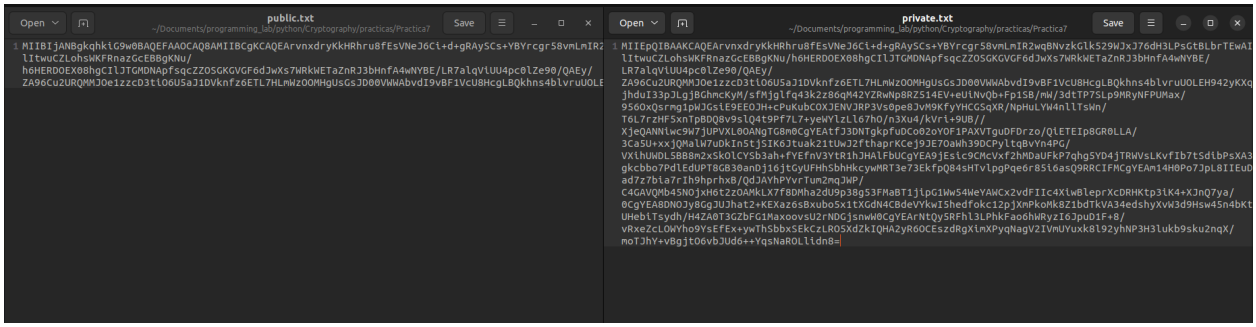
Archivos del proyecto



Ejecucion de main.py



Hash obtenidos de los distintos ficheros de prueba



Llaves pública y privada generadas para RSA