



**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	3
Estructura de main.py	3
Funcion readFile()	4
Funcion createFile()	4
Funcion toSHA()	5
Funcion createRSAKeys()	5
Funcionalidad principal del programa	6
Pruebas de funcionamiento	7

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
25
```

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:

```
37
38     # Realiza la creacion del par de llaves para RSA y los almacena en un fichero
39     def createRSAKeys():
40         key = RSA.generate(2048)
41         private_key_b64 = b64encode(key.export_key('DER'))
42         public_key_b64 = b64encode(key.public_key().export_key('DER'))
43
44         createFile(private_key_b64, 'private.txt')
45         createFile(public_key_b64, "public.txt")
46     return private_key_b64, public_key_b64
47
```

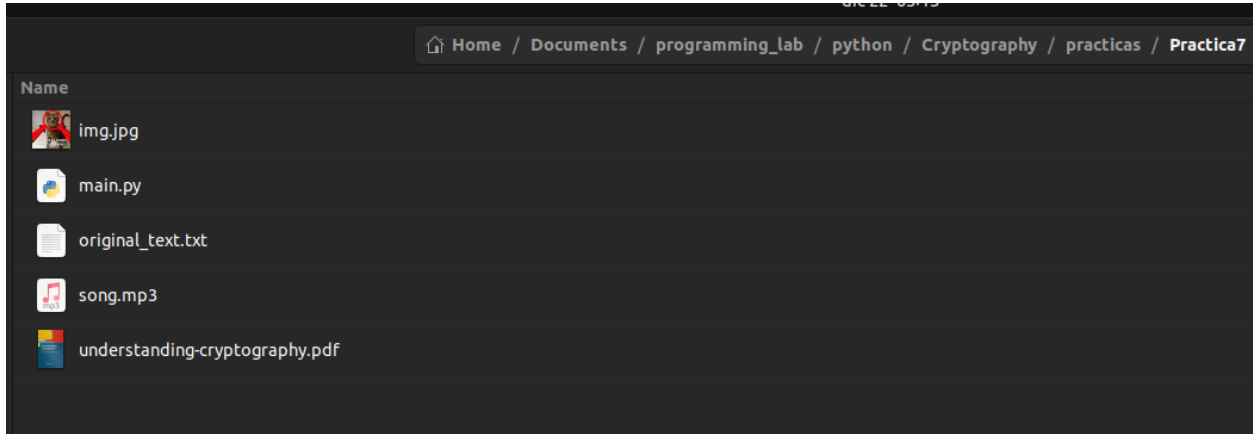
Funcionalidad principal del programa

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

```
daniel@ubuntu-pc: ~/Documents/programming_lab/python/Cryptography/practicas/Practica7
daniel@ubuntu-pc:~/Documents/programming_lab/python/Cryptography/practicas/Practica7$ ls
img.jpg main.py original_text.txt song.mp3 understanding-cryptography.pdf
daniel@ubuntu-pc:~/Documents/programming_lab/python/Cryptography/practicas/Practica7$
daniel@ubuntu-pc:~/Documents/programming_lab/python/Cryptography/practicas/Practica7$ python3 main.py
**** Lab7 - Hash Function and key generation in RSA - Cryptography - 22/Diciembre/2022 - 3CM11 ****
-- Gonzalez Barrientos Geovanni Daniel
-- Maldonado flores Marco de Jesus

A continuacion se realizara el calculo del hash de diversos ficheros. Por favor espere...

!!! Fichero 'output.txt' con los hash calculados se ha generado con exito !!!

A continuacion se realizara la creacion del par de llaves para RSA. Por favor espere...

!!! Archivos 'public.txt' y 'private.txt' generados con exito !!!

--- Programa Finalizado ---
daniel@ubuntu-pc:~/Documents/programming_lab/python/Cryptography/practicas/Practica7$
```

Ejecucion de main.py


```
output.txt
~/Documents/programming_lab/python/Cryptography/practicas/Practica7

1 ("sha_password": "5sg7KCrRLgIoRfLXIcWAu9pHYyRTfBd5+buE8EA54Wdua6HXPlNoQULE00qCjKp5Vh5rIkwwtYhNvwKPoX4uw==",
2 "sha_text": "krk54Qout02m8mRZ+Pu5UyPATu0EMxwvqw9CJ5ro/APT/JpGcNHQVxmA/zn0gF/icLNNcQ5eETfVJxCvjxt2Rg==",
3 "sha_pdf": "GMkptE976U9RpRd0iHu0Uk85tz8JgtD099APuBTB6crSH4/bQMR1enBfQHG6nMacA91+ULfZUZgtL0cAve3S1Q==",
4 "sha_audioFile": "YZDb6P9ZXVdQ7arOrUnnNepMu2Wfw0vIs+OR5CFXHXoT/BHa4W25Adq+qz6WCs10rFUZmdaP/dbRV3EouaI+og==",
5 "sha_img": "joV/6RXUgLo0o7s/w0ueQJ0s9JQUmQEmCu/ueEFZmzzNbZhs1hoHSYgett3Fu3rwcZ9C10f4LeWgLVnzNctU3w=="
```

Hash obtenidos de los distintos ficheros de prueba

```
public.txt
~/Documents/programming_lab/python/Cryptography/practicas/Practica7

MIIBTjANBgkqhkiG9w0BAQFAAQCAQ8AMIIBGKCAQEAfvnxdrykHhruBFESvNeJ6Cl+d+gRAySCs+YBYrcgr5BvnLnR2
l1twcZLohsMKfRnazCcEBgKku/h0HERD0EX08hgC1lJTG0NDapfsqCZ05GKGVGf6d3wxs7WRkMETaZnR33bHnFA4wMYBE/LR7a1qVLUU4pc0LZe90/QAEY/
ZA96Cu2URQMHJ0e1zzcd3tL06UsaJ1DVknfz6ETL7HLmhz00MHgUsGsJ000VMHAbvdI9VBFIvCU8HcgLBQkhns4blvrUOLE
jhdU133pLgJ8chmckYw/5fMjglFq43kZz86qM42YzRwnp8Z5I4EV+eULvQb+fp1S8/mm/3dTP7SLp9RyNFPUMax/
9560Xdsrapi9HJG5LESEEDJH+FuKubCDX3ENJRP3Vs0pe83Vn9KfyhHCsQXR/hphULY4nllTSn/
T6L7rZHF5xntP8DQ8v9s1Q4t9PF7L7+yeWYlZL167h0/n3Xu4/kvrl+9UB//
XjeQANNlwc9W7JUPVXL00AngTGM0cgyEATfJ3DNTGkpFuDCo02ov0F1PAKVTgudFDfz0/QLETEIp8GR0LLA/
3Ca5U+xxJQhAlW7u0kIn5tj5IK6Jtuak21tUw22ftthaprKceJ93ET0aWh39DCPyLlqBvYn4PG/
VXLH0DL58Bbn2sK0LCY5b3ah+fyEfnv3YER1h3HA1FbUcgyEA0JesLc9Chcvrf2HmduFkP7ahg5V04JTMWslkvfIb7t5dlbpxAa
gkcb07Pd1EdUPt8G3oandj16jtcyUHH5bhkkywNRt3e73EkfpQ84sHTVlpgfqe8B51easQ9RRCIFHCgyEAm4H0P07Jpl81IEu0
adz7b1a7r1h9hprxb/JQdJAYhPyvrtun2nqJWP/
C4GAVQMB45NOjyhStz20ANKLX7f8Dhaz2U9p38g53FhABT1jlpG1Ww54MeYAMCz2vdfI1c4XlWbLeprXCDRHKtp3LK4+XJnQ7ya/
0CpyEAB0NDJy8Gp3UJ2aTz+KEXaz0Bxb05x1XGdn4C8dYKw1Shedf0kcl2gJvmpk0R8Z1b0tVVA34edshyAVw3d9Hsw45n4bkt
UheblTsydh/H42ABT3GZBFClMxaoovU2rNDGj5nwMCGyEArNT0ySRFh13LpNFa08h8yz16Jpu01F+a/
vRxeZcL0Wyo9ysEFEx+YwThSbbsxEKczLR05XdZkIQHA2y860CESzdrGkLmXPYqNagV2lVnUvuxk8192yHNP3H3lukb9sku2nqX/
moTJhy+vBgjT06vbJ0d6++YqsNaROLlIdn8=

private.txt
~/Documents/programming_lab/python/Cryptography/practicas/Practica7

MIIEpQIBAAKCAQEAfvnxdrykHhruBFESvNeJ6Cl+d+gRAySCs+YBYrcgr5BvnLnR2wq8NvZkClK529H3xJ76dH3LPs0tBLBrTEwA1
l1twcZLohsMKfRnazCcEBgKku/h0HERD0EX08hgC1lJTG0NDapfsqCZ05GKGVGf6d3wxs7WRkMETaZnR33bHnFA4wMYBE/
LR7a1qVLUU4pc0LZe90/QAEY/
ZA96Cu2URQMHJ0e1zzcd3tL06UsaJ1DVknfz6ETL7HLmhz00MHgUsGsJ000VMHAbvdI9VBFIvCU8HcgLBQkhns4blvrUOLEH942yKXQ
jhdU133pLgJ8chmckYw/5fMjglFq43kZz86qM42YzRwnp8Z5I4EV+eULvQb+fp1S8/mm/3dTP7SLp9RyNFPUMax/
9560Xdsrapi9HJG5LESEEDJH+FuKubCDX3ENJRP3Vs0pe83Vn9KfyhHCsQXR/hphULY4nllTSn/
T6L7rZHF5xntP8DQ8v9s1Q4t9PF7L7+yeWYlZL167h0/n3Xu4/kvrl+9UB//
XjeQANNlwc9W7JUPVXL00AngTGM0cgyEATfJ3DNTGkpFuDCo02ov0F1PAKVTgudFDfz0/QLETEIp8GR0LLA/
3Ca5U+xxJQhAlW7u0kIn5tj5IK6Jtuak21tUw22ftthaprKceJ93ET0aWh39DCPyLlqBvYn4PG/
VXLH0DL58Bbn2sK0LCY5b3ah+fyEfnv3YER1h3HA1FbUcgyEA0JesLc9Chcvrf2HmduFkP7ahg5V04JTMWslkvfIb7t5dlbpxAa
gkcb07Pd1EdUPt8G3oandj16jtcyUHH5bhkkywNRt3e73EkfpQ84sHTVlpgfqe8B51easQ9RRCIFHCgyEAm4H0P07Jpl81IEu0
adz7b1a7r1h9hprxb/JQdJAYhPyvrtun2nqJWP/
C4GAVQMB45NOjyhStz20ANKLX7f8Dhaz2U9p38g53FhABT1jlpG1Ww54MeYAMCz2vdfI1c4XlWbLeprXCDRHKtp3LK4+XJnQ7ya/
0CpyEAB0NDJy8Gp3UJ2aTz+KEXaz0Bxb05x1XGdn4C8dYKw1Shedf0kcl2gJvmpk0R8Z1b0tVVA34edshyAVw3d9Hsw45n4bkt
UheblTsydh/H42ABT3GZBFClMxaoovU2rNDGj5nwMCGyEArNT0ySRFh13LpNFa08h8yz16Jpu01F+a/
vRxeZcL0Wyo9ysEFEx+YwThSbbsxEKczLR05XdZkIQHA2y860CESzdrGkLmXPYqNagV2lVnUvuxk8192yHNP3H3lukb9sku2nqX/
moTJhy+vBgjT06vbJ0d6++YqsNaROLlIdn8=
```

Llaves pública y privada generadas para RSA