Instituto Politécnico Nacional

Escuela Superior de Cómputo

Cryptography

Hash Functions

Group: 3CM15

Date: November 24, 2021

Student.

Maximiliano Cazares Martínez

Professor.

Sandra Díaz Santiago

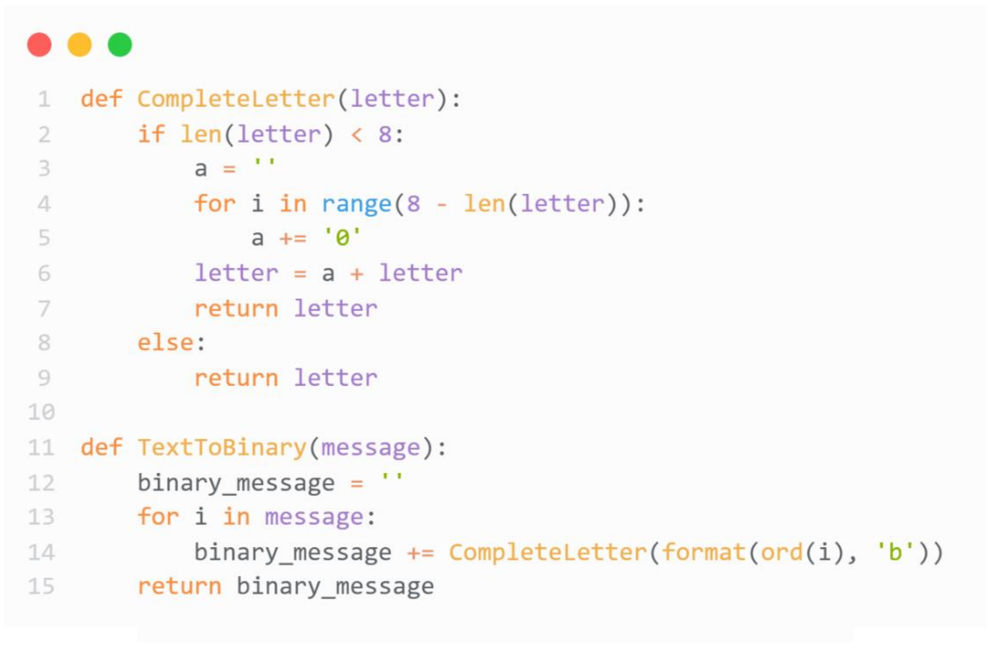
How to run the program.

The practice contains 3 python scripts called HashOwn.py, HashLibrary.py and HF_test.py. For running them, all we need is the library pycryptodome, we can install it typing "py -m pip install pycryptodome" in any computer with python 3 interpreter.

Organization.

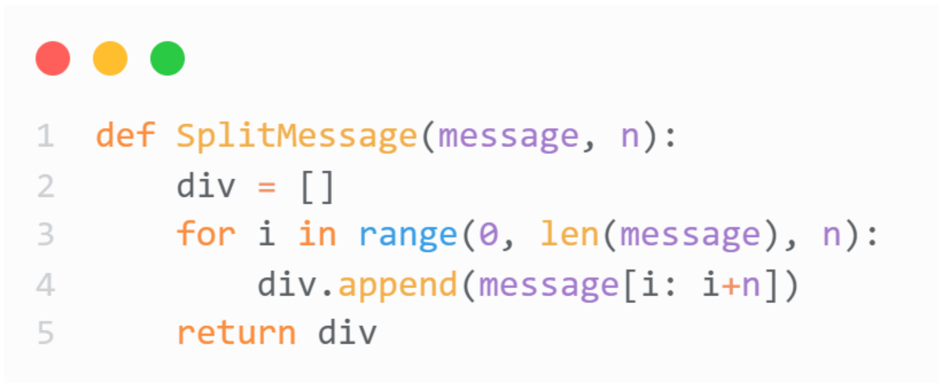
1) HashOwn.py is an own hash function based in an xor operation. This Hash functions compute a xor operation with n-size blocks from a text.

First, we turn each character from the message into a binary string of 8-bit size with the functions CompleteLetter and TextToBinary.



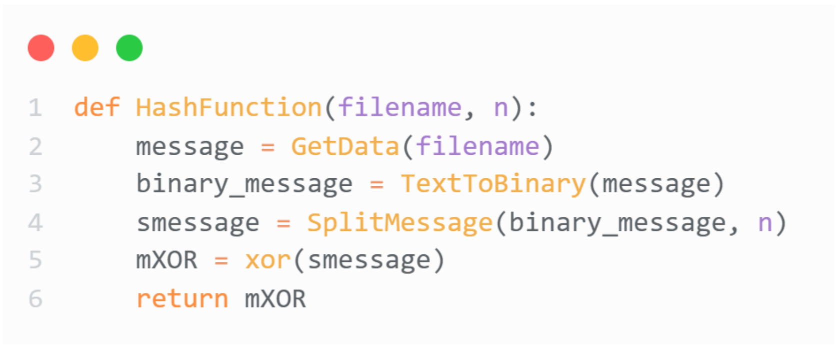
```python
def CompleteLetter(letter):
    if len(letter) < 8:
        a = ''
        for i in range(8 - len(letter)):
            a += '0'
        letter = a + letter
        return letter
    else:
        return letter

def TextToBinary(message):
    binary_message = ''
    for i in message:
        binary_message += CompleteLetter(format(ord(i), 'b'))
    return binary_message
```

Second, we split the text message in n-size block with the function SplitMessage which has two parameters, the message itself and the n-size block.



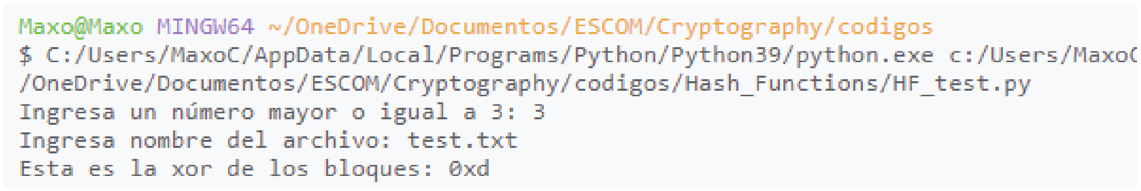
```python
def SplitMessage(message, n):
    div = []
    for i in range(0, len(message), n):
        div.append(message[i: i+n])
    return div
```

Finally, the function called HashFunction compute the xor operation with the n-size blocks.



```python
def HashFunction(filename, n):
    message = GetData(filename)
    binary_message = TextToBinary(message)
    smessage = SplitMessage(binary_message, n)
    mXOR = xor(smessage)
    return mXOR
```

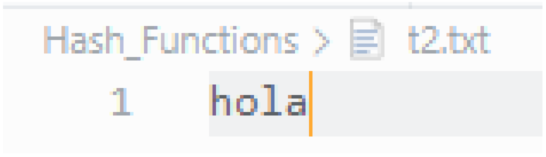
For running above exercise, we need to execute the function Exercise1 from HF_test.py script. Thus, we get the follows in the terminal.



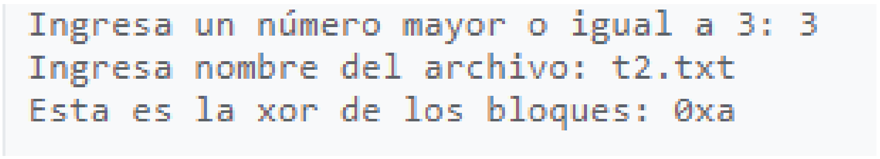
```
Maxo@Maxo MINGW64 ~/OneDrive/Documentos/ESCOM/Cryptography/codigos
$ C:/Users/MaxoC/AppData/Local/Programs/Python/Python39/python.exe c:/Users/MaxoC
/OneDrive/Documentos/ESCOM/Cryptography/codigos/Hash_Functions/HF_test.py
Ingresa un número mayor o igual a 3: 3
Ingresa nombre del archivo: test.txt
Esta es la xor de los bloques: 0xd
```

2) The previous hash functions have collisions when having the same n-size block we get a disorder message. In other words, if we have the messages “hola” and “olah” the xor operation is the same.

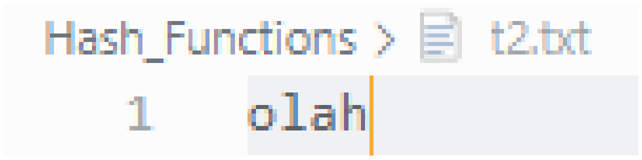
For this example, we will use the same file with different message.



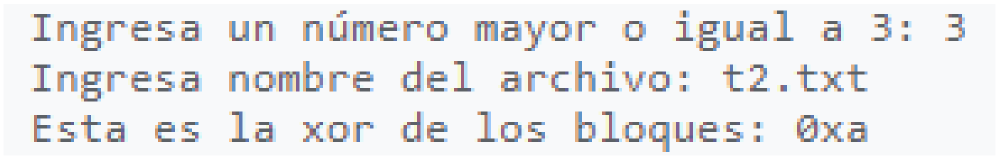
```
hola
```



```
Ingresa un número mayor o igual a 3: 3
Ingresa nombre del archivo: t2.txt
Esta es la xor de los bloques: 0xa
```


Hash_Functions > t2.txt

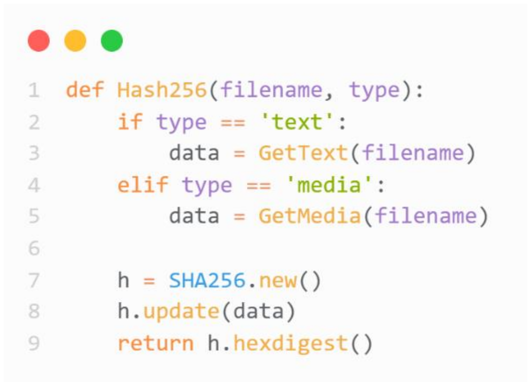
```
olah
```



```
Ingresa un número mayor o igual a 3: 3
Ingresa nombre del archivo: t2.txt
Esta es la xor de los bloques: 0xa
```

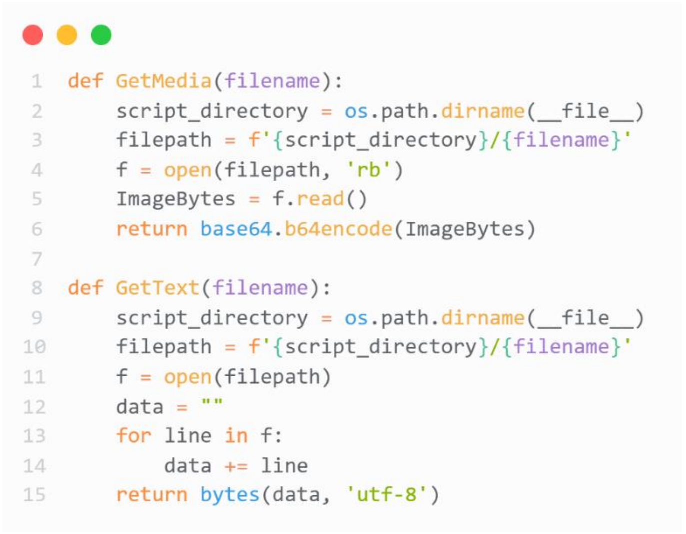
3) HashLibrary.py has an implementation of SHA-256 Hash Algorithm. For this implementation, we use for example a text file, an image and a video.

The function that pycryptodome library provide us is as easier as create a new object of SHA256 and give it a byte buffer with the data. Then, it returns us a hexadecimal digest string of 64-bytes length.



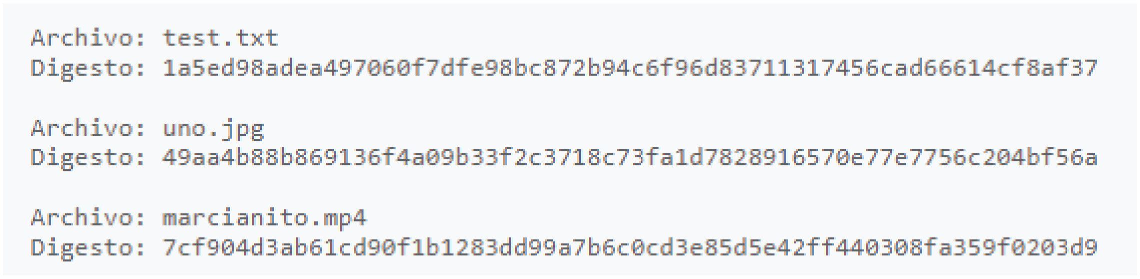
```python
def Hash256(filename, type):
    if type == 'text':
        data = GetText(filename)
    elif type == 'media':
        data = GetMedia(filename)

    h = SHA256.new()
    h.update(data)
    return h.hexdigest()
```

For running it, we need to specify which type of file we will choose. Two functions called GetMedia and GetText get the bytes from images, videos and text files respectively for being used in the hash function.



```python
def GetMedia(filename):
    script_directory = os.path.dirname(__file__)
    filepath = f'{script_directory}/{filename}'
    f = open(filepath, 'rb')
    ImageBytes = f.read()
    return base64.b64encode(ImageBytes)

def GetText(filename):
    script_directory = os.path.dirname(__file__)
    filepath = f'{script_directory}/{filename}'
    f = open(filepath)
    data = ""
    for line in f:
        data += line
    return bytes(data, 'utf-8')
```

How we have been saying, we proved this implementation with 3 files (text, image and video). To make it easier, we define the files info in HF_test.py, so, we got the follows as output in terminal.



```
Archivo: test.txt
Digesto: 1a5ed98adea497060f7dfe98bc872b94c6f96d83711317456cad66614cf8af37

Archivo: uno.jpg
Digesto: 49aa4b88b869136f4a09b33f2c3718c73fa1d7828916570e77e7756c204bf56a

Archivo: marcianito.mp4
Digesto: 7cf904d3ab61cd90f1b1283dd99a7b6c0cd3e85d5e42ff440308fa359f0203d9
```