ASSIGNMENT TITLE

SUBJECT NAME: Cryptography and Network Security

SUBJECT CODE: CS6008

MODULE: 08

NAME: R.Aadharsh

REG.NO.: 2019103604

DATE: 11/06/2022

AIM:

Computing MACs, Hashes and HMACs for messages

TOOLS INVOLVED:

VS Code

Python

PROBLEM DESCRIPTION:

HASHING

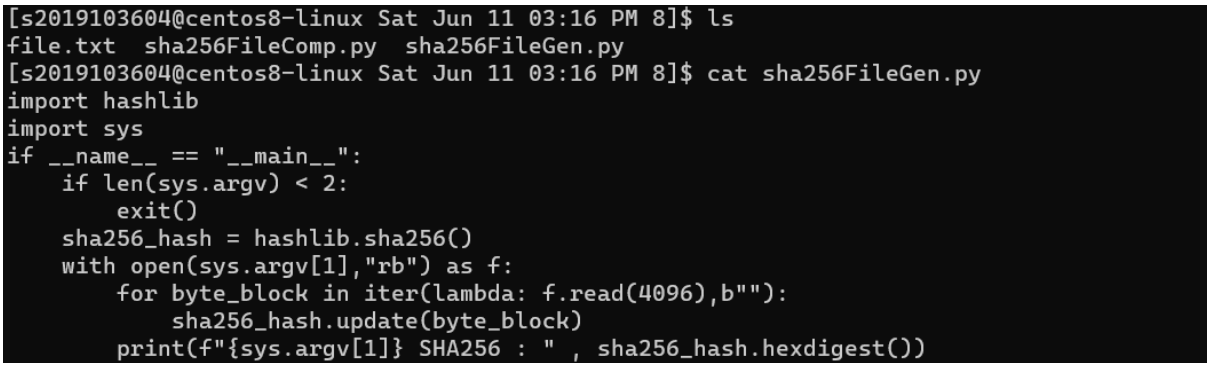
A cryptographic hash function is an algorithm that takes an arbitrary amount of data input a credential and produces a fixed-size output of enciphered text called a hash value, or just "hash" . Cryptographic hash algorithms produce irreversible and unique hashes. The larger the number of possible hashes, the smaller the chance that two values will create the same

hash.

SCREENSHOTS:

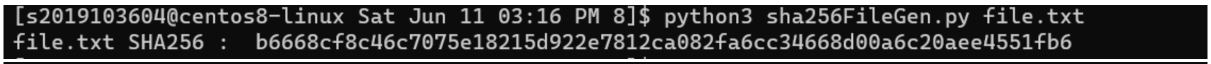
CODE:

SHA-256 generator for a given file

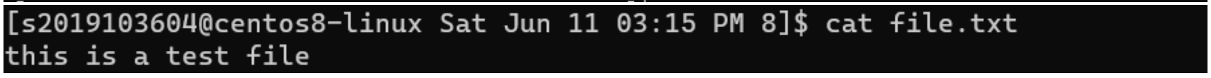


```
[s2019103604@centos8-linux Sat Jun 11 03:16 PM 8]$ ls
file.txt  sha256FileComp.py  sha256FileGen.py
[s2019103604@centos8-linux Sat Jun 11 03:16 PM 8]$ cat sha256FileGen.py
import hashlib
import sys
if __name__ == "__main__":
    if len(sys.argv) < 2:
        exit()
    sha256_hash = hashlib.sha256()
    with open(sys.argv[1],"rb") as f:
        for byte_block in iter(lambda: f.read(4096),b""):
            sha256_hash.update(byte_block)
        print(f"{sys.argv[1]} SHA256 : " , sha256_hash.hexdigest())
```

Compiling and generating hash code:



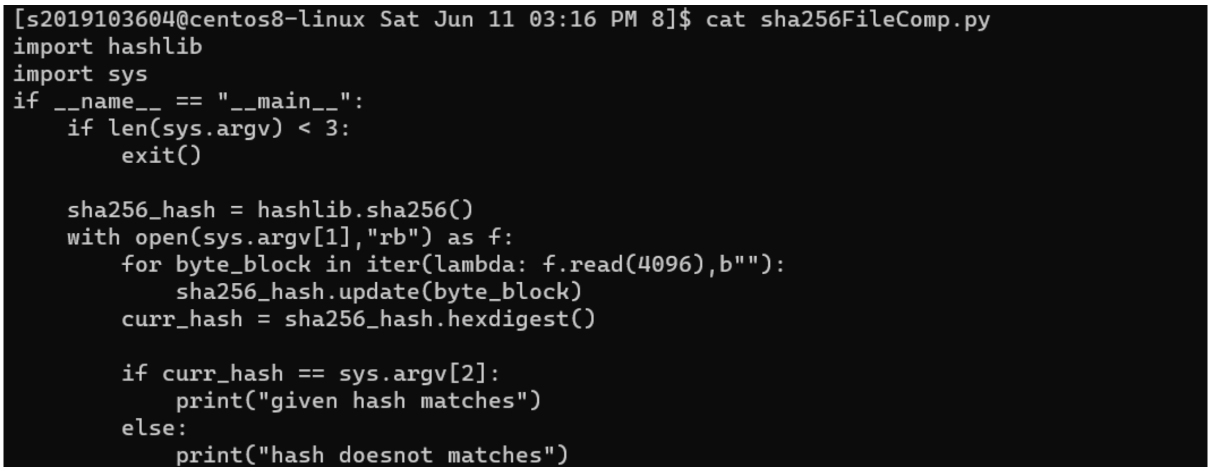
```
[s2019103604@centos8-linux Sat Jun 11 03:16 PM 8]$ python3 sha256FileGen.py file.txt
file.txt SHA256 :  b6668cf8c46c7075e18215d922e7812ca082fa6cc34668d00a6c20aee4551fb6
```



```
[s2019103604@centos8-linux Sat Jun 11 03:15 PM 8]$ cat file.txt
this is a test file
```

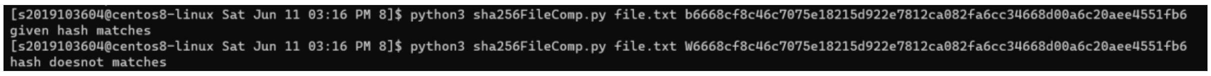
Comparing a given sha256 hash with a file

Code:



```
[s2019103604@centos8-linux Sat Jun 11 03:16 PM 8]$ cat sha256FileComp.py
import hashlib
import sys
if __name__ == "__main__":
    if len(sys.argv) < 3:
        exit()

    sha256_hash = hashlib.sha256()
    with open(sys.argv[1],"rb") as f:
        for byte_block in iter(lambda: f.read(4096),b""):
            sha256_hash.update(byte_block)
        curr_hash = sha256_hash.hexdigest()

        if curr_hash == sys.argv[2]:
            print("given hash matches")
        else:
            print("hash doesnot matches")
```

Compiling and comparing the hashes with the right hash value and the wrong one.



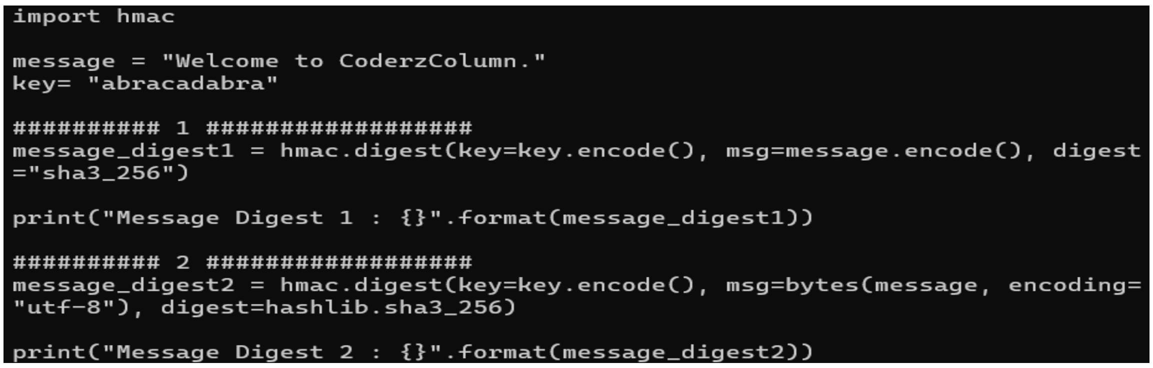
```
[s2019103604@centos8-linux Sat Jun 11 03:16 PM 8]$ python3 sha256FileComp.py file.txt b6668cf8c46c7075e18215d922e7812ca082fa6cc34668d00a6c20aee4551fb6
given hash matches
[s2019103604@centos8-linux Sat Jun 11 03:16 PM 8]$ python3 sha256FileComp.py file.txt W6668cf8c46c7075e18215d922e7812ca082fa6cc34668d00a6c20aee4551fb6
hash doesnot matches
```

HMAC –

Hash-based message authentication code

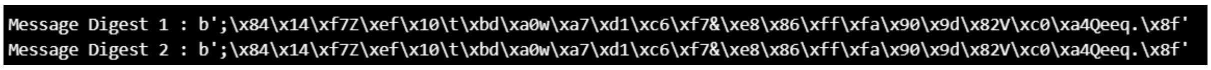
HMAC is an algorithm that generates a hash of the message using a cryptographic hash function and a secret cryptographic key. It can be used to check data for integrity and authenticity. It lets us calculate message authenticity and integrity using a shared key between two parties without the use of complex public key infrastructure involving certificates. Python provides us with the module name hmac which provides an implementation for this algorithm. It takes as input hashing algorithm name which is one of the algorithms which is available through hashlib library of Python.

Code:



```python
import hmac

message = "Welcome to CoderzColumn."
key= "abracadabra"

########## 1 ###################
message_digest1 = hmac.digest(key=key.encode(), msg=message.encode(), digest
="sha3_256")

print("Message Digest 1 : {}".format(message_digest1))

########## 2 ###################
message_digest2 = hmac.digest(key=key.encode(), msg=bytes(message, encoding=
"utf-8"), digest=hashlib.sha3_256)

print("Message Digest 2 : {}".format(message_digest2))
```

O/P:



```
Message Digest 1 : b';\x84\x14\xf7Z\xef\x10\t\xbd\xa0w\xa7\xd1\xc6\xf7&\xe8\x86\xff\xfa\x90\x9d\x82V\xc0\xa4Qeeq.\x8f'
Message Digest 2 : b';\x84\x14\xf7Z\xef\x10\t\xbd\xa0w\xa7\xd1\xc6\xf7&\xe8\x86\xff\xfa\x90\x9d\x82V\xc0\xa4Qeeq.\x8f'
```