MARMARA UNIVERSITY
FACULTY of ENGINEERING

CSE4057 Information Systems and Security – Homework 1

Abdulhalik ŞENSİN 150119598

Bilal TAN 150119630

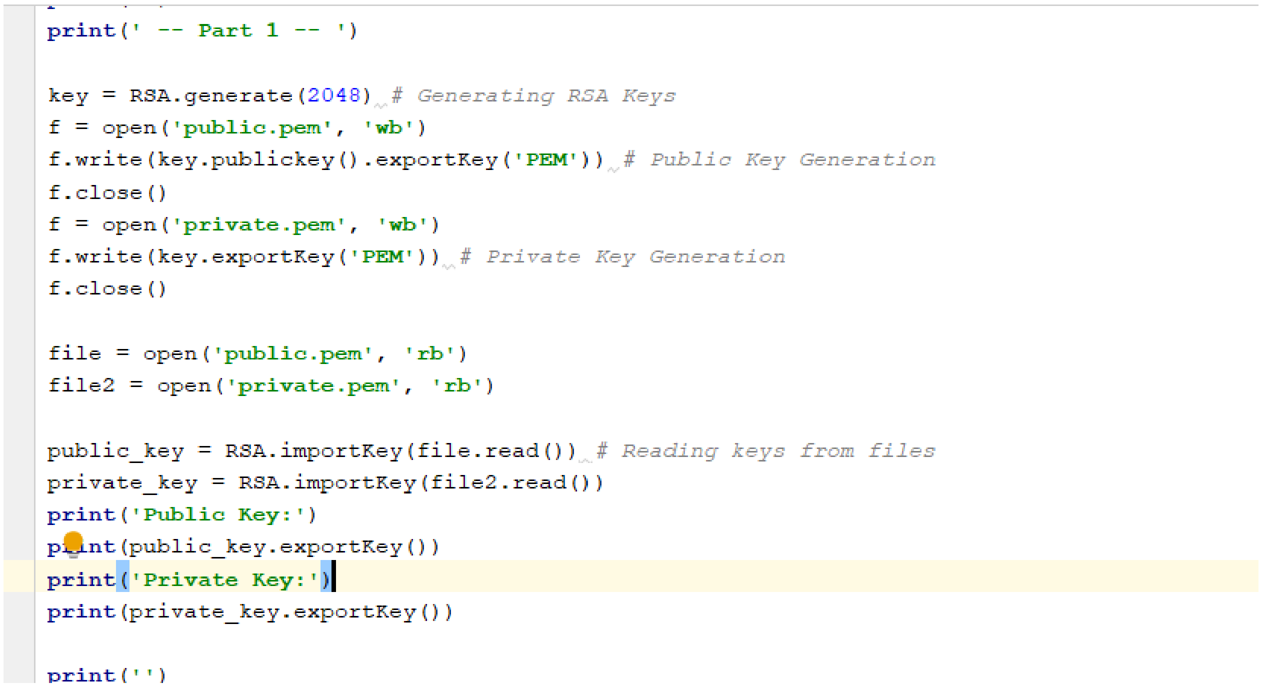
Sinem ONAL 150119576

Question 1

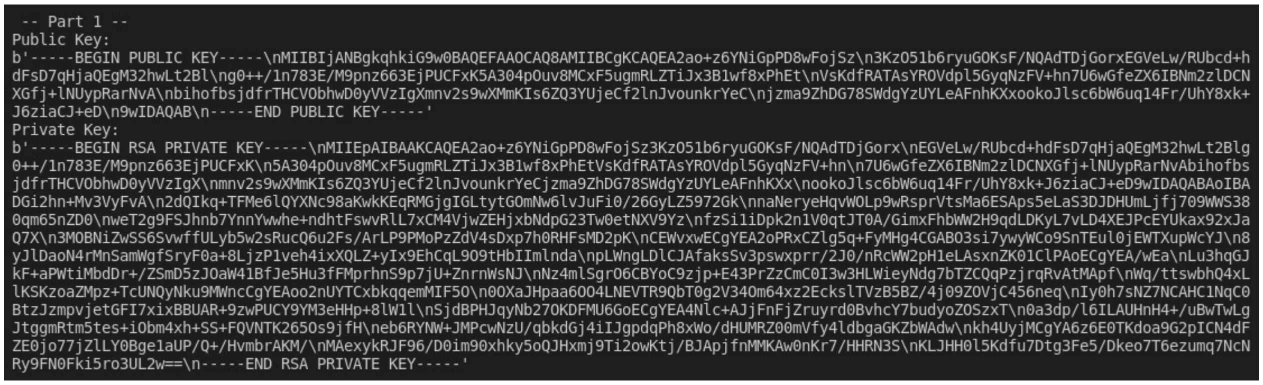
Generate an RSA public-private key pair. KA+ and KA-.

The RSA algorithm was published in 70's by Ron Rivest, Adi Shamir and Leonard Adleman. It is an asymmetric system, which means that a key pair will be generated, a public key and a private key, obviously it keep the private key secure and pass around the public one.

Firstly, we created RSA Key using RSA.generate from a factory method by specifying the algorithm. After that, we initialized RSA.generate with the key size of 2048. We got the public key using key. Publickey() and the public key using key.exportkey() from the KeyPair object.



```python
print(' -- Part 1 -- ')

key = RSA.generate(2048) # Generating RSA Keys
f = open('public.pem', 'wb')
f.write(key.publickey().exportKey('PEM')) # Public Key Generation
f.close()
f = open('private.pem', 'wb')
f.write(key.exportKey('PEM')) # Private Key Generation
f.close()

file = open('public.pem', 'rb')
file2 = open('private.pem', 'rb')

public_key = RSA.importKey(file.read()) # Reading keys from files
private_key = RSA.importKey(file2.read())
print('Public Key:')
print(public_key.exportKey())
print('Private Key:')
print(private_key.exportKey())

print('')
```



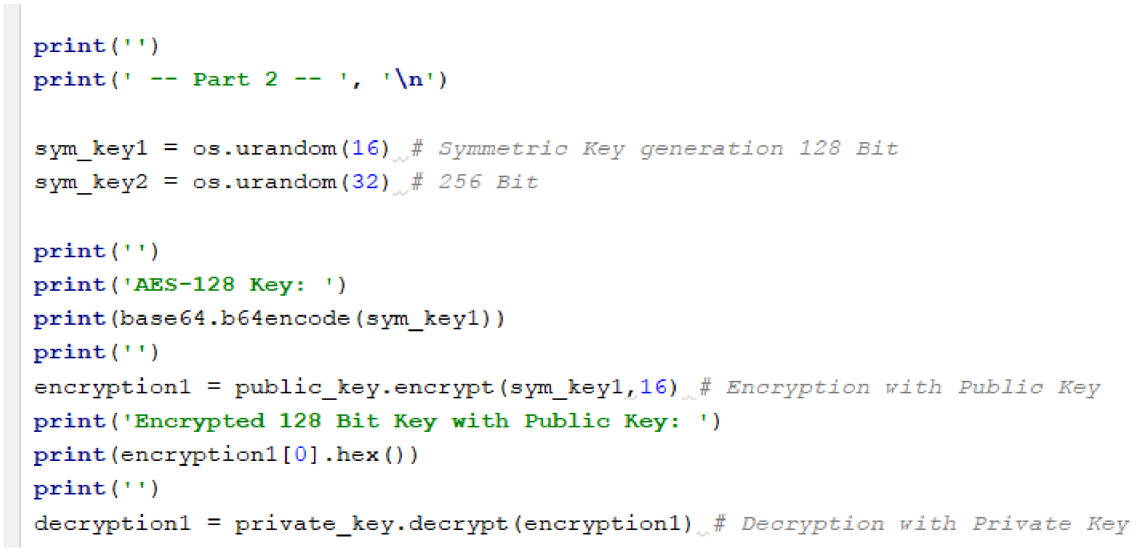
```
 -- Part 1 --
Public Key:
b'-----BEGIN PUBLIC KEY-----\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA2ao+z6YNiGpPD8wFojSz\n3KzO51b6ryuGOKsF/NQAdTDjGorxEGVeLw/RUbcd+h
dFsD7qHjaQEgM32hwLt2Bl\ng0++/1n783E/M9pnz663EjPUCFxK5A304pOuv8MCxF5ugmRLZTiJx3B1wf8xPhEt\nVsKdfRATAsYROVdpl5GyqNzFV+hn7U6wGfeZX6IBNm2zlDCN
XGfj+lNUypRarNvA\nbihofbsjdfrTHCVObhwD0yVVzIgXmnv2s9wXMmKIs6ZQ3YUjeCf2lnJvounkrYeC\njzma9ZhDG78SWdgYzUYLeAFnhKXxookoJlsc6bW6uq14Fr/UhY8xk+
J6ziaCJ+eD\n9wIDAQAB\n-----END PUBLIC KEY-----'
Private Key:
b'-----BEGIN RSA PRIVATE KEY-----\nMIIEpAIBAAKCAQEA2ao+z6YNiGpPD8wFojSz3KzO51b6ryuGOKsF/NQAdTDjGorx\nEGVeLw/RUbcd+hdFsD7qHjaQEgM32hwLt2Blg
0++/1n783E/M9pnz663EjPUCFxK\n5A304pOuv8MCxF5ugmRLZTiJx3B1wf8xPhEtVsKdfRATAsYROVdpl5GyqNzFV+hn\n7U6wGfeZX6IBNm2zlDCNXGfj+lNUypRarNvAbihofbs
jdfrTHCVObhwD0yVVzIgX\nmnv2s9wXMmKIs6ZQ3YUjeCf2lnJvounkrYeCjzma9ZhDG78SWdgYzUYLeAFnhKXx\nookoJlsc6bW6uq14Fr/UhY8xk+J6ziaCJ+eD9wIDAQABAoIBA
DGi2hn+Mv3VyFvA\n2dQIkq+TFMe6lQYXNc98aKwkKEqRMGjgIGLtytGOmNw6lvJuFi0/26GyLZ5972Gk\nnaNeryeHqvWOLp9wRsprVtsMa6ESAps5eLaS3DJDHUmLjfj709WWS38
0qm65nZD0\nweT2g9FSJhnb7YnnYwhe+ndhtFswvRlL7xCM4VjwZEHjxbNdpG23Tw0etNXV9Yz\nfzSi1iDpk2n1V0qtJT0A/GimxFhbWW2H9qdLDKyL7vLD4XEJPcEYUkax92xJa
Q7X\n3MOBNiZwSS6SvwffULyb5w2sRucQ6u2Fs/ArLP9PMoPzZdV4sDxp7h0RHFsMD2pK\nCEWvxwECgYEA2oPRxCZlg5q+FyMHg4CGABO3si7ywyWCo9SnTEul0jEWTXupWcYJ\n8
yJlDaoN4rMnSamWgfSryF0a+8LjzP1veh4ixXQLZ+yIx9EhCqL9O9tHbIImlnda\npLWngLDlCJAfaksSv3pswxprr/2J0/nRcWW2pH1eLAsxnZK01ClPAoECgYEA/wEa\nLu3hqGJ
kF+aPWtiMbdDr+/ZSmD5zJOaW41BfJe5Hu3fFMprhnS9p7jU+ZnrnWsNJ\nNz4mlSgrO6CBYoC9zjp+E43PrZzCmC0I3w3HLWieyNdg7bTZCQqPzjrqRvAtMApf\nWq/ttswbhQ4xL
lKSKzoaZMpz+TcUNQyNku9MWncCgYEAoo2nUYTCxbkqqemMIF5O\n0OXaJHpaa6OO4LNEVTR9QbT0g2V34Om64xz2EckslTVzB5BZ/4j09ZOVjC456neq\nIy0h7sNZ7NCAHC1NqC0
BtzJzmpvjetGFI7xixBBUAR+9zwPUCY9YM3eHHp+8lW1l\nSjdBPHJqyNb27OKDFMU6GoECgYEA4Nlc+AJjFnFjZruyrd0BvhcY7budyoZOSzxT\n0a3dp/l6ILAUHnH4+/uBwTwLg
JtggmRtm5tes+iObm4xh+SS+FQVNTK265Os9jfH\neb6RYNW+JMPcwNzU/qbkdGj4iIJgpdqPh8xWo/dHUMRZ00mVfy4ldbgaGKZbWAdw\nkh4UyjMCgYA6z6E0TKdoa9G2pICN4dF
ZE0jo77jZlLY0Bge1aUP/Q+/HvmbrAKM/\nMAexykRJF96/D0im90xhky5oQJHxmj9Ti2owKtj/BJApjfnMMKAw0nKr7/HHRN3S\nKLJHH0l5Kdfu7Dtg3Fe5/Dkeo7T6ezumq7NcN
Ry9FN0Fki5ro3UL2w==\n-----END RSA PRIVATE KEY-----'
```

Question 2

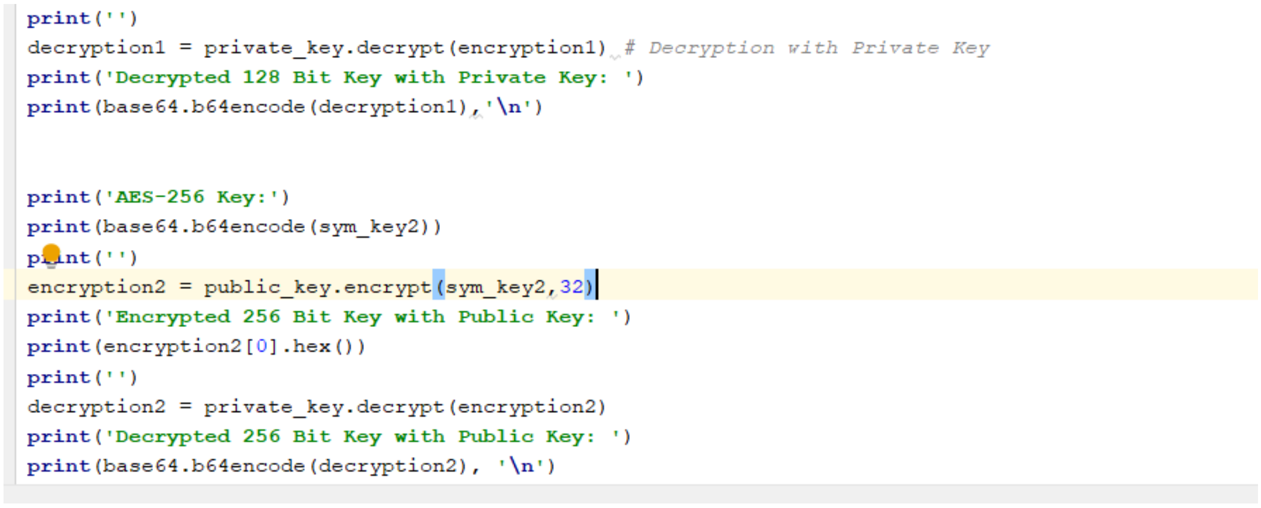
Generate two symmetric keys: 128-bit K1 and 256-bit K2. Print values of the keys on the screen. Encrypt them with KA+, print the results, and then decrypt them with KA-. Again, print the results. Provide a screenshot showing your results.

The symmetric key algorithms was initiated a process to select a symmetric-key encryption algorithm to be used to protect sensitive (unclassified) Federal information in furtherance of NIST's statutory responsibilities by the National Institute of Standards and Technology (NIST) in 1997. It is an encryption system in which the same key is used for the encoding and decoding of the data. The safe distribution of the key is one of the drawbacks of this method, but what it lacks in security it gains in time complexity.

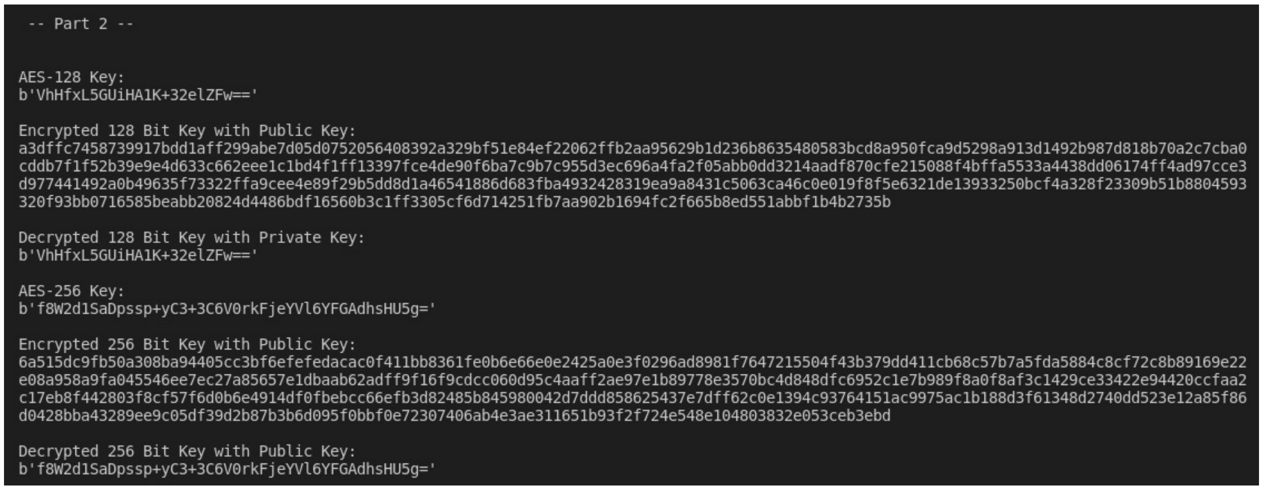
Firstly, we create AES Symmetric Key using os.urandom from a factory method by specifying the algorithm. And then, we initialized the AES with the key size of 128(K1) and 256(K2). We got the public key using Public_key.ecnrypt() and the private key using public_key.decrypt() from the KeyPair object. After that, we encrypted K1 and K2 with RSA asymmetric key KA+ and decrypted them with KA -.



```
print('')
print(' -- Part 2 -- ', '\n')

sym_key1 = os.urandom(16) # Symmetric Key generation 128 Bit
sym_key2 = os.urandom(32) # 256 Bit

print('')
print('AES-128 Key: ')
print(base64.b64encode(sym_key1))
print('')
encryption1 = public_key.encrypt(sym_key1,16) # Encryption with Public Key
print('Encrypted 128 Bit Key with Public Key: ')
print(encryption1[0].hex())
print('')
decryption1 = private_key.decrypt(encryption1) # Decryption with Private Key
```



```
print('')
decryption1 = private_key.decrypt(encryption1) # Decryption with Private Key
print('Decrypted 128 Bit Key with Private Key: ')
print(base64.b64encode(decryption1),'\n')


print('AES-256 Key:')
print(base64.b64encode(sym_key2))
print('')
encryption2 = public_key.encrypt(sym_key2,32)
print('Encrypted 256 Bit Key with Public Key: ')
print(encryption2[0].hex())
print('')
decryption2 = private_key.decrypt(encryption2)
print('Decrypted 256 Bit Key with Public Key: ')
print(base64.b64encode(decryption2), '\n')
```



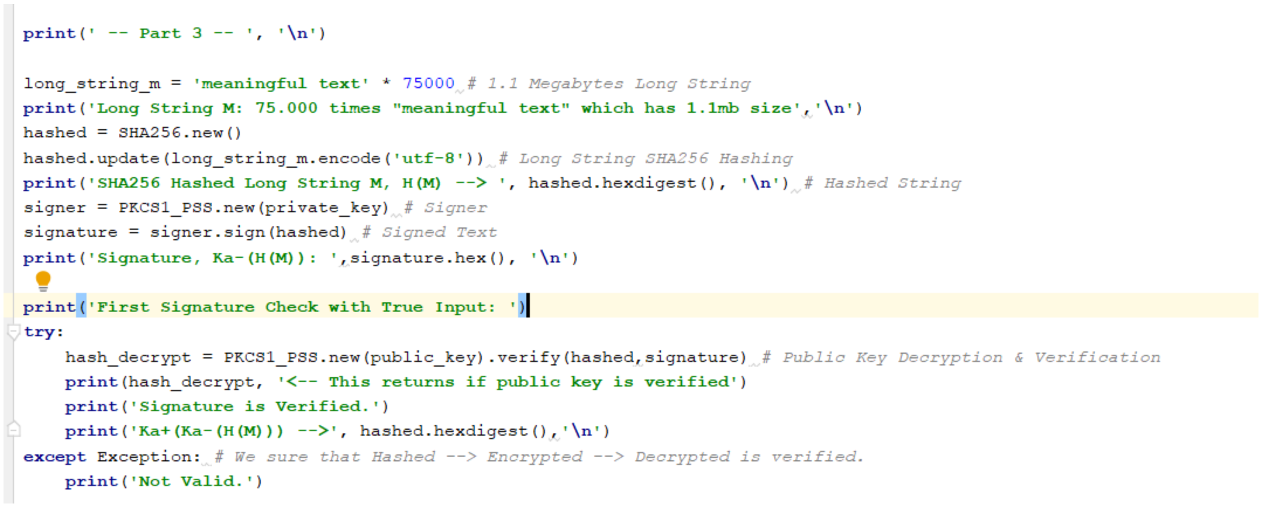
```
 -- Part 2 --


AES-128 Key:
b'VhHfxL5GUiHA1K+32elZFw=='

Encrypted 128 Bit Key with Public Key:
a3dffc7458739917bdd1aff299abe7d05d0752056408392a329bf51e84ef22062ffb2aa95629b1d236b8635480583bcd8a950fca9d5298a913d1492b987d818b70a2c7cba0
cddb7f1f52b39e9e4d633c662eee1c1bd4f1ff13397fce4de90f6ba7c9b7c955d3ec696a4fa2f05abb0dd3214aadf870cfe215088f4bffa5533a4438dd06174ff4ad97cce3
d977441492a0b49635f73322ffa9cee4e89f29b5dd8d1a46541886d683fba4932428319ea9a8431c5063ca46c0e019f8f5e6321de13933250bcf4a328f23309b51b8804593
320f93bb0716585beabb20824d4486bdf16560b3c1ff3305cf6d714251fb7aa902b1694fc2f665b8ed551abbf1b4b2735b

Decrypted 128 Bit Key with Private Key:
b'VhHfxL5GUiHA1K+32elZFw=='

AES-256 Key:
b'f8W2d1SaDpssp+yC3+3C6V0rkFjeYVl6YFGAdhsHU5g='

Encrypted 256 Bit Key with Public Key:
6a515dc9fb50a308ba94405cc3bf6efefedacac0f411bb8361fe0b6e66e0e2425a0e3f0296ad8981f7647215504f43b379dd411cb68c57b7a5fda5884c8cf72c8b89169e22
e08a958a9fa045546ee7ec27a85657e1dbaab62adff9f16f9cdcc060d95c4aaff2ae97e1b89778e3570bc4d848dfc6952c1e7b989f8a0f8af3c1429ce33422e94420ccfaa2
c17eb8f442803f8cf57f6d0b6e4914df0fbebcc66efb3d82485b845980042d7ddd858625437e7dff62c0e1394c93764151ac9975ac1b188d3f61348d2740dd523e12a85f86
d0428bba43289ee9c05df39d2b87b3b6d095f0bbf0e72307406ab4e3ae311651b93f2f724e548e104803832e053ceb3ebd

Decrypted 256 Bit Key with Public Key:
b'f8W2d1SaDpssp+yC3+3C6V0rkFjeYVl6YFGAdhsHU5g='
```

Question 3

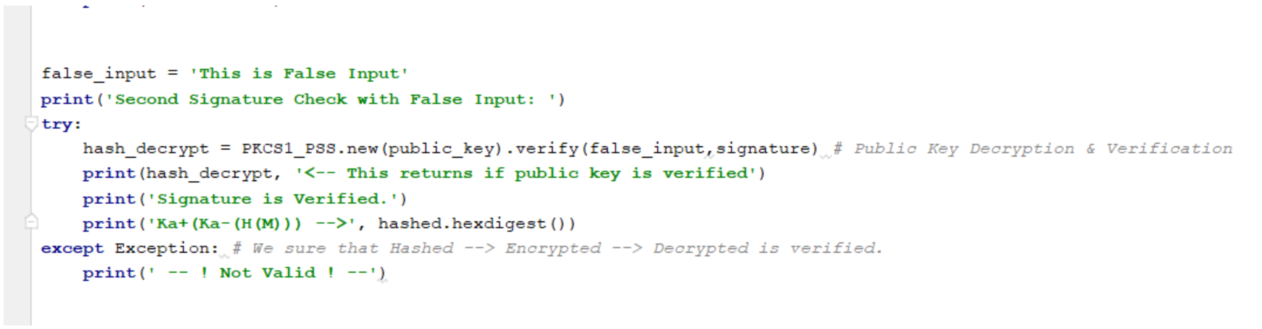
Consider a long text m. Apply SHA256 Hash algorithm (Obtain the message digest, H(m)). Then encrypt it with KA-. (Thus, generate a digital signature.) Then verify the digital signature. (Decrypt it with KA+, apply Hash algorithm to the message, compare). Print m, H(m) and digital signature on the screen. Provide a screenshot. (Or you may print in a file and provide the file).

The SHA256 designed by the NSA as a member of the SHA-2 cryptographic hash functions. It is a cryptographic hash function with digest length of 256 bits. It is a keyless hash function; that is, an MDC (Manipulation Detection Code).

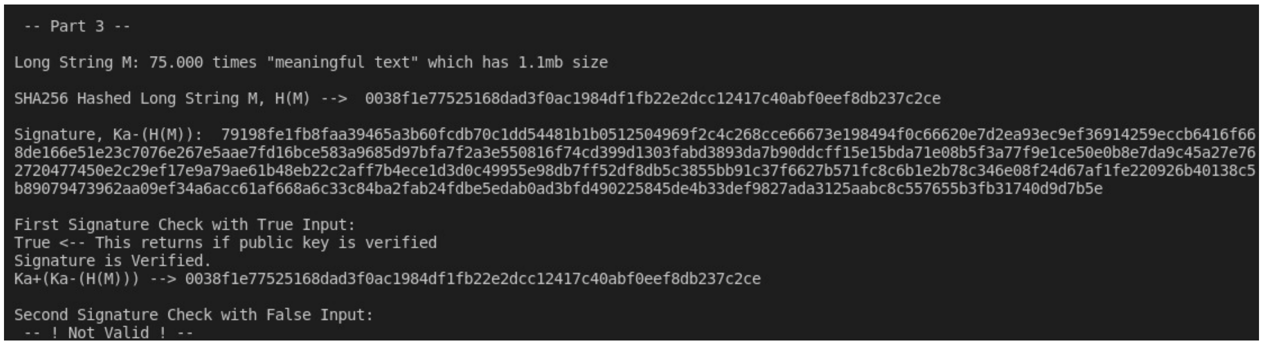
We chose the text as a long knee. We digested the message using hashed from a factory method, specifying the SHA256 algorithm. Next, we encrypted the message with the RSA asymmetric key KA - this is a digital signature. After that, we decrypted the signature with the RSA asymmetric key KA +. Finally, we compared the hash message (H (m)) with the decrypted digital signature and verified the digital signature.



```python
print(' -- Part 3 -- ', '\n')

long_string_m = 'meaningful text' * 75000 # 1.1 Megabytes Long String
print('Long String M: 75.000 times "meaningful text" which has 1.1mb size','\n')
hashed = SHA256.new()
hashed.update(long_string_m.encode('utf-8')) # Long String SHA256 Hashing
print('SHA256 Hashed Long String M, H(M) --> ', hashed.hexdigest(), '\n') # Hashed String
signer = PKCS1_PSS.new(private_key) # Signer
signature = signer.sign(hashed) # Signed Text
print('Signature, Ka-(H(M)): ',signature.hex(), '\n')

print('First Signature Check with True Input: ')
try:
    hash_decrypt = PKCS1_PSS.new(public_key).verify(hashed,signature) # Public Key Decryption & Verification
    print(hash_decrypt, '<-- This returns if public key is verified')
    print('Signature is Verified.')
    print('Ka+(Ka-(H(M)) -->', hashed.hexdigest(),'\n')
except Exception: # We sure that Hashed --> Encrypted --> Decrypted is verified.
    print('Not Valid.')
```



```python
false_input = 'This is False Input'
print('Second Signature Check with False Input: ')
try:
    hash_decrypt = PKCS1_PSS.new(public_key).verify(false_input,signature) # Public Key Decryption & Verification
    print(hash_decrypt, '<-- This returns if public key is verified')
    print('Signature is Verified.')
    print('Ka+(Ka-(H(M)) -->', hashed.hexdigest())
except Exception: # We sure that Hashed --> Encrypted --> Decrypted is verified.
    print(' -- ! Not Valid ! --')
```



```
 -- Part 3 --

Long String M: 75.000 times "meaningful text" which has 1.1mb size

SHA256 Hashed Long String M, H(M) -->  0038f1e77525168dad3f0ac1984df1fb22e2dcc12417c40abf0eef8db237c2ce

Signature, Ka-(H(M)):  79198fe1fb8faa39465a3b60fcdb70c1dd54481b1b0512504969f2c4c268cce66673e198494f0c66620e7d2ea93ec9ef36914259eccb6416f66
8de166e51e23c7076e267e5aae7fd16bce583a9685d97bfa7f2a3e550816f74cd399d1303fabd3893da7b90ddcff15e15bda71e08b5f3a77f9e1ce50e0b8e7da9c45a27e76
2720477450e2c29ef17e9a79ae61b48eb22c2aff7b4ece1d3d0c49955e98db7ff52df8db5c3855bb91c37f6627b571fc8c6b1e2b78c346e08f24d67af1fe220926b40138c5
b89079473962aa09ef34a6acc61af668a6c33c84ba2fab24fdbe5edab0ad3bfd490225845de4b33def9827ada3125aabc8c557655b3fb31740d9d7b5e

First Signature Check with True Input:
True <-- This returns if public key is verified
Signature is Verified.
Ka+(Ka-(H(M))) --> 0038f1e77525168dad3f0ac1984df1fb22e2dcc12417c40abf0eef8db237c2ce

Second Signature Check with False Input:
 -- ! Not Valid ! --
```

Question 4

Use any image file of size more than 1MB. Now consider following three algorithms:

i) AES (128 bit key) in CBC mode.

ii) AES (256 bit key) in CBC mode.

iii) AES (256 bit key) in CTR mode.

For each of the above algorithms, do the following:

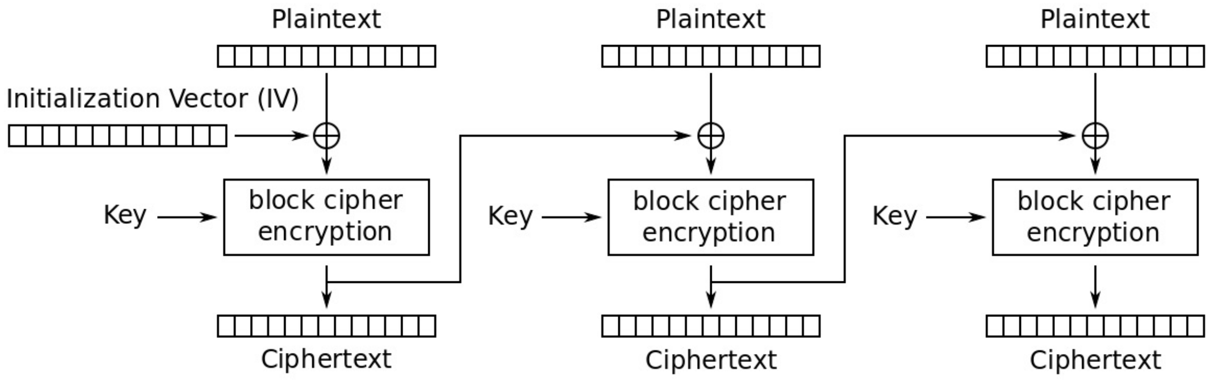
a) Encrypt the image file. Store the result (and submit it with the homework) (Note: Initialization Vector (IV) in CBC mode and nonce in CTR mode should be generated randomly, Key = $K1$ or $K2$).

b) Decrypt the file and store the result. Show that it is the same as the original image file.

c) Measure the time elapsed for encryption. Write it in your report. Comment on the result.

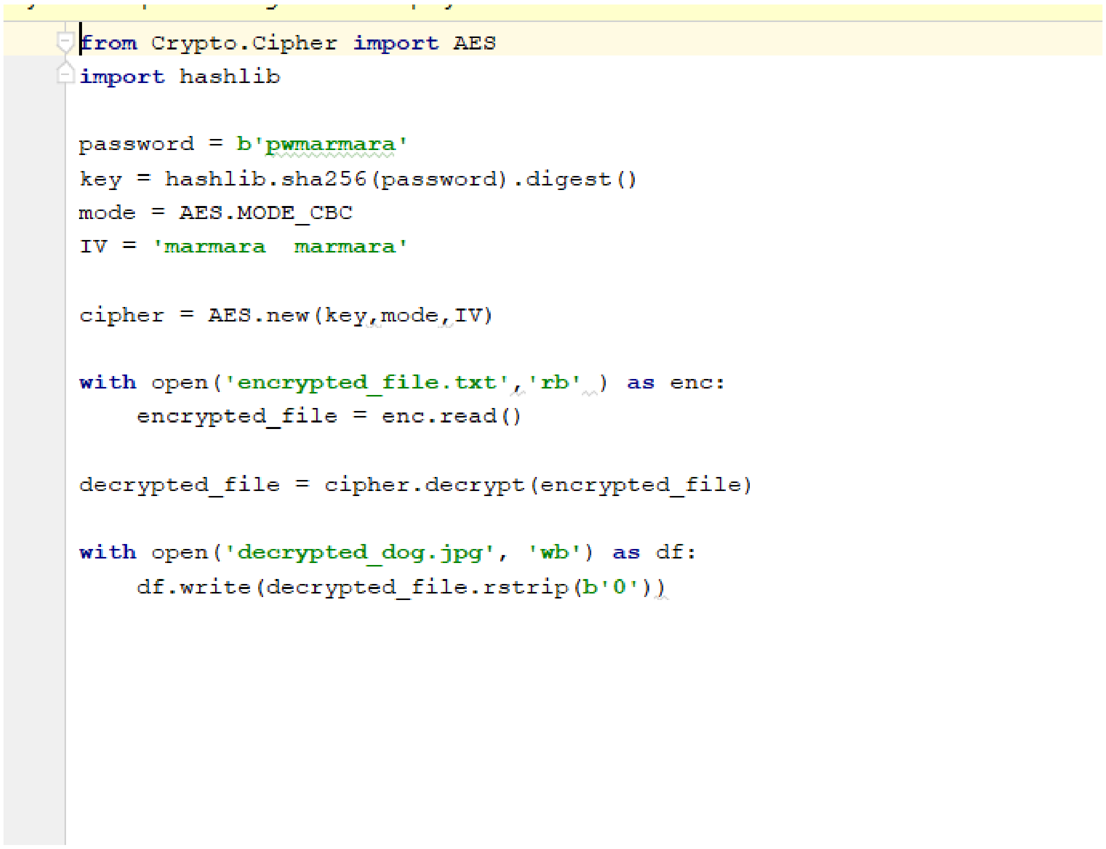
d) For the first algorithm, change Initialization Vector (IV) and show that the corresponding ciphertext chages for the same plaintext (Give the result for both).

The CBC encryption mode was invented in IBM in 1976. This mode is about adding XOR each plaintext block to the ciphertext block that was previously produced. The result is then encrypted using the cipher algorithm in the usual way. As a result, every subsequent ciphertext 10 block depends on the previous one. The first plaintext block is added XOR to a random initialization vector (commonly referred to as IV). The vector has the same size as a plaintext block.

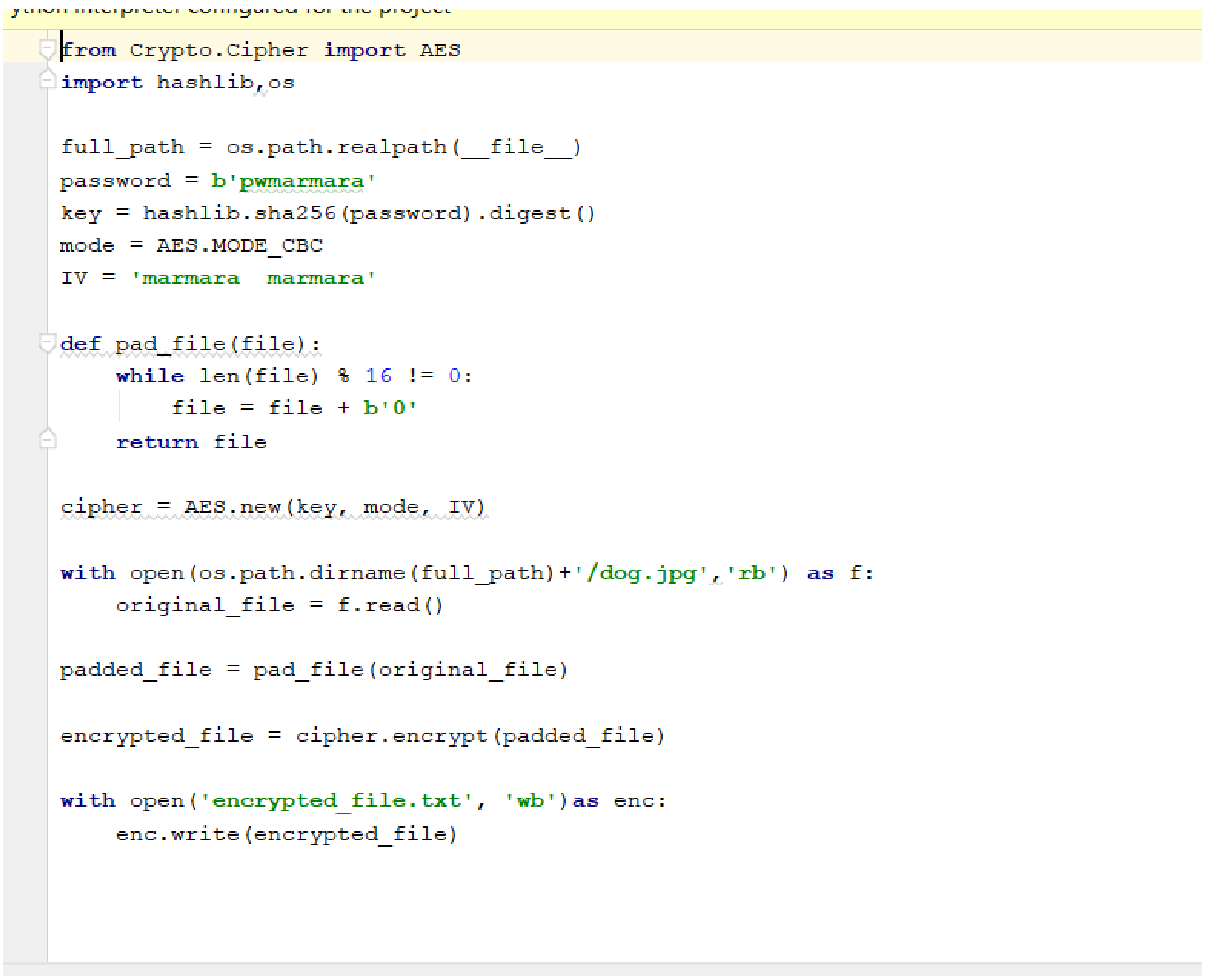


Cipher Block Chaining (CBC) mode encryption

During decrypting of a ciphertext block, one should add XOR the output data received from the decryption algorithm to the previous ciphertext block. Because the receiver knows all the ciphertext blocks just after obtaining the encrypted message, he can decrypt the message using many threads simultaneously.

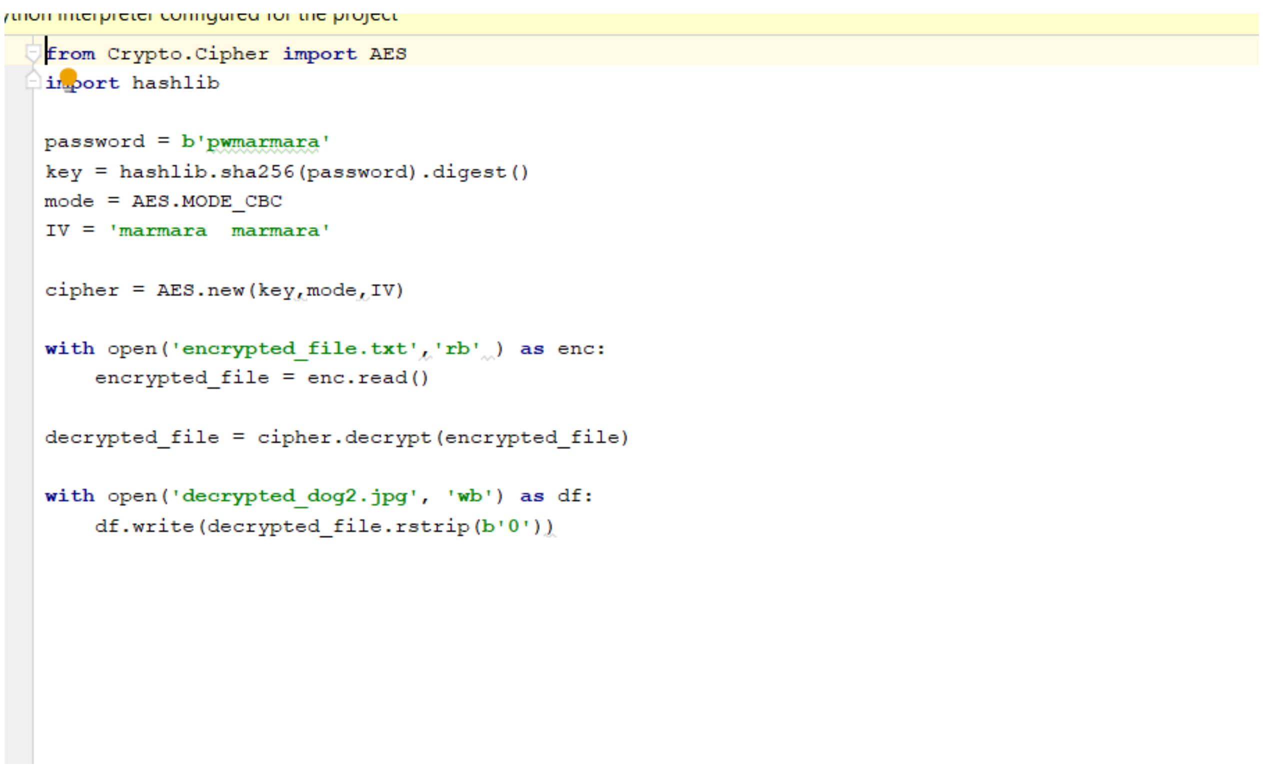


```python
from Crypto.Cipher import AES
import hashlib

password = b'pwmarmara'
key = hashlib.sha256(password).digest()
mode = AES.MODE_CBC
IV = 'marmara  marmara'

cipher = AES.new(key,mode,IV)

with open('encrypted_file.txt','rb' ) as enc:
    encrypted_file = enc.read()

decrypted_file = cipher.decrypt(encrypted_file)

with open('decrypted_dog.jpg', 'wb') as df:
    df.write(decrypted_file.rstrip(b'0'))
```

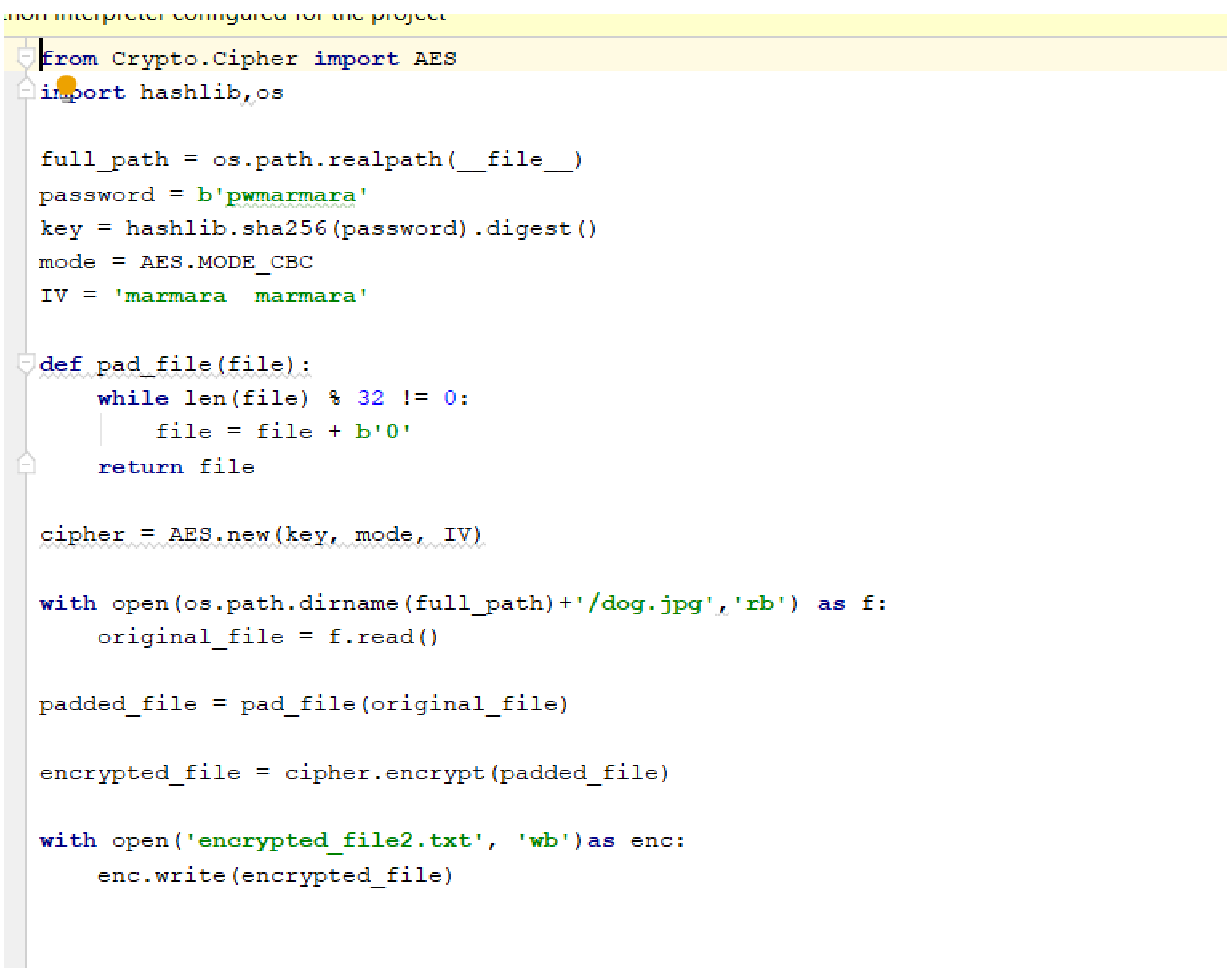


ython interpreter configured for the project

```python
from Crypto.Cipher import AES
import hashlib,os

full_path = os.path.realpath(__file__)
password = b'pwmarmara'
key = hashlib.sha256(password).digest()
mode = AES.MODE_CBC
IV = 'marmara  marmara'

def pad_file(file):
    while len(file) % 16 != 0:
        file = file + b'0'
    return file

cipher = AES.new(key, mode, IV)

with open(os.path.dirname(full_path)+'/dog.jpg','rb') as f:
    original_file = f.read()

padded_file = pad_file(original_file)

encrypted_file = cipher.encrypt(padded_file)

with open('encrypted_file.txt', 'wb')as enc:
    enc.write(encrypted_file)
```



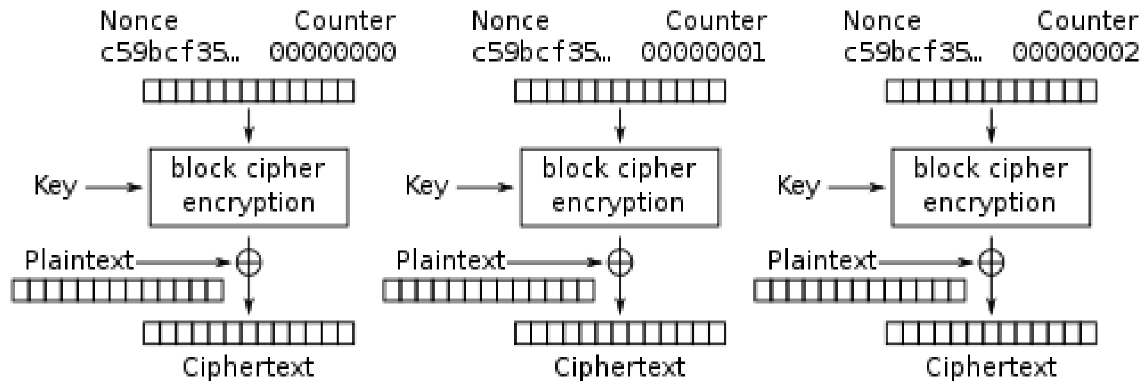
```python
from Crypto.Cipher import AES
import hashlib

password = b'pwmarmara'
key = hashlib.sha256(password).digest()
mode = AES.MODE_CBC
IV = 'marmara  marmara'

cipher = AES.new(key,mode,IV)

with open('encrypted_file.txt','rb' ) as enc:
    encrypted_file = enc.read()

decrypted_file = cipher.decrypt(encrypted_file)

with open('decrypted_dog2.jpg', 'wb') as df:
    df.write(decrypted_file.rstrip(b'0'))
```



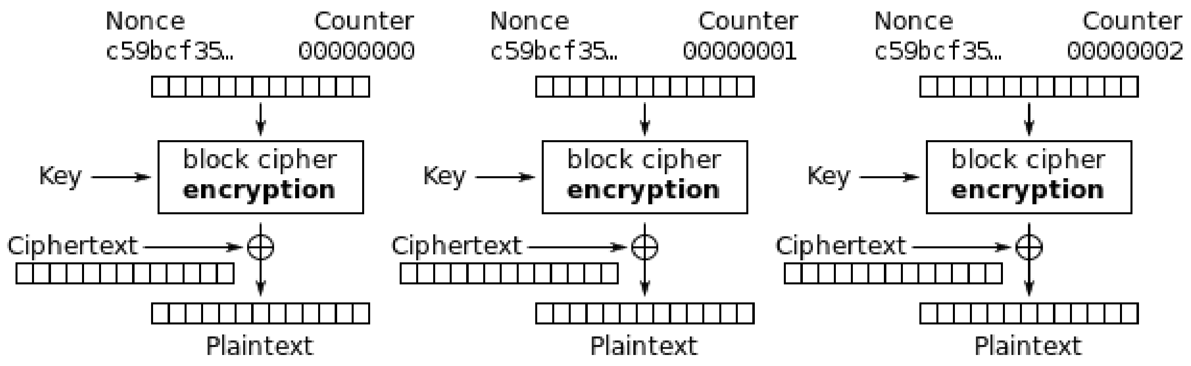
```python
from Crypto.Cipher import AES
import hashlib,os

full_path = os.path.realpath(__file__)
password = b'pwmarmara'
key = hashlib.sha256(password).digest()
mode = AES.MODE_CBC
IV = 'marmara  marmara'

def pad_file(file):
    while len(file) % 32 != 0:
        file = file + b'0'
    return file

cipher = AES.new(key, mode, IV)

with open(os.path.dirname(full_path)+'/dog.jpg','rb') as f:
    original_file = f.read()

padded_file = pad_file(original_file)

encrypted_file = cipher.encrypt(padded_file)

with open('encrypted_file2.txt', 'wb')as enc:
    enc.write(encrypted_file)
```

For the CTR MODE ;

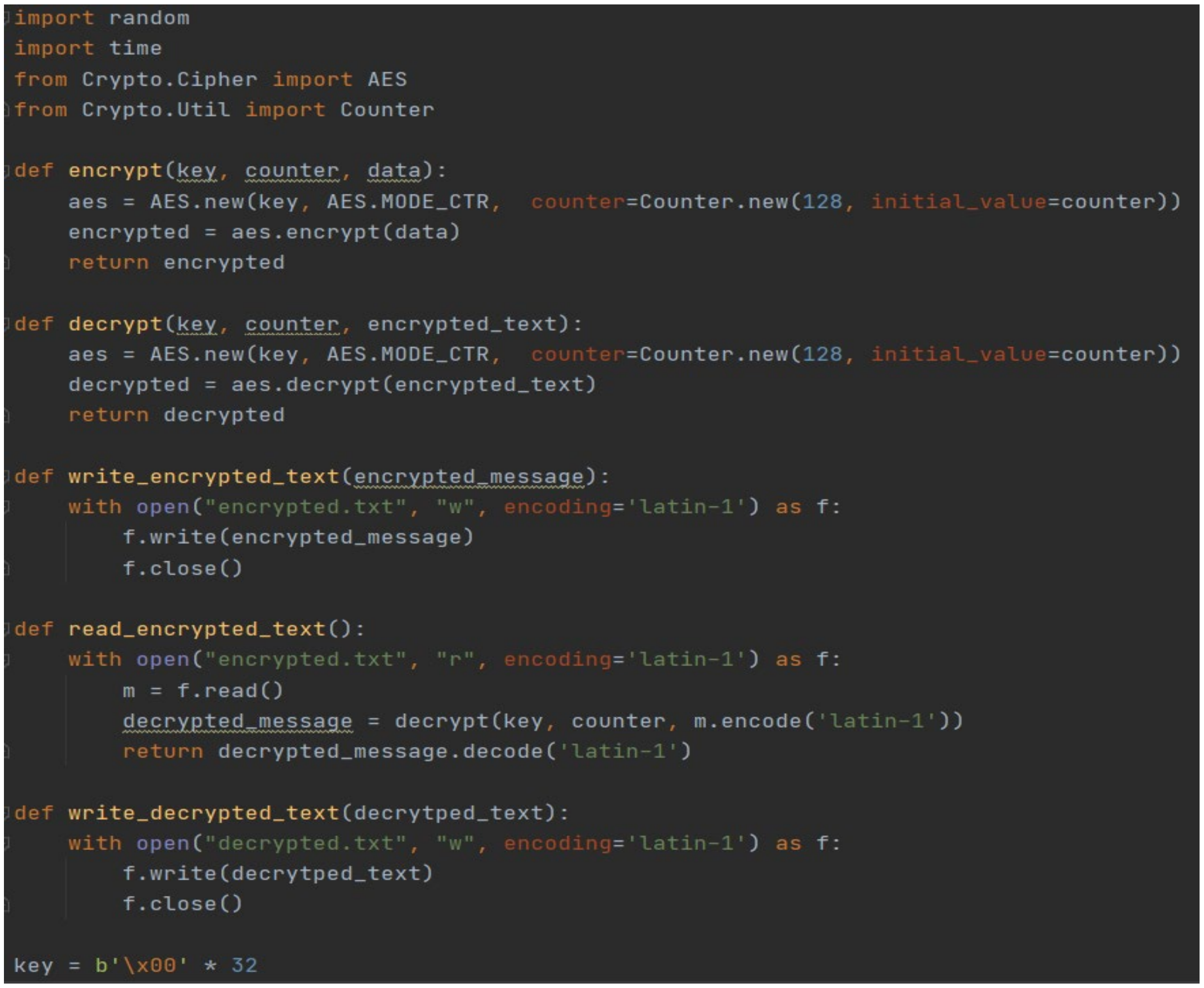
AES-CTR has many properties that make it an attractive encryption algorithm for in high-speed networking. **AES-CTR** uses the **AES** block cipher to create a stream cipher. Data is encrypted and decrypted by XORing with the key stream produced by **AES** encrypting sequential **counter** block values.



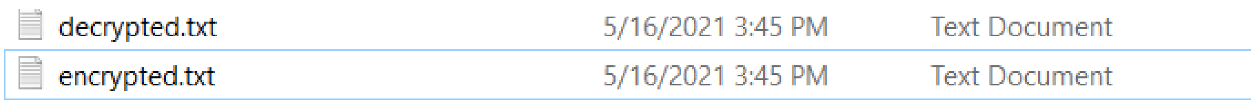
Counter (CTR) mode encryption



Counter (CTR) mode decryption



```
import random
import time
from Crypto.Cipher import AES
from Crypto.Util import Counter

def encrypt(key, counter, data):
    aes = AES.new(key, AES.MODE_CTR,  counter=Counter.new(128, initial_value=counter))
    encrypted = aes.encrypt(data)
    return encrypted

def decrypt(key, counter, encrypted_text):
    aes = AES.new(key, AES.MODE_CTR,  counter=Counter.new(128, initial_value=counter))
    decrypted = aes.decrypt(encrypted_text)
    return decrypted

def write_encrypted_text(encrypted_message):
    with open("encrypted.txt", "w", encoding='latin-1') as f:
        f.write(encrypted_message)
        f.close()

def read_encrypted_text():
    with open("encrypted.txt", "r", encoding='latin-1') as f:
        m = f.read()
        decrypted_message = decrypt(key, counter, m.encode('latin-1'))
        return decrypted_message.decode('latin-1')

def write_decrypted_text(decrytped_text):
    with open("decrypted.txt", "w", encoding='latin-1') as f:
        f.write(decrytped_text)
        f.close()

key = b'\x00' * 32
```



decrypted.txt	5/16/2021 3:45 PM	Text Document
encrypted.txt	5/16/2021 3:45 PM	Text Document

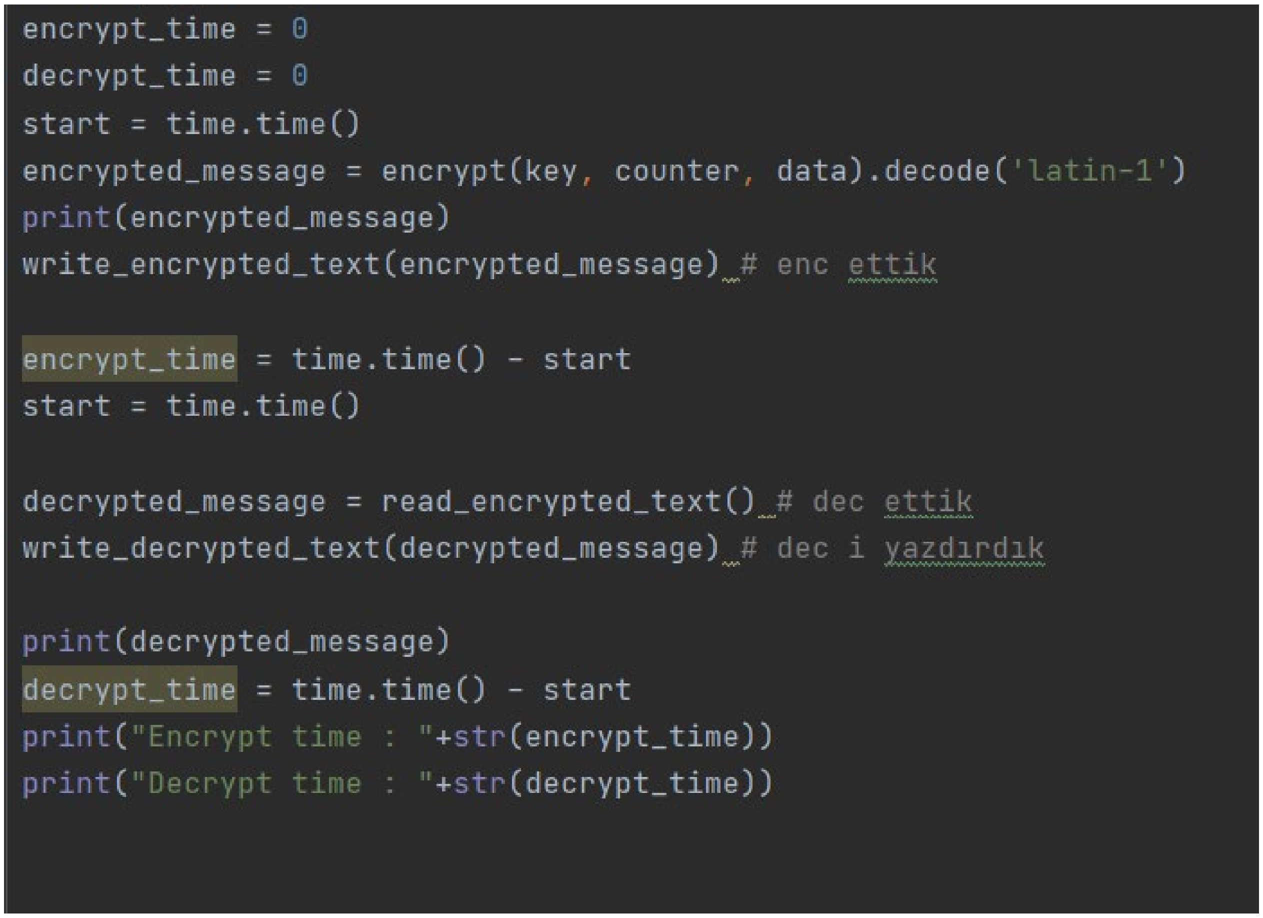
Here, for the aes ctr mode we create counter randomly then encrypt the key and counter. In order to get cipher text XORing the plaintext and counter which encrypted by key. We store the encrypted message as a text file.

```
□î□®":,x€-'‘cÜ+«ÖÖ~Mòá@”ÄJ¡P£ÒKuþÊ4□ˆ6ÑÈÿ3›}±dG ¿±        ä´Ôç^<Ö'Ä¥…Z□ 5%□`Ô»ŻÏþxÄw(4□□
;³□í(5□QuÁ)Ë¯¨Ëª7ëT~□ºu□%´)G□»
-:ò§k™ÉƒŽ        Ø□ðs□w8Y¦:‹‡ô□ ’Ò□□ÃùuŽÍ]ÁŒAB□h¡ç¦éÉWbãª_Èyˆx£$@®ážôr³□Ý□ ;6Sÿ·¦8§□□yÑûÝÃÑŠ›Ç%R
äOÀÞ½¾¿LŒJÊˆü3ªçÄ—y¬M·µ1õ¤™*ˆ¦“eEÑ™¯Ž□nëA=‰òÈ□_ÀaX±F*ÙA□  Z…ÎZ¥□é□Ø…□Ò□L05U£…û□æ&DªXEýƒ□¦$<•)Øâ -à□ÅÊÊÊÙ¢U1ß1bš×;Č
±Œú dQ¤9_ê”“□Çn□A¿YªÇè[“9ôãBŽì*ˆù□fÄa¨¡csZø¦õ^[pBqb¾›ø^mÞ□®€oo?&²’è ¸=èKv©á’Í:m`
□‘…k= š(O¾ PÁÜ†ž87# R,
ðì˜b□¼Ê©ÝÎÄ8Dð»„X □8Ÿemú^Ì        6k□fjI€"V¤Tÿ $%d§h´£Èÿ1÷□□;ß,O*´sgñb½0šT‘ä¹©!"Žeôªv5)¬g□þÄjµ¹s□À7G□□ÿsvWg0H-ök—C
âîç}Æ‡Ø]iD□¶ž□™□,^”KÈIîp
â…³'‰…ä©† )□ŸI7Y÷^õ♠Š¿□¡"W‰y{‘ÔvU…ÇÅ      >¬Ç…jU·‹û♠□6ÍÜ d□fŠ[Y□9r□ hÂ□c/òòxðô:G§x÷™b□rÆnâ$íÙ;™ýF□□ÇÚý□”¾³zþ„J
Û□z"xHV¬ëÐ©+ Ç6(£~í□‡ 3^4□□ Ó/\µ½-ÐY{/□_\□åAH □fe-¼Ix’†5@Å=Áðfò)þ˜uÈG(#Y?Í□¡Ö/□□È¶åêW‰„□□ÌÀ
É&□èp1û□HÛžþ□Ên2´ÊÝ□üU>œp[□□_□›ÚQÂ□ÖÎåÂ“\ùüþd¼~*b….u™½\Ê[Š¥<]qþ%¡-Ï-ç©1ão?øµ¥rû¦ %□'¬uãh(□—†…w□cžQb fÏæ□F□tÒ
\Ú‡ôÁÇ^4ˆ¼|;rNöº=q” Çwèh¹□□±èô~\□□™Q«`t¸¹˜žJdi…Ú□Äøg<ÆËž’ÔÃ±□À□-hØâ □’~u□l/S;>Xº®|ƒ[X²fÇ`Ú□□] k×+×~‘š¡ä,□ýŸ„Ñ□µÒ¤r>-
[žŸµVœ‹ÕÉ#]óV×>§UK□:¯□□ÀT/ýïrk□hl;.
…ÿ¶—+Ô‘CÎùµ„z□š¤□½èRRpÿÒ»J#ZbC3•ˆ□@N”¶y#AV-□4>èø"³ÎD ã ½¾byüˆ•8²’E€g€‹□□á(ïj*□é¥
€+•Ás"DÁò|Í1åXwð„Jç‰ºžy…ŸÉ
œÎ □ã{+□RXanvWµ±h
’ê]dWá□pkÙ‘V„â□□□ÚÙ□çÀnÄÿzÐG”_éXíU□c,ÃÈ3|Ú¼¼¡áLÓò□öðo±)□F□DŸ_'óHOdFWdÖŸBÅ‘îW ”†♠B¨Œ‡R¿□€êcN„
```

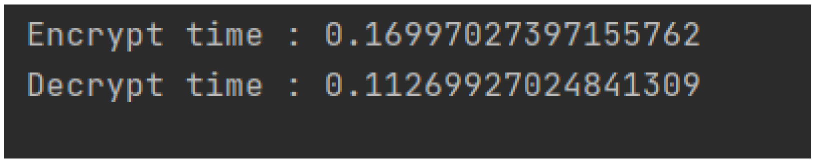
For the decryption, receiver know the counter. We encrypt counter and key first, then XOR the cipher text and encrypted counter. It gives us the original message.

```
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful sextmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful te□tmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful sextmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningfrl textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textjeaningful textmeaningful textmeaningaul textmeaningful textmeaningful sextmeaningful textmeaningful
textmeaningful textmeaniigful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful tb□tmeaningfrl textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful sexsmbaningful textmeaningful textmeaningful textmeanin`ful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful'textmeaningful textmeaningful textmeaningfrl textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaiingful textmeaningful
textmeaningful textmeaningfuk textmeaningful textmeaningful textmeaningful te□tmefningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful sextmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaniigful textmeaningful textmeaniigful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaiingful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmbaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmbaningful textmeaningful textmeaningful
textmeanin`ful textmbaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful'textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeanin`ful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful te□tmeaningful textmeaningful
textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
textmeaningful textmeaningful textmeaningfrl textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful textmeaningful
```

Finally, we measure the encrypt time and decrypt time.



```python
encrypt_time = 0
decrypt_time = 0
start = time.time()
encrypted_message = encrypt(key, counter, data).decode('latin-1')
print(encrypted_message)
write_encrypted_text(encrypted_message) # enc ettik

encrypt_time = time.time() - start
start = time.time()

decrypted_message = read_encrypted_text() # dec ettik
write_decrypted_text(decrypted_message) # dec i yazdırdık

print(decrypted_message)
decrypt_time = time.time() - start
print("Encrypt time : "+str(encrypt_time))
print("Decrypt time : "+str(decrypt_time))
```



```
Encrypt time : 0.16997027397155762
Decrypt time : 0.11269927024841309
```