

REPORT S3/L3

TRACCIA

Dato il seguente messaggio trovare la chiave e decifrarlo:

HSNFRGH

QWJhIHZ6b2VidHl2bmdyIHb1ciB6ciBhciBucHBiZX Ri

Se uso il cifrario di cesare con chiave 3 per il primo messaggio, spostando di 3 lettere a sinistra otterrò:

EPICODE

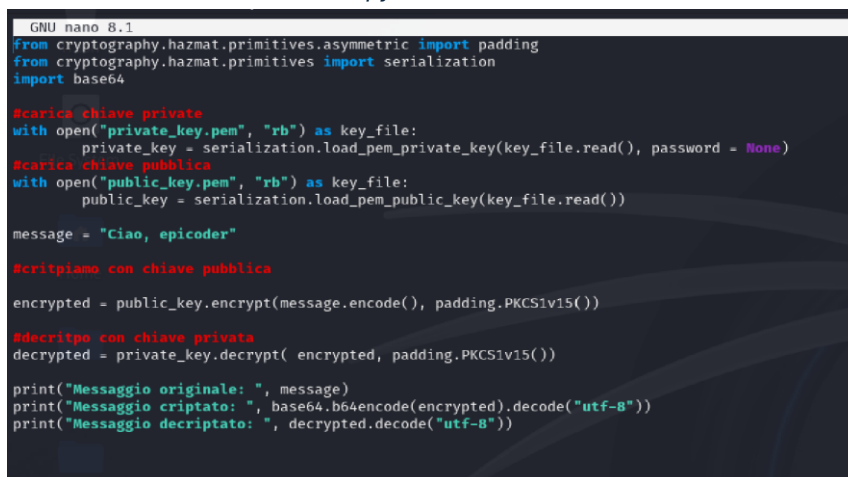
Per quanto riguarda il secondo messaggio non avendo una chiave ho cercato quale fossero i metodi più comuni di crittografia, sono andato quindi per prima cosa a convertirlo in base64 (converte dati binari in stringhe ascii) e utilizzo un famoso cifrario di cesare che è il rot13 per decrittare:

Non imbrogliate che me ne accorgo

ESERCIZIO BONUS

Per prima cosa andremo a generare le chiavi pubblica e privata direttamente da terminale, generando due file, rispettivamente **private_key.pem** e **public_key.pem**.

Andiamo a creare il file **encdec.py** con il comando **touch** e andremo ad editarlo con **nano** come segue.



```
GNU nano 8.1
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.hazmat.primitives import serialization
import base64

#carica chiave privata
with open("private_key.pem", "rb") as key_file:
    private_key = serialization.load_pem_private_key(key_file.read(), password = None)
#carica chiave pubblica
with open("public_key.pem", "rb") as key_file:
    public_key = serialization.load_pem_public_key(key_file.read())

message = "Ciao, epicoder"

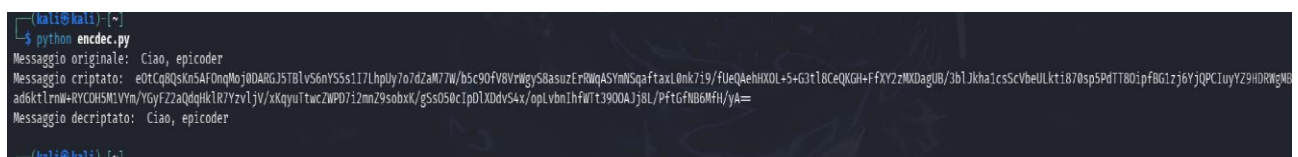
#crittiamo con chiave pubblica
encrypted = public_key.encrypt(message.encode(), padding.PKCS1v15())

#decritto con chiave privata
decrypted = private_key.decrypt(encrypted, padding.PKCS1v15())

print("Messaggio originale: ", message)
print("Messaggio criptato: ", base64.b64encode(encrypted).decode("utf-8"))
print("Messaggio decrittato: ", decrypted.decode("utf-8"))
```

Il modulo **padding** mi permette di aggiungere dati ad un messaggio in maniera da renderlo della lunghezza tale per essere crittografato, il modulo **serialization** mi permetterà di leggere chiavi pubbliche e private ed infine **base64** mi permette di leggere in base 64. Con **open** invece leggeremo dai file le variabili trasferendole in **private_key** e **public_key**.

Dopo aver scritto il programma con python lo andremo ad eseguire per verificare che effettivamente gli output siano il messaggio originale, il messaggio criptato e la sua versione decrittata ovviamente uguale al messaggio originale.



```
(kali@kali)~$ python encdec.py
Messaggio originale: Ciao, epicoder
Messaggio criptato: eOtCqBQsKt5AFOnqMoJ0dARGJ5TbLV56nY55s1I7LhpUy7o7dZaM7W/b5c90Fv8VrWgYS8asuzErWqASymNSqaftaxL0mk7i9/fueQ4ehHXOL+5+G3t18CeQKGH+FfXY2zMXDagUB/3b1Jkha1csScVbeULkt1870sp5P0TT80ipfBG1zj6YjQPC1uyYZ9H0RWgNB
adokt1rm+RYCQH5M1VYm/YGyFZ2aQdqHkLR7YzvljV/xKqyuTwcZWP07i2mnZ9sobxK/gSs050c1pDLX0dvS4x/oplvbn1hfWtT3900AJj8L/PftGfNB6MH/yA=
Messaggio decrittato: Ciao, epicoder
(kali@kali)~$
```