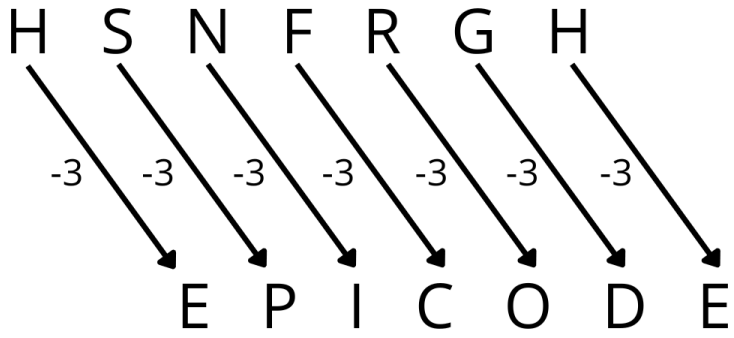
S3/L4 crittografia

Il laboratorio di oggi sulla crittografia richiedeva di trovare il testo in chiaro, partendo da un messaggio cifrato.

Il messaggio cifrato era "**HSNFRGH**"

Ho ipotizzato che il messaggio fosse stato criptato tramite il cifrario di Cesare, una tipologia di cifrario utilizzata da Giulio Cesare per proteggere le comunicazioni militari. Il cifrario di Cesare si basa sullo spostamento fisso delle lettere dell'alfabeto.

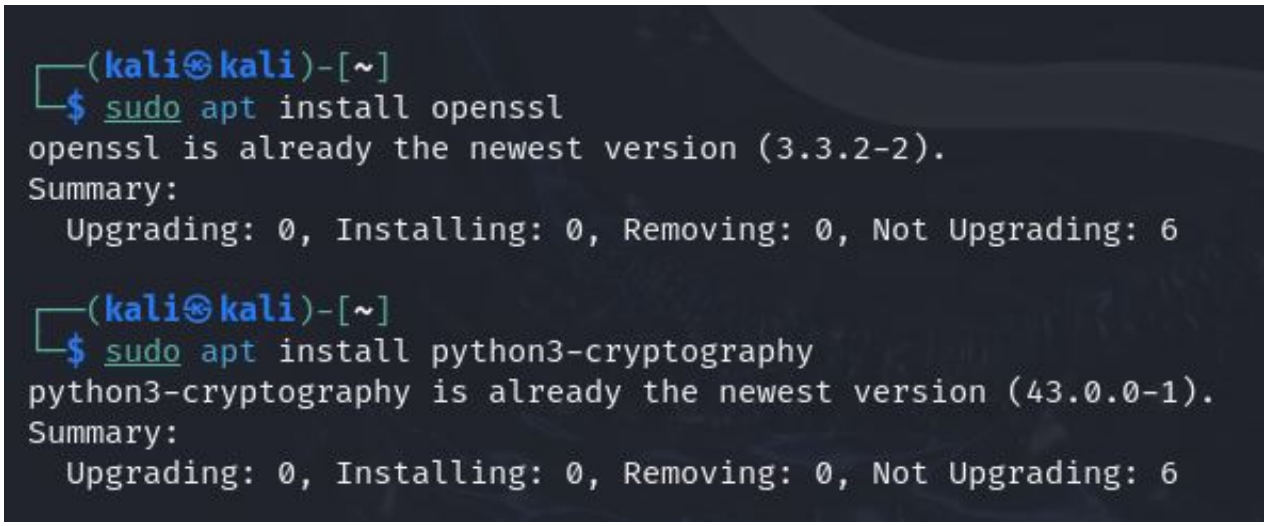
Ho tentato varie chiavi, ho provato varie chiavi fino a che non ho ottenuto il messaggio in chiaro "EPICODE"



Il secondo esercizio invece era sulla criptazione e firmatura con OpenSSL e Python. L'obiettivo era quello di creare uno script per criptare e decriptare messaggi e per firmare e verificare messaggi.

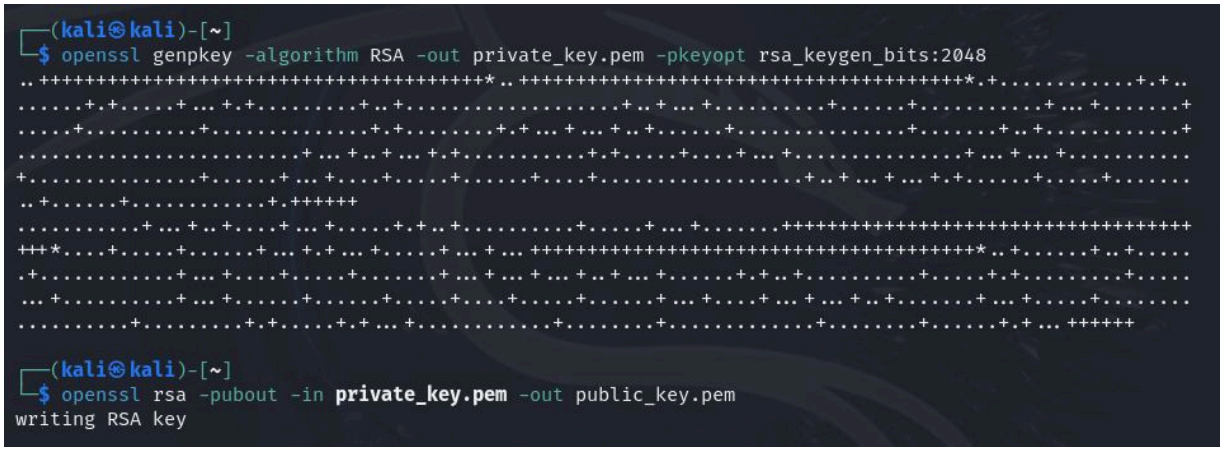
Ho utilizzato **OpenSSL** per la generazione delle chiavi e la libreria **Cryptography** in Python.

Ho verificato che OpenSSL e la libreria pytthon crypthography fossero istallate su macchina virtuale Kali Linux



Ho quindi generato una chiave privata RSA tramite il comando <*openssl genpkey -algorithm RSA -out private_key.pem -pkeyopt rsa_keygen_bits:2048*> ed estratto la

chiave pubblica tramite <*openssl rsa -pubout -in private_key.pem -out public_key.pem*>



Sono passato poi alla creazione del programma in Python. Lo script importa dalla libreria cryptograhy: *padding* (aggiunge dati extra ad un messaggio), *serialization* (per leggere le chiavi private e pubbliche) e *base64* (per mostrare il resto criptato in base64).

Il codice commentato mostra i passaggi per leggere la chiave privata (*private_key*) e la chiave pubblica (*public_key*). Ho inserito un input per permettere all'utente di inserire la frase da criptare e salvato la stringa fornita nella variabile "*message*".
Ho utilizzato le funzioni *encrypt* per criptare il messaggio con la chiave pubblica e *decrypt,* per decriptare con la chiave privata. Infine ho stampato delle stringhe che mostrano il messaggio originale, il messaggio criptato in base64 e il messaggio decriptato.

```python
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.hazmat.primitives import serialization
import base64

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

#Carico la chiave pubblica
with open("public_key.pem", "rb") as key_file:
        public_key = serialization.load_pem_public_key(key_file.read())

message = input("Inserisci il messaggio da criptare: ")

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

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

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

L'ultima richiesta era quella di creare uno script per firmare e verificare i messaggi. Ho dunque creato un nuovo programma, aggiungendo al precedente delle piccole modifiche. Ho importato

hashes dalla libreria cryptography, che permette di creare l'hash del messaggio.
Ho aggiunto il codice per inserire la firma, tramite la chiave privata e la verifica.

```python
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives import serialization
import base64

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

#Carico la chiave pubblica
with open("public_key.pem", "rb") as key_file:
        public_key = serialization.load_pem_public_key(key_file.read())

message = input("Inserisci il messaggio da criptare: ")

#Firmo con la chiave privata
signed = private_key.sign(message.encode(), padding.PKCS1v15(), hashes.SHA256())

#Verifico la firma
try:
        encrypted_b64 = base64.b64encode(signed).decode('utf-8')
        public_key.verify(signed, message.encode(), padding.PKCS1v15(), hashes.SHA256())
        print("Base64 della firma:", encrypted_b64)
        print("Messaggio originale da confrontare:", message)
        print("La firma e' valida")
except Exception as e:
        print("La firma non e' valida", str(e))
#Criptazione con la chiave pubblica
encrypted = public_key.encrypt(message.encode(), padding.PKCS1v15())

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

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

Ho inoltre provato ad avviare entrambi i programmi, inserendo un messaggio da criptare:

```
┌──(kali㉿kali)-[~]
└─$ python encdec.py
Inserisci il messaggio da criptare: Messaggio da criptare in maniera sicura
Messaggio originale: Messaggio da criptare in maniera sicura
Messaggio criptato: PXL/ayXWIhWyTvB3JMct6Udg9BIKZ0EWwwmK3Wb8NHIwmM6CEMSnfKZSNYk6M4Og+sY4L7VeBjCr0BYh1Oq
etdWUAmTKgUn4UDpb5FJrK0xNZRwtzGROZvM9nSBKAyRVUafSsu1EonOYFnpzg6+rQIAQKT1Tq3y44qaeOutE1mvFUR9/R4F7M6zY0v
x7xmQpjKid/lBQ3SW/b1GoOy4Y/C0JQoUgxHpPIunrR1o2zzbSuHmGphkx/ezJq3rVlAJ9j8FbCBge8xvvPIyBtlpx+ngc0QRvUUO8B
XtFG80soGxnAoAw69gVbru5fOQcQ6RwvnJ4iIxxHMTmJmEseN3bsg==
Messaggio decriptato: Messaggio da criptare in maniera sicura

┌──(kali㉿kali)-[~]
└─$ python firma.py
Inserisci il messaggio da criptare: Nuovo messaggio da criptare
Base64 della firma: jtoOa+1rNXBfu+4C2nKzwRMjxvhxHdxXBqCFpL8bFtJSRVcMPMGZpYkNuRIzXgVfxgH5FRo7118OYqZqF/p
+XvOgx5h70pYhQjFviIcJq3Oi6L7bLvc8k3opcdE2pDjVlxGWwhfV2R8Oopz2NNsWGOfWx7wP3evjitRuMnh4ZSDsVJrHiwkuQHGUUz
PwBZNYzHk/7/gO2HZskcbMEezFZ43aDVGqwBwHXBTn3/KgA0n4kgwyk6ds8ENXu8A9fxmKMExY0jVSL4YAkaW4WJyxhrJd1F2z81FY0
1/odO2eKGNTZhXIyImpUjXnthJF5X5doUfrUBKvPmwd4k5Cb4buxw==
Messaggio originale da confrontare: Nuovo messaggio da criptare
La firma e' valida
Messaggio originale: Nuovo messaggio da criptare
Messaggio criptato: PyzZImZJXSP6zaLJ5WmHUojb6OWkjpRKIQqmn9rTteWUxEj2ng90h0iIqPou3wb+KnqYpbixyO0dwIxwGJk
RXdpTxKbPOEjkU+mFOwnMMerio+V6MOs3KciOhJvy58lXIxLwlbtYbAoPJ7uxQlRFkHbs/ZDwP7lmBbmLZIolgzj0LcN6inV4Ei8PZ8
LIMRePoR33cUT0QHB/H06ylSwSB5QSHUjB5DODTjp5qCtoErVPJuPoneLSqYgdyFAIbKmH7M81pL8fraasb3u6ZlHX2/EegqrFbk7rL
XXoLJ13wuuYV38v+eh3jcONPSRYPKcxYL0CEo0e9degU6T7UigF1g==
Messaggio decriptato: Nuovo messaggio da criptare
```