

12/12/2024

Esercizio di oggi

Criptazione e Firmatura con OpenSSL e Python:

Obiettivi dell'esercizio:

- Generare chiavi RSA. Esercizio Traccia e requisiti
- Estrarre la chiave pubblica da chiave privata.
- Criptare e decriptare messaggi.
- Firmare e verificare messaggi.

Strumenti utilizzati:

- OpenSSL per la generazione delle chiavi.
- Libreria cryptography in Python.

L'esercizio ci guidava in questi passaggi:

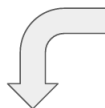
Installazione di OpenSSL:

```
$ sudo apt update  
$ sudo apt install openssl
```



Installazione della libreria per Python:

```
$ sudo apt install python3-pip  
$ pip3 install cryptography
```



Comando per generare la chiave privata RSA:

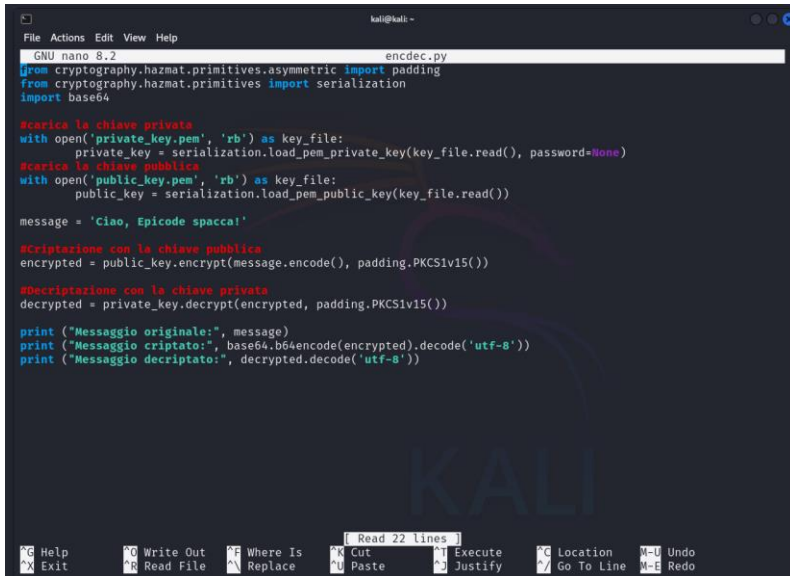
```
$ openssl genpkey -algorithm RSA -out private_key.pem -pkeyopt rsa_keygen_bits:2048
```



Comando per estrarre la chiave pubblica:

```
$ openssl rsa -pubout -in private_key.pem -out public_key.pem
```

Una volta seguiti i passaggi ho creato un file .py di nome encdec e ho scritto al suo interno il codice che ci è stato dato



```
GNU nano 8.2 encdec.py
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.hazmat.primitives import serialization
import base64

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

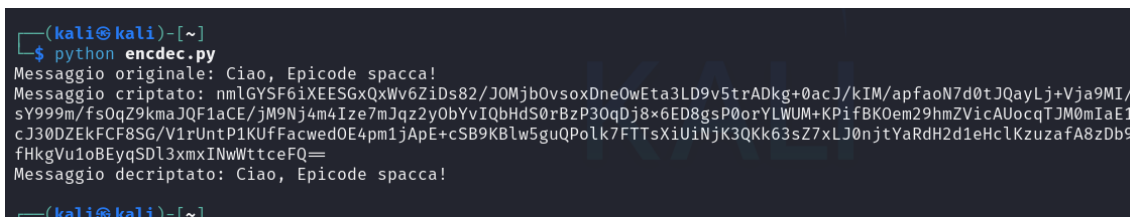
message = 'Ciao, Epicode spacca!'

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

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

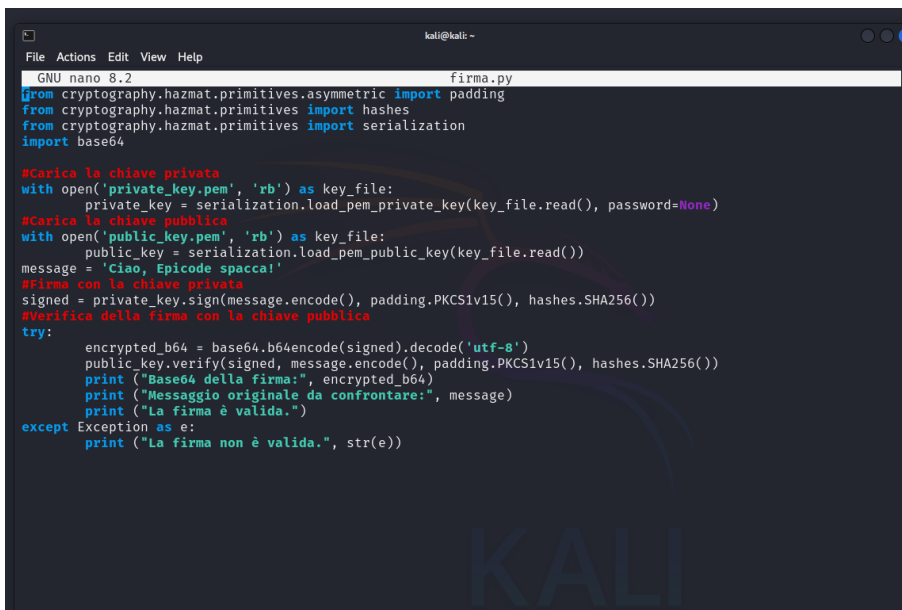
print ("Messaggio originale:", message)
print ("Messaggio criptato:", base64.b64encode(encrypted).decode('utf-8'))
print ("Messaggio decrittato:", decrypted.decode('utf-8'))
```

Ho salvato, sono uscito e ho lanciato il programma con il comando: python nome_file.py



```
(kali@kali)-[~]
$ python encdec.py
Messaggio originale: Ciao, Epicode spacca!
Messaggio criptato: nmlGYSF6iXEEsGxQxWv6ZiDs82/J0Mjb0vsoxDne0wEta3LD9v5trADkg+0acJ/kIM/apfaoN7d0tJQayLj+Vja9MI/
sY999m/fs0qZ9kmaJQF1aCE/jM9Nj4m4Ize7mJqz2y0bYvIQbHds0rBzP30qDj8x6ED8gsP0orYlWUM+KpifBK0em29hmZVicAUocqTJM0mIaE1
cJ30DZEKfCF8SG/V1rUntP1KUfFacwedOE4pm1jApE+cSB9KBlw5guQPolk7FTTsXiUiNjK3QKk63sZ7xLJ0njtYaRdH2d1eHcLkzuzafA8zDb9
fHkgVu1oBEyqSDl3xmxINwWttceFQ==
Messaggio decrittato: Ciao, Epicode spacca!
(kali@kali)-[~]
```

La consegna ci chiedeva di creare un altro file, di nome firma. Ho eseguito gli stessi passaggi di prima e ho scritto il codice



```
GNU nano 8.2 firma.py
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives import serialization
import base64

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

message = 'Ciao, Epicode spacca!'

#Firma con la chiave privata
signed = private_key.sign(message.encode(), padding.PKCS1v15(), hashes.SHA256())
#Verifica della firma con la chiave pubblica
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 è valida.")
except Exception as e:
    print ("La firma non è valida.", str(e))
```

Ho salvato, sono uscito e ho lanciato il programma

```
(kali㉿kali)-[~]  
$ python firma.py  
Base64 della firma: Xo2NZVf6UF/SUYLYA3VxxF0amLZsbaz052PQC0uWFPWtjpUPqi0HjxrUPc0cIDTP5hhsnZ2aPXovG2DvPU4tAgyfzE7  
C15nMGt8RStJrLfqAXeVE1gZ8/aPS+0okAuGTRYjw7zaLuX01ceD8772u00NmBiuRg5PNQYkHScDgKYFkStIB3KW0Y1x5sljWELck3THV93T03Q  
8pfF9wd646nfXjWQNAuk6vg0UK0cCD+Q0mpuDihsv6Wi0sqbRXAWpUc2AViJedSbdVL25DKZwT7HtZqQHGUz87/7Ykm400FsdbC6St+VvNtjOM/  
+mxu2eJ4xwidno3V0VYPA7IOxP7sw=  
Messaggio originale da confrontare: Ciao, Epicode spacca!  
La firma è valida.
```