S3/L4

L'esercizio di oggi riguarda la decriptazione del seguente codice:

QWJhIHZ6b2VidHl2bmdyIHB1ciB6ciBhciBucHBiZXRi

Esercizio Kali opzionale

Dopo aver creato la chiave privata e aver estratto la chiave pubblica abbiamo dovuto creare i 2 programmi il primo per la criptografia e il secondo per la firma digitale.

Criptografia:

```
  GNU nano 8.2
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)

with open ('public_key.pem', 'rb')as key_file:
        public_key=serialization.load_pem_public_key(key_file.read())

message= 'Ciao, Epicode spacca!'
#Criptazione con la chiave pubblica
encrypted = public_key.encrypt(message.encode(), padding.PKCS1v15())
#Decriptazione con la chiave privata
decripted = private_key.decrypt (encrypted, padding.PKCS1v15())

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

Output:

```
┌──(kali㉿kali)-[~/Desktop/python/criptografia e firma]
└─$ python encdec.py
Messaggio originale: Ciao, Epicode spacca!
Messaggio criptato: Uci4AosI634P76b6EFg9kGgSvPz9XO0/zXjCExSstWIcDXK1Ma4QiT6FfXM/K0LdzJvdm7ny9JJ2rYxHGw
+XskUxyxrfrgJvqNrVHRk2ebwg3jj6dL/O/OCjPSAId5ZzHfEIEQNxUpPHLSmitHfHMXMuGzktcsz1CZBQHPJoWmy4esO/8mFcba0
Messaggio decriptato: Ciao, Epicode spacca!
```

Firma:

```
GNU nano 8.2
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))
```

Output:

```
┌──(kali㉿kali)-[~/Desktop/python/criptografia e firma]
└─$ python firma.py
Base64 della firma: acTzX1k5/E7zqTqFRsgWh4eQLZUuhV4xg8+sIyYJXIVJ3q6i5XVfiw8GP2l29H
DmPx+UtawfxEe3wJP2hvtOCSeW7b2+wdCeEsgYIE+ZqsnCZkkP8XR8Udvdw58Sw7GZeQOYWLyaLkFndtaa
Messaggio originale da confrontare: Ciao, Epicode spacca!
La firma è valida.
```