Izvještaj za 3. laboratorijske vježbe

Prvo kreiramo virtualno okruženje sljedećim naredbama:

- python -venv srp
- cd Scripts
- activate
- cd.. 2 puta

i otvaramo prazni VS Code file s naredbom "code."

PRVI IZAZOV

U ovom izazovu prikazali smo na koji način generiramo MAC za poruku uz pomoć ključa te provjeravamo očuvanost integriteta poruke.

Kod:

```
from cryptography.hazmat.primitives import hashes, hmac

def generate_MAC(key, message):

  if not isinstance(message, bytes):

    message = message.encode()

  h = hmac.HMAC(key, hashes.SHA256())

  h.update(message)

  signature = h.finalize()

    return signature

if __name__ == "__main__":

  key=b"my private key"

  with open("message.txt", "rb") as file:

    content = file.read()
```

```
  mac = generate_MAC(key, content)

  print(mac.hex())
```

U terminalu dobijemo ispis našeg MAC (Message Authentication Code).

```
mac = generate_MAC(key, content)

with open("message.sig", "wb") as file:

  file.write(mac)
```

U message.sig file-u zapisujemo našu poruku koju želimo zaštititi

Dodajemo:

```
from cryptography.exceptions import InvalidSignature

def verify_MAC(key, mac, message):

if not isinstance(message, bytes):

message = message.encode()

h = hmac.HMAC(key, hashes.SHA256())

h.update(message)

try:

h.verify(mac)

except InvalidSignature:

return False

else:

return True

with open("message.sig", "rb") as file:

  mac = file.read()

  is_authentic = verify_MAC(key, mac, content)

    print(is_authentic)
```

U terminalu se ispisuje "True". Funkcija vertify_MAC nam govori o očuvanosti integriteta poruke te ukoliko je integritet očuvan u terminalu ispisuje "True". Kada bismo promijenili sadržaj poruke te time narušili integritet poruke, funkcija bi u terminalu ispisala "False".

DRUGI IZAZOV

S lokalnog servera skidamo 10 poruka i 10 MAC-ova pomoću naredbe wget.

Trebamo učitati preuzete file-ove te provjeriti je li porukama narušen integritet. Na izlazu trebamo ispisati sadržaj poruke te provjeriti poklapa li se naš MAC za određenu poruku sa MAC-om kojeg smo preuzeli za tu poruku.

Kod:

```
from cryptography.hazmat.primitives import hashes, hmac

from cryptography.exceptions import InvalidSignature

import os

def verify_MAC(key, mac, message):

  if not isinstance(message, bytes):

    message = message.encode()

  h = hmac.HMAC(key, hashes.SHA256())

  h.update(message)

  try:

    h.verify(mac)

  except InvalidSignature:

    return False

  else:

    return True

if __name__ == "__main__":

  key="mirtakardum".encode()
```

```
  path = os.path.join("challenge", "mirtakardum", "mac_challenge")

  print(path)

  for ctr in range(1, 11):

    msg_filename = f"order_{ctr}.txt"

    sig_filename = f"order_{ctr}.sig"

    print(msg_filename)

    print(sig_filename)

    path_msg = os.path.join(path, msg_filename)

    path_sig = os.path.join(path, sig_filename)

    with open(path + "\\" + msg_filename, "rb") as file:

      content_file = file.read()

    with open(path + "\\" + sig_filename, "rb") as file:

      signature = file.read()

    is_authentic = verify_MAC(key, signature, content_file)

    print(f'Message {content_file.decode():>45} {"OK" if is_authentic else "NOK":<6}')
```

DIGITAL SIGNATURES USING PUBLIC-KEY CRYPTOGRAPHY

U ovom izazovu trebamo odrediti koja od dvije slike je autentična. Iz repozitorija preuzimamo slike, digitalne potpise i javni ključ koji nam je potreban s obzirom da se radi o public-key kriptografiji.

Deseralizacija ključa:

```
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.backends import default_backend

def load_public_key():
  with open(PUBLIC_KEY_FILE, "rb") as f:
    PUBLIC_KEY = serialization.load_pem_public_key(
      f.read(),
```

```
      backend=default_backend()
    )
  return PUBLIC_KEY

print(load_public_key)
```

Naredbom

`print(load_public_key)`

provjeravamo je li ključ uredan.

Provjeravamo ispravnost digitalnog potpisa:

```
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.hazmat.primitives import hashes
from cryptography.exceptions import InvalidSignature

def verify_signature_rsa(signature, message):
    PUBLIC_KEY = load_public_key()
    try:
        PUBLIC_KEY.verify(
            signature,
            message,
            padding.PSS(
                mgf=padding.MGF1(hashes.SHA256()),
                salt_length=padding.PSS.MAX_LENGTH
            ),
        hashes.SHA256()
        )
    except InvalidSignature:
        return False
    else:
        return True
```

Učitavamo sadržaj:

```
with open("image_2.sig", "rb") as file:
    signature = file.read()

with open("image_2.png", "rb") as file:
    image = file.read()
```

Provjeravamo autentičnost:

```
is_authentic = verify_signature_rsa(signature, image)
print(is_authentic)
```