Sigurnost računala i podataka (Lab 3)

Message authentication and integrity

Message Authentication Code (MAC)

IZAZOV 1

Stvorimo virtualno okruženje.

```
C:\Users\A507\kz>python -m venv ka
C:\Users\A507\kz>cd ka
C:\Users\A507\kz\ka>cd Scripts
C:\Users\A507\kz\ka\Scripts>activate
(ka) C:\Users\A507\kz\ka\Scripts>cd ..
(ka) C:\Users\A507\kz\ka\Scripts>pip install cryptography
```

U lokalnom direktoriju kreiramo tekstualnu datoteku odgovarajućeg sadržaja čiji integritet želimo zaštititi te učitamo sadržaj datoteke u memoriju.

```
 with open(filename, "rb") as file:
     content = file.read()
```

Kreiramo funkciju za izračun MAC vrijednosti.

```
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":
pass
```

Dobijemo MAC vrijednost.

```
if name == "main":
key=b"tajna"
with open("message.txt", "rb") as file:
    content=file.read()

mac= generate_MAC(key, content)

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

Zapisujemo generirani MAC u novi file message.sig.

```
with open("./message.sig", "wb") as file:
        file.write(mac)
```

Funkcija za provjeru validnosti MAC-a za danu poruku.

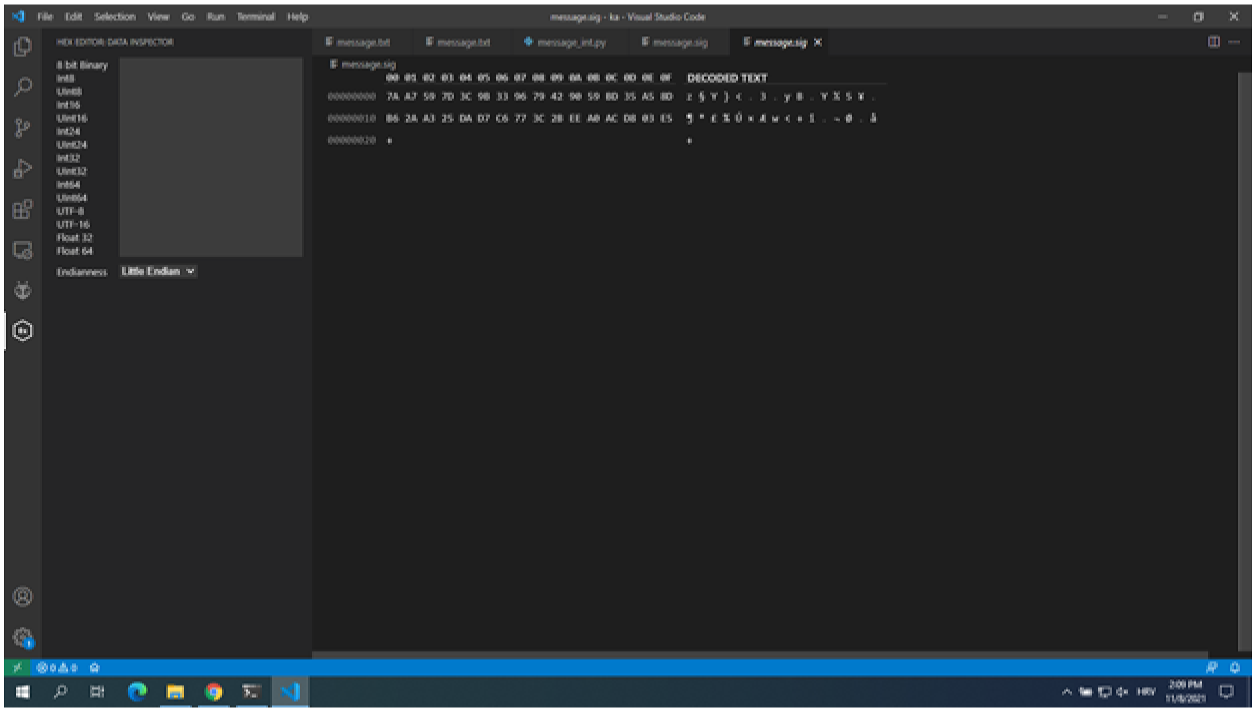
```
from cryptography.hazmat.primitives import hashes, hmac
from cryptography.exceptions import InvalidSignature


def verify_MAC(key, signature, message):
    if not isinstance(message, bytes):
        message = message.encode()

    h = hmac.HMAC(key, hashes.SHA256())
    h.update(message)
    try:
        h.verify(signature)
    except InvalidSignature:
        return False
    else:
        return True
```

Nadogradimo kod te završen glasi ovako.

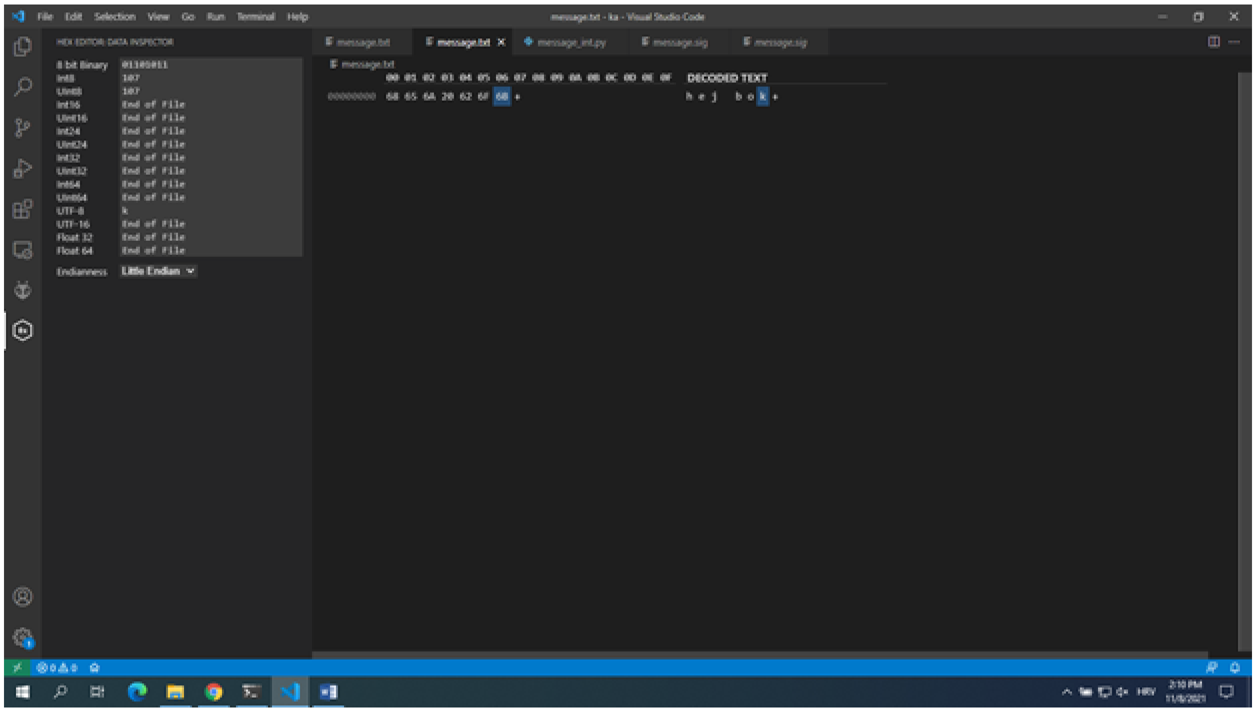
```
from cryptography.hazmat.primitives import hashes, hmac
from cryptography.exceptions import InvalidSignature

def verify_MAC(key, signature, message):
if not isinstance(message, bytes):
message = message.encode()
h = hmac.HMAC(key, hashes.SHA256())
h.update(message)
try:
    h.verify(signature)
except InvalidSignature:
    return False
else:
    return True

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"tajna"
with open("message.txt", "rb") as file:
    content=file.read()

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

is_valid = verify_MAC(key, signature, content)
print(is_valid)
```

Poziv i rezultat funkcije.

```
(ka) C:\Users\A507\kz\ka>python .\message_int.py
True
```



```
message.sig
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F   DECODED TEXT
00000000  7A A7 59 7D 3C 98 33 96 79 42 90 59 BD 35 A5 8D   z § Y } < . 3 . y B . Y ½ 5 ¥ .
00000010  B6 2A A3 25 DA D7 C6 77 3C 2B EE A0 AC D8 03 E5   ¶ * £ % Ú × Æ w < + î . ¬ Ø . å
00000020  +                                                 +
```



```
8 bit Binary  01101011
Int8          107
UInt8         107
Int16         End of File
UInt16        End of File
Int24         End of File
UInt24        End of File
Int32         End of File
UInt32        End of File
Int64         End of File
UInt64        End of File
UTF-8         k
UTF-16        End of File
Float 32      End of File
Float 64      End of File
Endianness    Little Endian
```

```
message.txt
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F   DECODED TEXT
00000000  68 65 6A 20 62 6F 6B +                            h e j   b o k +
```

Za kraj smo mijenjali poruke i potpis te gledali kako se rezultat funkcije mijenja bez obzira što promijenili.

IZAZOV 2

U ovom izazovu **želite utvrditi vremenski ispravnu skevencu transakcija (ispravan redosljed transakcija) sa odgovarajućim dionicama**. Digitalno potpisani (primjenom MAC-a) nalozi za pojedine transakcije nalaze se na lokalnom web poslužitelju.

Sa servera preuzimamo personalizirane izazove
(direktorij zupanovic_karmen/mac_challege).

Preuzmemo sve izazove sa servera prema uputama.

1. Preuzmemo program `wget` dostupan na wget download.
2. Pohranimo ga u direktorij gdje ćemo pisati Python skriptu rješavanje ovog izazova.
3. Osobne izazove preuzimamo izvršavanjem sljedeće naredbe u terminalu:

   `wget.exe -r -nH -np --reject "index.html*" http://a507-server.local/challenges/<zupanovic_karmen>/`

Tajna vrijednost koja se koristi kao ključ u MAC algoritmu.

`key = "<prezime_ime>".encode()`

Ispis poruke i statusa inegriteta 10 poruka.

```
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="zupanovic_karmen".encode()

  path = os.path.join("challenge", "milicevic_mario", "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 trebate odrediti autentičnu sliku (između dvije ponuđene) koju je profesor potpisao svojim privatnim ključem.

Provjera koja je slika autentična korištenjem javnog ključa. U funkciju verify_signature_rsa šaljemo potpis za određenu sliku i sliku.

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

def load_public_key():
with open("public.pem", "rb") as f:
PUBLIC_KEY = serialization.load_pem_public_key(
f.read(),
backend=default_backend()
)
return PUBLIC_KEY

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

#Reading from a file
with open("image_1.png", "rb") as file:
image= file.read()
#Reading from a file
with open("image_1.sig", "rb") as file:
signature= file.read()

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

```
(ka) C:\Users\A507\ka>python .\digital_signature.py
<cryptography.hazmat.backends.openssl.rsa._RSAPublicKey object at 0x00000259685C0D30>
```

Provjerimo za prvu sliku i dobijemo False.

```
(ka) C:\Users\A507\ka>python .\digital_signature.py
False
```

Kod prepravimo za drugu sliku i stavimo potpis za drugu.

```
(ka) C:\Users\A507\ka>python .\digital_signature.py
True
```