Četvrta laboratorijska vjezba

U četvrtoj laboratorijskoj vježbi smo na primjeru demonstrirali korištenje Message authentification code(MAC) mehanizma za zaštitu integriteta poruke.

1) Prvi Zadatak

Kreirali smo novi folder, unutar njega file message.txt u koji smo spremili našu “važnu” poruku.

Unutar istog foldera napravimo python script koji će sadržavati logiku našeg programa.

Cilj je zaštiti autentičnost sadržaja message.txt file-a pomoću primitive MAC funkcije.

Potrebno nam je:

#SIGNING PROCESS

1) Pročitaj sadržaj message.txt file-a

2) Hashirati(MAC primitive function) sadržaj file-a kako bismo dobili potpis(signature).

3) Nadodati potpis na poruku(u ovom primjeru spremamo potpis u odvojenu datoteku).

#VERIFICATION PROCESS

1) Pročitamo sadržaj file-a u kojem je “važna” poruka

2) Pročitamo sadržaj file-a u kojem je signature

3) Pomoću MAC primitive funkcije stvorimo novi potpis od “važne poruke”

4) usporedimo novi potpis sa onim u file-u gdje je originalni potpis

2) Drugi zadatak

U drugom zadatku cilj je bio iz 10 postojećih datoteka i njihovih potpisa, pronaći one kojima je očuvana autentičnost.

Postupak je sličan onome iz prvog zadatka, s time da je to ovdje trebalo napravit 10 puta, za svaku datoteku posebno.

Kopirali smo naše osobne challengeove pomoću wget-a.

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

from pathlib import Path
import re  # regular expression  regularni izrazi
import datetime


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


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)  # sigurna usporedba signature-ova
    except InvalidSignature:
        return False
    else:
        return True


if __name__ == "__main__":

    # zadatak 1
    # Signing process
```

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

# print(content)
key = "my secure secret".encode()
sig = generate_MAC(key, message)

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

# Signing process end

# Verification process

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

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

key = "my secure secret".encode()

is_authentic = verify_MAC(key, sig, message)

print(f"Message is {'OK' if is_authentic else 'NOK'}")

# Verification process end

# zadatak 2

key = "cvitanic_vjeran".encode()

PATH = "challenges/g1/cvitanic_vjeran/mac_challenge/"

messages = []

for ctr in range(1, 11):
    msg_filename = f"order_{ctr}.txt"
    sig_filename = f"order_{ctr}.sig"
    # print(msg_filename)
    # print(sig_filename)

    message_file_path = Path(PATH + msg_filename)
    signature_file_path = Path(PATH + sig_filename)
    with open(message_file_path, "rb") as file:
        message = file.read()

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

    # print(message)

    is_authentic = verify_MAC(key, sig, message)
```

```
        """print(
            f'Message {message.decode():>45} {"OK" if is_authentic else "NOK":<6}')
Kad prominimo poruku (i zakomentiramo generiranje signaturea) dobit cemo NOK poruku
"""
        if is_authentic:
            # decode jer je u binarnom formatu
            messages.append(message.decode())

    messages.sort(key=lambda m: datetime.datetime.fromisoformat(
        re.findall(r'\(.*?\)', m)[0][1:-1]))

    print("Valid messages:\n")
    for msg in messages:
        print(
            f'Message {msg:>45} {"OK":<6}')
```