Sigurnost računala i podataka

Vježba 4: Message authentication and integrity

Cilj vježbe je primijeniti autentikaciju i zaštitu integriteta poruka. Koristili smo simetrični kriptografski mehanizam: message authentication code (MAC) zasnovan na simetričnim ključevima.

Zadatak 1:

Cilj prvog zadatka je zaštita integriteta sadržaja poruke primjenom MAC algoritma. Koristili smo HMAC mehanizam iz Python biblioteke cryptography . Učitali smo poruku čiji integritet želimo zaštiti. Izračunali smo MAC vrijednost za zadani file koristeći funkciju generate_MAC. Onda smo učitali poruku i potpis . Za učitanu poruku smo izračunali MAC vrijednost. Izračunati MAC smo usporedili s učitanim potpisom pomoću verify_MAC funkcije. Ako su MAC-ovi jednaki integritet je očuvan.

Zadatak 2:

Cilj drugog zadatka bio je utvrditi vremenski autentičnu skevencu transakcija dionica. Preuzeli smo niz transakcija i njihovih autentikacijskih kodova. Znali smo da je tajna korištena kao ključ u MAC algoritmu bila u obliku "<prezime_ime>".encode(). Učitavali smo svaku transakciju i njen MAC tag i uspoređivali ih koristeći funkcije generate_MAC i verify_MAC. Na kraju smo pohranili sve autentične poruke u niz messages koji smo onda sortirali po timestampu.

Kod koji smo koristili:

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

from cryptography.hazmat.primitives import hashes,

hmac from cryptography.exceptions import

InvalidSignature 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)
except InvalidSignature:
                    return
False     else:
return True

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

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

key = "my super secure secret".encode() is_authentic
= verify_MAC(key, sig, message)

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

from pathlib import Path import re
import datetime key =
"radovnikovic_tonci".encode()
PATH = "challenges/g1/munivrana_luka/mac_challenge/" messages =
[ ]

for ctr in range(1, 11):
```

```
        msg_filename = f"order_{ctr}.txt"
sig_filename = f"order_{ctr}.sig"

        msg_file_path = Path(PATH + msg_filename)
sig_file_path = Path(PATH + sig_filename)

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

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

        is_authentic = verify_MAC(key, sig, message)
        print(
                f'Message {message.decode():>45} {"OK" if is_authentic else "NOK":<6}')

        if is_authentic:
                messages.append(message.decode())

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

        for m in messages:
                print(f'Message { m:> 45 } { "OK":< 6 }')
```