Sigurnost računala i podataka

Lab 4: Message authentication and integrity

Cilj vježbe je primijeniti teoretske spoznaje o osnovnim kriptografskim mehanizmima za autentikaciju i zaštitu integriteta poruka u praktičnom primjerima. Pri tome smo koristili simetrični kriptografski mehanizam: *message authentication code (MAC)* zasnovan na simetričnim ključevima.

Zadatak 1

Cilj prvog zadatka je bio zaštiti integritet sadržaja poruke primjenom MAC algoritma. Koristili smo HMAC mehanizam iz Python biblioteke `cryptography`.

Prvo smo učitali poruku čiji integritet želimo zaštiti. Izračunali smo MAC vrijednost za zadani file koristeći funkciju generate_MAC i spremili je u zasebni file. Onda smo, da bi provjerili integritet poruke, učitali poruku i potpis. Za učitanu poruku smo izračunali MAC vrijednost ponovno koristeći funkciju generate_MAC. Izračunati MAC smo usporedili s učitanim potpisom pomoću verify_MAC funkcije. Ako su MAC-ovi jednaki poruka nije mijenjana odnosno njen integritet je očuvan.

Zadatak 2

Cilj drugog zadatka bio je utvrditi vremenski ispravnu/autentičnu skevencu transakcija (ispravan redosljed transakcija) dionica. Sa servera smo preuzeli niz transakcija i njihovih autentikacijskih kodova (MAC tagova). Osim toga znali smo da je tajna korištena kao ključ u MAC algoritmu bila u obliku `"<prezime_ime>".encode()`. Redom smo učitavali svaku transakciju i njen MAC tag i uspoređivali ih kao i u 2. zadatku koristeći funkcije generate_MAC i verify_MAC. Na kraju smo, da bi dobili ispravnu sekvencu, pohranili sve autentične poruke u niz messages koji smo onda sortirali po timestampu.

Kod (zadatak 1 i zadatak 2):

GOAL: Protect message authenticity using MAC primitives

Signing process

1. read file

2. sign the msg content using MAC primitives

3. store the signature in a separate file

Verification process

1. read file

2. read signature

3. sign the msg content using MAC primiives

4. compare generated signature with the recieved one

```
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__":
# ================================
# MAC - INTRO
# ================================
# Reading from a file

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

```
#key = "my super secure secret".encode()
#sig = generate_MAC(key, message)

# with open("message.sig", "wb") as file:
# file.write(sig)
# ------------------------------------------------------------
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"}')

# ================================
# MAC CHALLENGE - protecting transactions
# ================================

from pathlib import Path
import re
import datetime
key = "biuk_ivan".encode()
PATH = "challenges/g1/biuk_ivan/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)

    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}')
```