Lab 4: Message authentication and integrity

- zaštita integritea → autentikacija poruka
- 1. zadatak
- U prvom dijelu glumit ćemo nekog tko šalje i prima poruku, a u drugom nekog tko autenticira (iako koristimo message autentication u praksi to svakako zovemo potpisivanje)
- Prvi dio → potpisivanje filea

1. otvoriti file (pročitati sadržaj)
2. hashirati sadržaj kako bi dobili potpis
3. dodati potpis na poruku

- Drugi dio → verifikacija

1. učitajemo sadržaj
2. učitajemo potpis
3. potpisujemo sadržaj
4. uspoređujemo novi potpis sa originalnim

- 2. zadatak
- Odrediti koje su transakcije autenticne te ih po vremenu posložiti (na kraju ispišemao za svaku transakciju je li OK ili NOK)

```
import datetime
import re
from pathlib import Path
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__":

    # # 1. Sign the file content
    # # 1.1  Read the file content
    # with open("message.txt", "rb") as file:
    #     message = file.read()

    # # print(content)
    # # 1.2  Sign the content
    # key = "my super secure secret".encode()
    # signature = generate_MAC(key=key, message=message)
    # # print(signature)
    # # 1.3  Save the signature into a file
    # with open("message.sig", "wb") as file:
    #     file.write(signature)

    # # 2. verify message authenticity
    # # 2.1 Read the recived file
    # with open("message.txt", "rb") as file:
    #     content = file.read()
    # # 2.2 Read the recived signature
    # with open("message.sig", "rb") as file:
    #     signature = file.read()
    # # 2.3.1 Sign the recived file
    # # 2.3.2 Compare locally generated signature with the received one

    # key = "my super secure secret".encode()
    # is_authentic = verify_MAC(key=key, signature=signature, message=content)
    # print(f"Message is {'OK'if is_authentic else 'NOK'}")
```

```
    PATH = "challenges/g2/galiatovic_marija/mac_challenge/"
    KEY = "galiatovic_marija".encode()
    authentic_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)
        with open(msg_file_path, "rb") as file:
            message = file.read()

        sig_file_path = Path(PATH + sig_filename)
        with open(sig_file_path, "rb") as file:
            signature = file.read()

        is_authentic = verify_MAC(
            key=KEY, signature=signature, message=message)
       # print(f'Message {message.decode():>45} {"OK" if is_authentic else "NOK":<6}')
       # ode je rjesena vjezba ali jos mozemo sortirati po vrimenu
        if is_authentic:
            authentic_messages.append(message.decode())

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

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