LAB 3

Message authentication and integrity

- u prvom dijelu vježbe smo napravili funkciju za verifikaciju poruke i zatim pokušali narušiti integritet poruke mijenajući njen sadržaj te zamijetili da u tom slučaju funkcija verify_MAC vraća false
- probali smo promijeniti i signature (hex) no funkcija je isto vratila false

```python
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"my super secret"

        # Reading from a file
        with open("message.txt", "rb") as file:
            content = file.read()

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

        # mac = generate_MAC(key, content)

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

        is_authentic = verify_MAC(key, mac, content)
        print(is_authentic)
```

- u drugom dijelu vježbe smo trebali utvrditi vremenski ispravnu sekvencu transakcija sa odgovarajućim dionicama
- moj kod za rješavanje izazova 2:

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


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 = "celan_dea".encode()
    path = "mac_challenge"
    for ctr in range(1, 11):
        msg_filename = f"order_{ctr}.txt"
        with open(os.path.join(path, msg_filename), "rb") as file:
            message = file.read()

        mac = generate_MAC(key, message)

        sig_filename = f"order_{ctr}.sig"
        with open(os.path.join(path, sig_filename), "rb") as file:
            signature = file.read()
        is_authentic = verify_MAC(key, signature, mac)

        print(f'Message {message.decode():>45} {"OK" if is_authentic else "NOK":<6}')
```