Lab 4:Message authentication and integrity

Cilj vježbe je bio primjeniti teoretske spoznaje o osnovnim kritografskim mehanizmima za autentikaciju i zaštitu integriteta poruka u praktičnom primjerima. Pri tome koristiti simetrične i asimetrične kriptografske mehanizme: *message authentication code (MAC)* i *digitalne potpise*.

1.zadatak

Implementirati zaštitu integriteta sadržaja poruke primjenom odgovarajućeg *message authentication code (MAC)* algoritma. Koristite pri tome HMAC mehanizam iz Python biblioteka cryptography.

Kreirali smo novi folder, unutar njega spremili važnu poruku u file message.txt te unutar tog foldera napravimo python script koji će sadržavati logiku našeg programa.

1.dio: sign the proces

Učitali sadržaj poruke u memoriju zatim koristeći funkciju za izračun Mac vrijednosti nad sadržajem dobili potpis. Te na kraju na potpis nadodali poruku s tim da smo u ovom primjeru spremili potpis u odvojenu datoteku.

2.dio: verification
Pročitali sadržaj file-a u kojem je važna poruka kao i sadržaj filea u kojem je signature. Zatim opet uz pomoć MAC funkcije stvorimo novi potpis od važne poruke i na kraju usporedimo sa onim fileom gdje je signature jesu li isti a trebali bi biti.

kod s naznačenim dijelovima zadataka:

```
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*":

# # 1. Sign the file content

# # 1.1 Read the file content

# # Reading from a file

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

# content = file.read()

# print(content)

# # 1.2 Sign the content

# key = "my super secure secret".encode() #pretvara niz u bajtove tj enodira

# signature = generate_MAC(key = key, message = content) #ako vise puta pozovemo, uvik ce se ist
i generirat jer je ista hash funkcija i kljuc

# 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 received file

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

# 2.2 Read the received signature

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

```
# 2.3.1 Sign the received 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'}")
```

2.zadatak: **Utvrditi vremenski ispravnu/autentičnu skevencu transakcija (ispravan redosljed transakcija) dionicama**

U drugom iz 10 postojećih datoteka i njihovih potpisa trebalo je pronaći one kojima je očuvana autentičnost.
Kopirali smo naše osobne challengeove pomoću wget-a.

`wget.exe -r -nH -np --reject "index.html*" http://challenges.local/challenges/<id_grupe>/<prezime_ime>/mac_challenge/`

Tajna vrijednost koja se koristi kao ključ u MAC algoritmu dobivena je iz vašeg imena:

`key = "<prezime_ime>".encode()`

Te za provjeru datoteka smo odlucili koristiti funkciju kako ne bi morali svaku posebno te smo provjeravali je li autentična te smo ih dodavali u niz te na kraju ih sortirali unutar toga niza.

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

```
#2.zadatak

from pathlib import Path
import re
import datetime
key = "botic_katarina".encode()
PATH = "challenges/g2/botic_katarina/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]))
```