LAB 2 - Symmetric key cryptography - a crypto challenge

Sigurnost računala i podataka

U sklopu vježbe student će riješiti odgovarajući *crypto* izazov, odnosno dešifrirati odgovarajući *ciphertext* u kontekstu simetrične kriptografije. Izazov počiva na činjenici da student nema pristup enkripcijskom ključu.

Uvod

Za pripremu *crypto* izazova, odnosno enkripciju korištena je Python biblioteka `cryptography`. *Plaintext* koji student treba otkriti enkriptiran je korištenjem *high-level* sustava za simetričnu enkripciju iz navedene biblioteke - Fernet.

Na početku vježbe napravili smo python virtualno okruženje koje smo nazvali "env".

```
python -m venv env
```

Iduće smo se pozicionirali u direktorij "/Scripts" i pokrenuli skriptu

```
cd env
```

```
cd Scripts
```

```
activate
```

Pokrenuli smo python shell te ubacili naredbu

- * `from** **cryptography.fernet** **import** Fernet`

Za generiranje ključa koristili smo:

```
key = Fernet.generate_key()
```

Inicijalizacija fernet sustava pomoću danog ključa:

```
f = Fernet(key)
```

Enkripcija nekakvog plaintexta:

```
ciphertext = f.encrypt(plaintext)
```

Dekripcija ciphertexta:

```
deciphertext = f.decrypt(ciphertext)
```

Kada smo se upoznali s osnovnim naredbama fernet sustava mogli smo krenuti rješavati crypto izazov.

Pristupamo serveru http://a507-server.local na kojem se nalaze imena file-ova dobivena enkriptiranjem koristeći SHA-256 algoritam.

U vlastitom direktoriju kreirali smo sljedeću skriptu.

```
code brute_force.py
```

Zalijepili smo slijedeći kod u skriptu

```
from cryptography.hazmat.primitives import hashes

def hash(input):
    if not isinstance(input, bytes):
        input = input.encode()

    digest = hashes.Hash(hashes.SHA256())
    digest.update(input)
    hash = digest.finalize()

    return hash.hex()

filename = hash('prezime_ime') + ".encrypted"
```

Malo smo izmjenili kod i nadodali funkciju:

```
if __name__ == "__main__":
  hash_value = hash("vuko_matej")
  print(hash_value)
```

Vratili smo se u cmd i pokrenuli naš kod

```
python brute_force.py
```

nakon čega se izgeneriralo naše hashirano ime te smo mogli preuzeti izazov koji odgovara našem imenu i spremiti ga u datoteku gdje je brute_force.py .

Za enkripciju smo koristili **ključeve ograničene entropije - 22 bita**. Znamo ciphertext koji želimo dekriptirati te znamo da je plaintext u formatu .png te preostaje nam samo da otkrijemo key tj. ključ.

Unutar brute_force.py stvaramo novu funkciju **brute_force():**

```
def brute_force():
  ctr = 0
  while True:
    key_bytes = ctr.to_bytes(32, "big")
```

```
    key = base64.urlsafe_b64encode(key_bytes)

    if not (ctr + 1) % 1000:
      print(f"[*] Keys tested: {ctr +1:,}", end = "\r")

    ctr += 1
```

Moramo na početku koda dodati base64 sa naredbom `import base64`. Te unutar "**main**" zakomentirati što smo dosad imali te staviti poziv funkcije `brute_force()`

Zasad nam ova funkcija beskonačno mnogo generira ključeve te ispisuje kada generira svako tisućuti ključ.

Konačni kod će nam izgledati ovako:

```
import base64
from cryptography.hazmat.primitives import hashes
from cryptography.fernet import Fernet

def hash(input):
  if not isinstance(input, bytes):
    input = input.encode()
  digest = hashes.Hash(hashes.SHA256())
  digest.update(input)
  hash = digest.finalize()
  return hash.hex()

def test_png(header):
  if header.startswith(b"\211PNG\r\n\032\n"):
    return True

def brute_force():
# Reading from a file
  filename = "hashprezime_ime.encrypted"
  with open(filename, "rb") as file:
    ciphertext = file.read()

# Now do something with the ciphertext
  ctr = 0
  while True:
    key_bytes = ctr.to_bytes(32, "big")
    key = base64.urlsafe_b64encode(key_bytes)
    if not (ctr + 1) % 1000:
      print(f"[*] Keys tested: {ctr +1:,}", end = "\r")
    try:
      plaintext = Fernet(key).decrypt(ciphertext)
      header = plaintext[:32]
      if test_png(header):
        print(f"[+] KEY FOUND: {key}")
        # Writing to a file
        with open("BINGO.png", "wb") as file:
          file.write(plaintext)
            break
```

```
    except Exception:
      pass

    ctr += 1

if __**name__** == "__**main__**":
  brute_force()
```

Potrebno nam je bilo na početku brute_force() funkcije učitati naš file tj. naš ciphertext:

```
filename = "hashprezime_ime.encrypted"
  with open(filename, "rb") as file:
    ciphertext = file.read()
```

Pomoću svakog ključa pokušali smo dekriptirati naš cyphertex

`plaintext =Fernet(key).decrypt(ciphertext)` te znali smo da smo uspjeli ako u headeru tog plaintexta se nalazi dio specifičan za .png fileove.

`header = plaintext[:32]` → za header smo uzimali prva 32 bita plaintexta

Funkcija kojom smo testirali da li je header od .png file izgleda ovako:

```
def test_png(header):
  if header.startswith(b"\211PNG\r\n\032\n"):
    return True
```

Ukoliko nam plaintext tj. header plaintexta prođe ovaj test to znači da smo pronašli naš plaintext te možemo isprintati ključ i spremiti naš plaintext u odvojeni file i možemo prekinuti petlju pomoću naredbe `break`.