Izvještaj laboratorijske vježbe

2. Symmetric key cryptography - a crypto challenge

Zadatak

Riješiti **crypto** izazov dešifrirajući **ciphertext** u kontekstu **simetrične enkripcije**. Izazov počiva na činjenici da nemamo pristup **enkripcijskom ključu**.

Upoznavanje sa bibliotekom "Cryptography" i sustavom "Fernet"

Biblioteka cryptography Pythonova je biblioteka za enkripciju dok je Fernet high-level sustav za simetričnu enkripciju koji koristi low-level kriptografske mehanizme

- AES šifru sa 128 bitnim ključem
- CBC enkripcijski način rada
- HMAC sa 256 bitnim ključem za zaštitu integriteta poruka
- Timestamp za osiguravanje svježine (*freshness*) poruka

- Instalacija cryptography biblioteke

```
$ pip install cryptography
```

- Kratki osvrt na Fernet enkripcijski sustav

```
from cryptography.fernet import Fernet

key = Fernet.generate_key()
f = Fernet(key)

ciphertext = f.encrypt(b"Very secret information!")
print(ciphertext)

plaintext = f.decrypt(ciphertext)
print(plaintext)
```

```
> python brute_force.py
b'gAAAAABhgVD1ufkCvtH3iQLf8royAMZL19Sb5Lv9g50xeFtquBG45EprYXkk7W8eWK9I1PieSbJvtHDFDzvmzOFUjsZvbLlyYeYUDGpVB9kRwiryDODMvzI='
b'Very secret information!'
```

Generirali smo enkripcijski ključ i spremili ga u varijablu **key**. Potom smo stvorili Fernet objekt imena **f** koji će koristiti generirani ključ. U varijblu **ciphertext** spremili smo enkriptirani sadržaj funkcijom **encrypt** koja traži da podaci koji se enkriptiraju, odnosno naša poruka bude tipa **bytes** (slovo b ispred poruke). Na kraju smo u varijablu **plaintext** spremili dekriptirani sadržaj varijable **ciphertext** koristeći funkciju **decrypt**.

Crypto challenge

- Naš crypto challenge rezultat je enkripcije plaintexta korištenjem fernet sustava. Izazov smo skinuli sa internog servera, ali smo prije toga morali odrediti ime personaliziranog izazova funkcijom **hash**, poznavajući obrazac po kojem je ime dodijeljeno ('**prezime**_**ime**').

```
from cyrptography.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()

if __name__ == "__main__":
    h = hash('pijuk_mario')
    print(h)
```

- Za enkripciju smo koristili **ključeve ograničene entropije - 22 bita**. Ključevi su generirani na sljedeći način:

```
 # Encryption keys are 256 bits long and have the following format:
 #
 #              0...000b[1]b[2]...b[22]
 #
```

```
 # where b[i] is a randomly generated bit.
 key = int.from_bytes(os.urandom(32), "big") & int('1'*KEY_ENTROPY, 2)

 # Initialize Fernet with the given encryption key;
 # Fernet expects base64 urlsafe encoded key.
 key_base64 = base64.urlsafe_b64encode(key.to_bytes(32, "big"))
 fernet = Fernet(key_base64)
```

- Upoznali smo se i sa načinom **učitavanja** i **spremanja** datoteka u **Pythonu**

```
# Otvaranje datoteke za citanje
with open(filename, "rb") as file:
    ciphertext = file.read()
    # U ciphertext ucitali smo sadrzaj datoteke imena filename

# Otvaranje datoteke za pisanje
with open(filename, "wb") as file:
    file.write("Hello world!")
```

- Crypto izazov riješili smo koristeći metodu "Grube sile" ("Brute force") gdje smo provjeravali svaki mogući ključ gore navedene entropije od **22 bita**. Bitno za naglasiti je da smo **apriori** znali da je naša **poruka (plaintext)** sadržavala **SLIKU**. U funkciji **"brute_force"** prvo smo učitali enkriptirani sadržaj danog nam file-a u varijablu **ciphertext**. Zatim smo iterirali kroz sve moguće ključeve, odnosno za svaki ključ smo u varijablu **plaintext** spremali dekriptirani sadržaj varijable ciphertext. Da ne bismo zauvijek ostali u while petlji iskoristili smo "trik" na temelju površnog poznavanja plaintexta. Naime, u svakoj iteraciji smo za početni dio poruke (**header**) funkcijom **"test_png"** provjeravali postoji li dio sa ekstenzijom **PNG** što je uobičajeno kada je sadržaj poruke slika. Sada je jasno da smo pronašli **pravi ključ** i **izvornu poruku** kada funkcija "test_png" vrati vrijednost **"True"**. Na kraju se izvorna poruka spremi u datoteku imena "BINGO.png".

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

def brute_force():

    filename = "f8227977...encrypted"
    with open(filename, "rb") as file:
        ciphertext = file.read()

    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}")

                 with open("BINGO.png", "wb") as file:
                     file.write(plaintext)
```

```
                break
        except Exception:

        ctr += 1

if __name__ == "__main__":
    brute_force()
```

Zaključak

U danoj laboratorijskoj vježbi upoznali smo se sa kriptografskim mehanizmom **simetrične enkripcije**. Nakon što smo riješili crypto izazov metodom **"Grube sile"** zaključujemo da možemo "probiti" bilo koji ključ, ako ne vodimo računa o **entropiji ključa** i **njegovoj veličini**. S druge strane, ako "napadač" zna enkripcijski i dekripcijski algoritam (**E** i **D**) kao i ciphertext (**C**) on nije u mogućnosti dekriptirati ciphertext kao ni pronaći ključ (**K**). Zapravo, naša izvorna poruka (**P**) je sigurna čak i ako "napadač" zna sve osim ključa. **Dakle, za visoku razinu sigurnosti moramo koristiti snažan enkripcijski algoritam i čuvati naš ključ na sigurnom!**