

Vježba 2 - simetrična kriptografija

Cilj vježbe je bio primijeniti teorijska znanja vezana uz simetrične kriptosustave da bi se, služeći se njima, dektirprala enkriptirana datoteka. Programski kod potreban za izvođenje vježbe pisan je u Pythonu, a preduvjet za uspješno izvođenje vježbe jest poznavati osnovne dijelove simetričnog kriptosustava, a to su:

- **Plaintext** - poruka (podatak) koji enkriptiramo
- **Enkripcijski i dekripcijski algoritam** - algoritam koji se koristi za enkriptiranje i dektiptiranje podataka.
- **Tajni ključ** - ključ koji utječe na to kako će izgledati enkriptirani podatak. Kod simetričnih sustava se isti ključ koristi za enkriptiranje i dekriptiranje, dakle obje strane u komunikaciji koriste isti , naravno samo njima poznat, tajni, ključ. Bez ključa enkriptiranje i dektiptiranje nije moguće.
- **Ciphertext** - enkriptirani podatak

Prije dekriptiranja, trebalo je pronaći sebi namijenjenu datoteku pohranjenu na lokalnom poslužitelju čiji je naziv vrijednost koju daje kriptografska hash funkcija kad kao argument dobije određeni tekst tj. ulaz. Za to je bio potreban

```
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()

if __name__ == "__main__":
    filename = hash("cagalj_josip") + ".encrypted"
    print(filename)

    if not path.exists(filename):
        with open(filename, "wb") as file:
            file.write(b"")
```

Dakle, ime datoteke jest ono što se dobije kada se hash funkciji kao argument pošalje “cagalj_josip” i na hash vrijednost koju funkcija dađe za taj argument doda “.encrypted”.

Po uspješnom otkrivanju sebi namijenjene datoteke, ista se stvara u radnom direktoriju i u nju se kopira sadržaj iz tražene datoteke na poslužitelju koji predstavlja chipertext.

S obzirom da za dekriptiranje treba znati ključ kojim je datoteka enkriptirana, a spomenuti nije poznat postoje, općenito govoreći, dvije mogućnosti:

- Kriptoanaliza, koja u ovom slučaju nije od pomoći jer je za enkripciju korišten napredni servis Fernet (temeljen na AES-u)
- Napad korištenjem “grube sile” (eng. brute force attack) koji je primijenjen.
 - Izvodi se tako što se chipertext pokušava dekriptirati korištenjem mogućeg ključa u nadi da će rezultat biti originalni podatak (plaintext).
 - Ukoliko korišteni ključ nije ispravan, postupak se iterativno ponavlja dok se ne otkrije odgovarajući ključ (u našem se slučaju u svakoj iteraciji ključ uvećava za 1).
 - Brute force napad će, u teoriji, uvijek rezultirati pronalaskom odgovarajućeg ključa, ali ključevi koji se u realnim sustavima koriste za enkripciju imaju veliku entropiju i njihova duljina je tolika (toliko je mogućih kombinacija) da brute force napad nije izvediv u prihvatljivom vremenu.

Iako ima spomenutu veliku manu, u našem slučaju je brute force napad dobar izbor jer je datoteka enkriptirana uz korištenje ključa duljine od samo 22 bita (u praksi se koriste ključevi od, npr., 128, 192 ili 256 bita). Zahvaljujući tako kratkom ključu, računalo može u relativno kratkom vremenskom intervalu pronaći odgovarajući ključ. Kod čijim se interpretiranjem izvodi brute force napad je:

```
from cryptography.fernet import Fernet, InvalidToken
def brute_force_attack(chipertext):
    ctr = 0
    while True:
        key_bytes = ctr.to_bytes(32, "big")
        key = base64.urlsafe_b64encode(key_bytes)
        try:
            plaintext = Fernet(key).decrypt(chipertext)
            header = plaintext[:32]
            if test_png(header):
```

```

        print(f"Kljuc nadjen: {key}")
        with open("BINGO.png", "wb") as file:
            file.write(plaintext)
            break
    except InvalidToken:
        pass
    ctr += 1

```

Da bismo znali je li korišteni ključ traženi, treba utvrditi ima li plaintext dobiven dekriptiranjem chipertexta korištenjem korištenog ključa, smisla, odnosno je li plaintext onog oblika koji očekujemo. Da bismo to provjerili moramo znati što pokušavamo dekriptirati. U našem slučaju riječ je bila o PNG slici pa je za provjeru "smislenosti" plaintexta korištena funkcija `test_png` koja provjerava započinje li plaintext upravo onim bitovima kojima započinje PNG datoteka. Njezin kod je:

```

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

```

Konačno, sav kod korišten na vježbi je:

```

import base64
from cryptography.hazmat.primitives import hashes
from cryptography.fernet import Fernet, InvalidToken
from os import path

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
    return False

def brute_force_attack(chipertext):
    ctr = 0
    while True:
        key_bytes = ctr.to_bytes(32, "big")
        key = base64.urlsafe_b64encode(key_bytes)

```

```

try:
    plaintext = Fernet(key).decrypt(chipertext)
    header = plaintext[:32]
    if test_png(header):
        print(f"Kljuc nadjen: {key}")
        with open("BINGO.png", "wb") as file:
            file.write(plaintext)
        break

except InvalidToken:
    pass

ctr += 1

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

if name == "main":
    filename = hash("cagalj_josip") + ".encrypted"
    print(filename)


    if not path.exists(filename):
        with open(filename, "wb") as file:
            file.write(b"")

    with open(filename, "rb") as file:
        encrypted_challenge = file.read()

    brute_force_attack(encrypted_challenge)

```

Rezultat dekriptiranja moje datoteke je bila ova PNG slika:



Congratulations Cagalj Josip!

You made it!