SRP lab 3

U trećem labu smo rješavali crypto challenge, trebali smo dešifrirati odgovarajući chipertext iako nemamo pristup odgovarajućem enkripcijskom ključu.

Svatko je imao svoj personalizirani izazov te da bi mogli koristiti svoj trebali smo enkriptirati naše ime i pronaći odgovarajući izazov, a to smo napravili pomoću ovog dijela koda:

```
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("krizanac_stela") + ".encrypted"
print(filename)
```

Na početku smo znali kojim chipertextom raspolažemo te smo također znali da je datoteka slika u PNG formatu, odnosno znali smo prvih 8 byteova plaintexta. Za uspješan napad nam je trebao još enkrpicijski ključ kojeg smo dobili korištenjem brute-force napada.

Na početku smo stavili da je counter 0 odnosno krenuli smo od nultog ključa i provjeravali hoćemo dobiti file PNG formata dekripcijom. Ukoliko je ključ neispravan counter bi se povećao i tako dok ne pronađemo odgovarajući. Mogućih ključeva je bilo 2^22, a rezultat smo mogli očekivati nakon što je probano otprilike pola mogućih ključeva. Ukoliko je broj probanih ključeva prešao ukupan broj, to naravno znači da imamo neku pogrešku. Ako je pronađen odgovarajući ključ i ako je korišten ispravni osobni izazov, u mapi je stvorena datoteka BINGO.png koja se nalazi na dnu izvještaja koja je i rezultat izvršenog brute-force napada.

Korišteni kod:

```
import base64
from cryptography.fernet import Fernet
from os import path
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()


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


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

        # Now initialize the Fernet system with the given key
        # and try to decrypt your challenge.
        # Think, how do you know that the key tested is the correct key
        # (i.e., how do you break out of this infinite loop)?
        try:
            plaintext = Fernet(key).decrypt(ciphertext)
            header = plaintext[:32]

            if test_png(header):
                print(f"BINGO: {key}")
                with open("BINGO.png", "wb") as file:
                    file.write(plaintext)
                break
        except Exception:
            pass
        ctr += 1
        if not ctr % 1000:
            print(f"[*] Keys tested: {ctr:,}", end="\r")
```

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

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

    # Create a file with the filename if it does not already exists
    if not path.exists(filename):
        with open(filename, "wb") as file:
            file.write(b"")

    # Open your challenge file

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

    # print(ciphertext)
    brute_force(ciphertext)
```

Rezultat: