Lab 3: Symmetric key cryptography (a crypto challenge)

- Prvo smo hash-ali nase ime kako bi ga nasli medu file-ovima (hashiramo jer je nemoguce odhashirat)
- Stvorili smo file sa nasim hashiranim imenom i tu zalijepili sve ono sa stranice s nasim imenom
- Challange je eknriptiran simetricnim kljucem i za dekriptirati nam treba kljuc (sa fernetom je enkriptirano i generiran je kljuc)
- Entropija kljuca je 22 bita (generiran je tako da je vecina pocetnih bitova 0 a zadnja 22 su generirana slucajno)
- Kako rjesit izazov? —> pogadati 22 bita (pretpostavit cemo neki kljuc, dekriptirat i vidjeti ima li to rjesenje smisla)
- Ako nasumicno pogadamo kljuc moze se dogoditi da testiramo isti kljuc vise puta —> treba prolaziti po kljucevima po redu

```
if sam_dobio_nesto_smisleno(plaintext):
  print(key)
  save(plaintext)
  break;
//Ako sam dobio nesto smisleno isprintaj key i spremi dekriptirani challange i onda izadi iz petlje
```

- Decrypted challenge mora biti nesto definirano prije funkcije
  `plaintext = Fernet(key).decrypt(ciphertext)` //ciphertext je nas ovaj challenge
- Odakle nam ciphertext? —> proslijedit cemo ga bruteforce-u kao varijablu
- Moramo imati ideju sto je eknriptirano
  npr. ako je enkriptiran tekst onda ce u tekstu biti rijeci iz tog jezika (rijec “je” je los primjer jer se lako nasumicno moze poklopiti —> moraju biti sto duze jer se smanjuje vjerojatnost da ce se slucajno poklopiti)
- Enkriptirana je slika u png formatu
- Kako editori slika znaju —> postoje neki karakteristicni podatci za npr. png slike (https://en.wikipedia.org/wiki/Portable_Network_Graphics)

- Testiramo je li enkriptirana slika —> poslat cemo header iz dekriptiranog filea te vidjeti poklapa li se s informacijama na Wikipediji da vidimo poklapaju li se byteovi
- Kad smo pokrenili imali smo exception na decrypt —> pokusavamo decryptirat ispravan cipher s neispravnim kljucem
- Kako sprijeciti prekid koda tj. da se nastavi izvrsavat —> `try catch`

```
import base64
from os import path
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"\221PNG\r\n\032\n"):
        return True
    return False


def brute_force(input):
    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")
```

```
if __name__ == "__main__":
    filename = hash('galiatovic_marija') + ".encrypted"

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


# open your challenge file and read your challange
with open(filename, "rb") as file:
    ciphertext = file.read()

# provjera eli radi
# print(ciphertext)
# Start the attack

brute_force(ciphertext)
```