LAB 3:Symmetric key cryptography (a crypto challenge)

U ovoj vježbi smo dešifrirali odgovarajući ciphertext bez pristupa enkripcijskom ključu pomoću brute force-napada.

Prvo se izgeneriralo naše hashirano ime te smo mogli preuzeti izazov koji odgovara našem imenu i spremili smo ga u datoteku.

Za enkripciju smo koristili ključeve ograničene entropije od 22 bita(2^22 kombinacija ključeva). S obzirom da se radilo o relativno malom broju, koristili smo brute-force napad.
Decrypted_challange smo definirali kao fermet(key).decrypt(ciphertext). Taj ciphertext brute-force napadu prosljeđujemo kao varijablu.

Enkriptirana je slika u png formatu.

Kad smo pokrenili imali smo exception na decrypt što znači da smo pokušavali decryptirat ispravan chiper s neispravnim ključem. Prekid koda smo spriječili pomoću try-catch.

```
import base64
from cryptography.hazmat.primitives import hashes
from os import path
from cryptography.fernet import Fernet, InvalidToken
import base64
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(ciphertext):
    ctr=0
    while True:
        key_bytes = ctr.to_bytes(32, "big")
        key = base64.urlsafe_b64encode(key_bytes)

        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 InvalidToken:
            pass

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


if __name__=="__main__":
    filename = hash('delic_karla') + ".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)import base64
from cryptography.hazmat.primitives import hashes
from os import path
from cryptography.fernet import Fernet, InvalidToken
import base64
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(ciphertext):
    ctr=0
    while True:
        key_bytes = ctr.to_bytes(32, "big")
        key = base64.urlsafe_b64encode(key_bytes)

        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 InvalidToken:
            pass

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


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