

Lab 3: Symmetric key cryptography (a crypto challenge)

- Prvo smo hash-ali nase ime kako bi ga nasli među file-ovima (hashiramo jer je nemoguće odhashirati)
- Stvorili smo file sa nasin hashiranin imenom i tu zalpili sve ono sa stranice sa nasin imenom
- Challenge enkriptiran simetricnin kljucen i za dekriptirat nan triba kjuc (sa ferneton 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 i dekriptirat i vidit el to rjesenje ima smisla)
- Ako nasumicno pogadamo kljuc onda ce se dogodit da testiramo isti kljuc vise puta —> triba prolazit po kljucevima po redu
- Ako sam dobio nesto smisljeno isprintaj key i spremi dekriptirani challenge i onda izadi iz petlje

```
if sam_dobio_nesto_smisljeno(plaintext):  
    print(key)  
    save(plaintext)  
    break;
```

- Decrypted challenge mora biti nesto definirano prije funkcije
`plaintext = Fernet(key).decrypt(ciphertext)` //ciphertext je nas ovaj challenge
- Odakle nam ciphertext? —> prosljedit cemo ga bruteforce-u kao varijablu
- Moramo imati ideju sto je enkriptirano
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 karakteristici podatci za npr. png slike (https://en.wikipedia.org/wiki/Portable_Network_Graphics)
- Testiramo je li slika —> poslati cemo header iz dekriptiranog filea te viditi poklapa li se sa 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 spriječiti prekid koda tj. da se nastavi izvorsavati —> `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"\211PNG\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 challenge
    with open(filename, "rb") as file:
        ciphertext = file.read()

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

    brute_force(ciphertext)

```