na ovim lab vježbama smo pokušali enkriptirani text dekriptirat pomoću brute force napada

prvo hashiramo svoje ime kako bi nasli medu ponudenim fileovima nase ime(hashiramo jer je nemoguce odhashirat)

stvorili smo file sa nasin hashiranin imenon i tu zalpili sve ono sa stranice sa nasin imenon

imali smo challange eknriptiran simetricnin kljucen i za dekriptirat nan triba kjuc

Entropija kljuca je 22 bita (vecina pocetnih bitova je 0 ,zadnja 22 su generirana). Problem je pogoditi 22 bita pa cemo mi pretpostavit neki kljuc i vidit ima li to ikakvog smisla.Ako ne uspijemo pronaci key onda prelazimo na drui po principu prvo krcemo sa 1 pa ako nije on idemo dalje na 2 a na 3 itd. , ovo radimo jer postoji mogucnost da jedan te isti kljuc vise puta testiramo

beskonacna petlja se sama nece zaustviti pa je moramo mi sami prisilno zaustavit, kostur trenutno radi ali ne bas onako kako mi zelimo, mi zelimo da prolazi kroz sve kljuce dok ne pronade onaj koji nama treba

pseudo kod bi glasio:

**ako si nasa nesto smisleno, isprintaj taj kljuc, spremi decrypted\_challange i break**

* decrypted\_challange moramo dedfinirati kao fermet(key).decrypt(ciphertext) -ovaj ciphertext brute force napadu proljedujemo ko varijablu
* beskonacnu petlju prekidamo kada pronademo kljuc,a tome nan pomaze funkcija koja ce nam rec jel to sta smo dobili ima smisla,ovo je moguce jedino ako imamo otprilike ideju o cemu je rijec u enkriptiranom tekstu, kako je ovdje rijec o slici u png formatuto bi bilo da pocinje sa 211png, ako sa tim ne pocinje onda nije ispravno pa se ide na drugi,kada nademo pravi kljuc zelimo ga isprintati
* sliku cemo sacuvati u file pomocu bingo.png i taj file zelimo napisati plaintext sta je zapravo nase ime, ali javia nan se neki problem(exception); invalid token, to se javlja zbog toga sta zelimo dekriptirat neispravnim kljucem, ovo cemo rjesit tako da ubacimo try i except
* sada je iteracija beskonacne petlje puno sporija(dekripcija placa vremenom), neki ce od nas nac kljuc prije od drugih jer svi imamo razlicite kljuce
* na bingo.png ce izaci slika sa okvirem u kojem pise nase ime
* tvoj je neki oko 2 miljona

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('belamaric\_gina\_aurora') + ".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('belamaric\_gina\_aurora') + ".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)

