import cv2

import os

import numpy as np

def xor\_encrypt\_decrypt(text, key):

"""Encrypt or decrypt text using a repeating XOR key."""

return ''.join(chr(ord(text[i]) ^ ord(key[i % len(key)])) for i in range(len(text)))

def encode\_message(img\_path, output\_path, msg, password):

img = cv2.imread(img\_path)

if img is None:

print("Error: Image not found!")

return

height, width, \_ = img.shape

max\_chars = height \* width - 2 # Reserve 2 pixels for length storage

if len(msg) > max\_chars:

print(f"Message too long! Max length is {max\_chars} characters.")

return

# Encrypt the message using XOR encryption

encrypted\_msg = xor\_encrypt\_decrypt(msg, password)

msg\_length = len(encrypted\_msg)

# Store message length in the first pixel (avoiding uint8 overflow)

img[0, 0, 0] = np.uint8(msg\_length // 256) # High byte

img[0, 0, 1] = np.uint8(msg\_length % 256) # Low byte

# Store encrypted message in pixels

n, m = 0, 2 # Start after length storage

for i in range(msg\_length):

img[n, m, 0] = np.uint8(ord(encrypted\_msg[i])) # Store in blue channel only

m += 1

if m >= width:

n += 1

m = 0

cv2.imwrite(output\_path, img)

print("Message encrypted successfully!")

os.system(f"start {output\_path}")

def decode\_message(img\_path, password):

img = cv2.imread(img\_path)

if img is None:

print("Error: Image not found!")

return

# Retrieve message length from the first pixel (convert to int explicitly)

length = (int(img[0, 0, 0]) \* 256) + int(img[0, 0, 1])

encrypted\_message = ""

n, m = 0, 2 # Start after length storage

for \_ in range(length):

encrypted\_message += chr(int(img[n, m, 0])) # Ensure correct integer conversion

m += 1

if m >= img.shape[1]:

n += 1

m = 0

# Decrypt message

decrypted\_msg = xor\_encrypt\_decrypt(encrypted\_message, password)

print("Decrypted message:", decrypted\_msg)

# Example Usage

img\_file = "mypic.jpg"

output\_file = "Encryptedmsg.png"

secret\_msg = input("Enter secret message: ")

password = input("Enter password: ")

encode\_message(img\_file, output\_file, secret\_msg, password)

# Decryption

user\_pass = input("Enter passcode for Decryption: ")

decode\_message(output\_file, user\_pass)