Pixel Manipulation for Image Encryption in Python

I've recently completed a project focused on image encryption using pixel manipulation techniques in Python. This project leverages the Pillow and NumPy libraries to encrypt and decrypt images, showcasing my skills in image processing and cryptography.

Code:

```
from PIL import Image
import numpy as np

def encrypt_image(input_image_path, output_image_path, key):
    img = Image.open(input_image_path)
    img_array = np.array(img)
    encrypted_array = np.bitwise_xor(img_array, key)
    encrypted_img = Image.fromarray(encrypted_array)
    encrypted_img.save(output_image_path)
    print("Image encrypted successfully.")
    return encrypted_array

def decrypt_image(encrypted_array, output_image_path, key):
    decrypted_array = np.bitwise_xor(encrypted_array, key)
    decrypted_img = Image.fromarray(decrypted_array)
    decrypted_img.save(output_image_path)
    print("Image decrypted successfully.")

if __name__ == "__main__":
    input_image_path = r"C:\Users\hp\OneDrive\Pictures\Spidy.jpg"
    img = Image.open(input_image_path)
    width, height = img.size
    key = np.random.randint(0, 256, size=(height, width, 3), dtype=np.uint8)  # 3 for RGB channels
    encrypted_image_path = r"C:\Users\hp\OneDrive\Pictures\Spidy2.jpg"
    encrypted_array = encrypt_image(input_image_path, encrypted_image_path, key)
    decrypted_image_path = r"C:\Users\hp\OneDrive\Pictures\Spidy3.jpg"
    decrypt_image(encrypted_array, decrypted_image_path, key)
```

Sample Outputs:

IMAGE SPIDY:

IMAGE SPIDY2 (ENCRYPTED IMAGE):

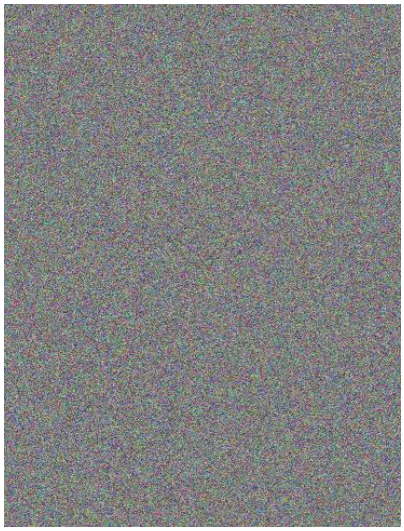

IMAGE SPIDY3 (DECRYPTED IMAGE):