基于混沌系统的图像加密方案

一． 作品概述

该作品首先使用按位异或操作实现对图片的加密和解密处理，然后设计了一个图形操作界面来让用户使用这个对图片的加密和解密的程序，然后分析了这种算法选择明文攻击和唯密文攻击的安全性，随后分析了当前图像加密的研究现状，最后结合混沌系统提出一种新的图像加密算法。

二． 作品设计与实现

2.1 加密与解密原理

按位异或也被称为半加计算，其运算规则如下：

算数1	算数2	结果	python代码
0	0	0	0^0
0	1	1	0^1
1	0	1	1^0
1	1	0	1^1

简单的概括按位异或运算的规则是：相同的两个数运算为 0，不同两个数运算为 1。

按位异或的方法对图像加密和解密的流程为，通过对原始图像和密钥图像进行按位异或运算，得到加密后的图像，然后根据对称性，将加密后的图像与密钥进行按位异或运算，则可以得到原图像。规定异或的符号为 xor，假设：xor(a,b) = c，则有 xor(c,b) = a, xor(c,a) = b,因此，我们假设 a 为图像，b 为密钥，则通过 xor(a,b)计算出的 c 就是得到的密文。

在这个方案中，我们把图片根据 RGB 像素通道变成 500X500 的矩阵格式，每一个像素点都在 0-255 之间，然后我们通过输入密钥，将密钥设置为随

机数生成种子，从而得到一个 500X500 的随机数在 0-255 之间的矩阵，根据伪随机数的特性，只要输入的是相同的数字作为密钥，则生成的随机数矩阵是固定的，那么我们可以得到根据学号为密钥的 xor 加解密算法。

混沌系统 henon 映射和按位异或相结合的加密算法：

在了解了目前最新的图像加密研究方向后，我想到了将混沌系统与按位异或相结合的一种加密方案。首先解释一下 henon 映射，henon 映射是一种二维非线性动力系统，由以下迭代方程定义：

$x(n+1) = y(n) + 1 - a * x(n)^2$

$y(n+1) = b * x(n)$

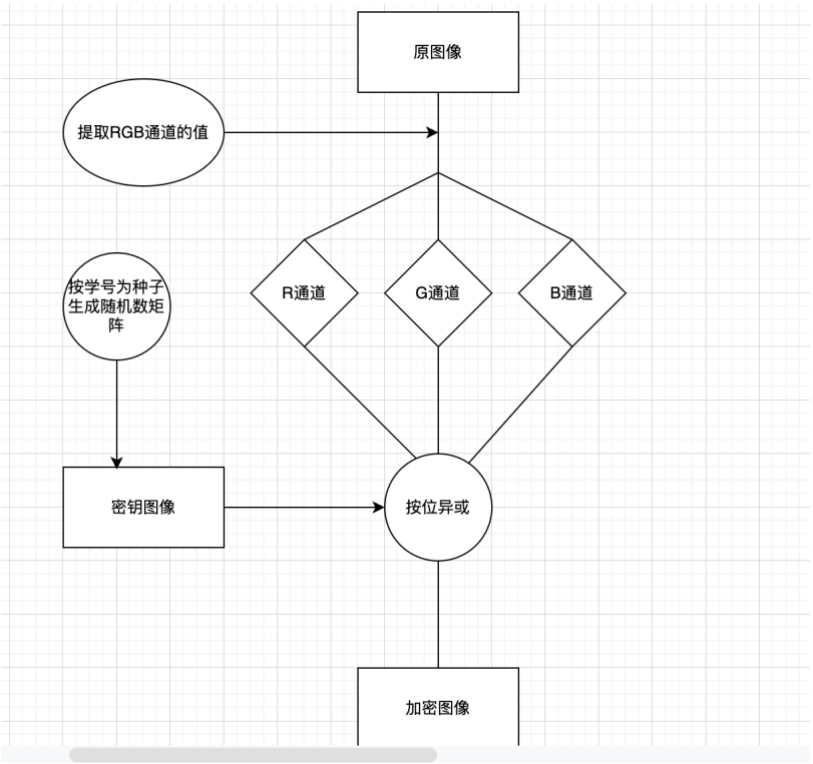
其中，x(n),y(n)是当前的状态，x(n+1),y(n+1)是下一个状态，a 和 b 是映射的参数，通过多轮次的迭代从而产生复杂的，看似随机的混沌表。

方案即为，在原来的按位异或的基础上，设计了一个 henon_map 函数，通过迭代计算 x 和 y 的值，并将它们相加取模 256，从而产生一个混沌表，然后在原来 xor 得到的加密图像 rgb 矩阵的基础上和混沌表相互作用(演示算法中直接相加)，从而得到最终的加密图像。解密的过程同理。

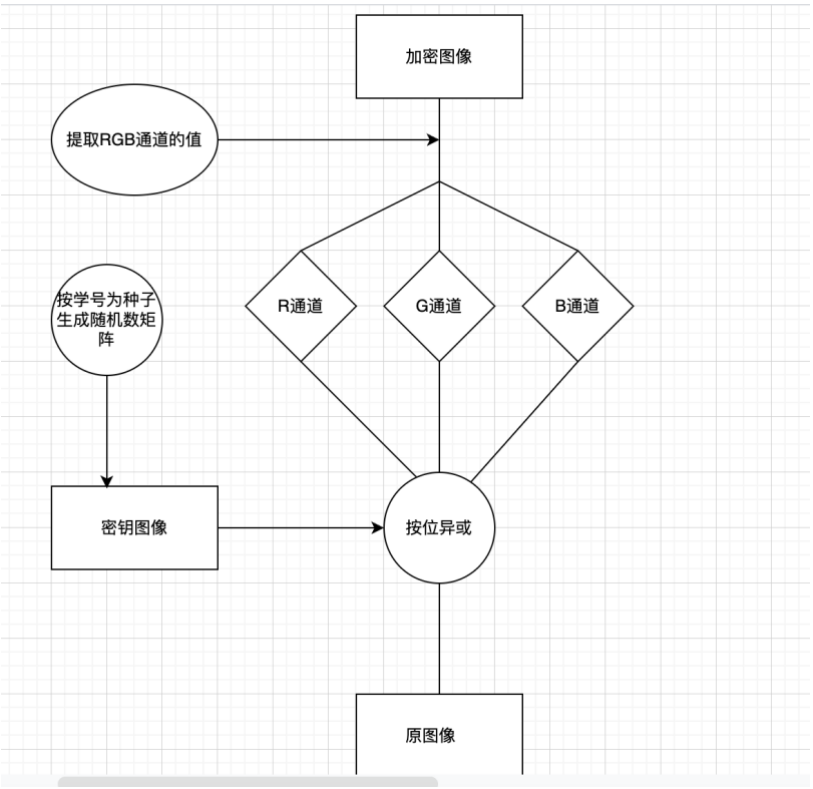
2.2 方案流程图

按位异或：

加密：

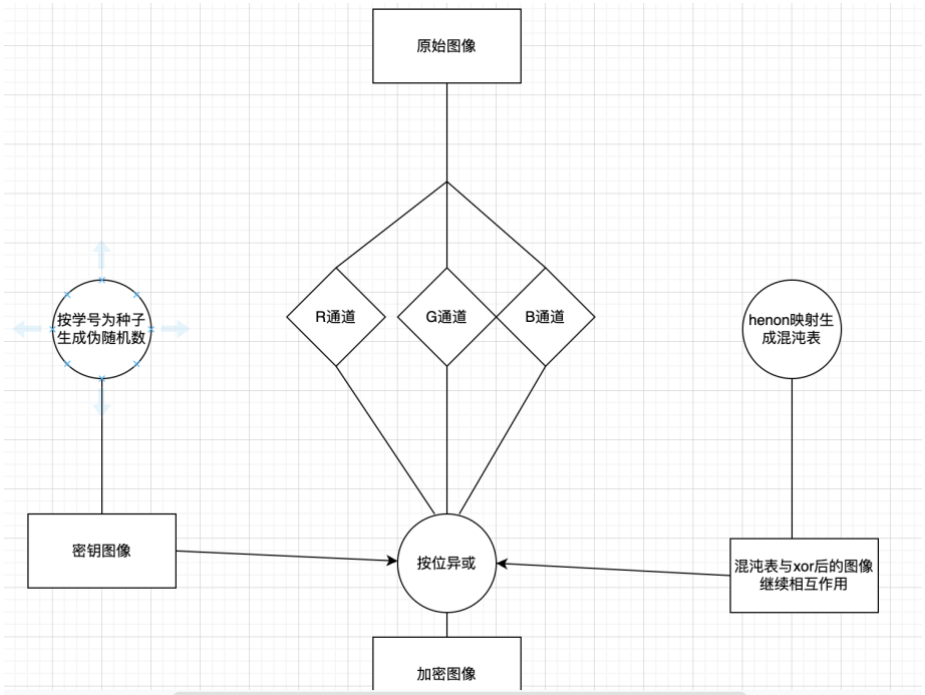


解密:

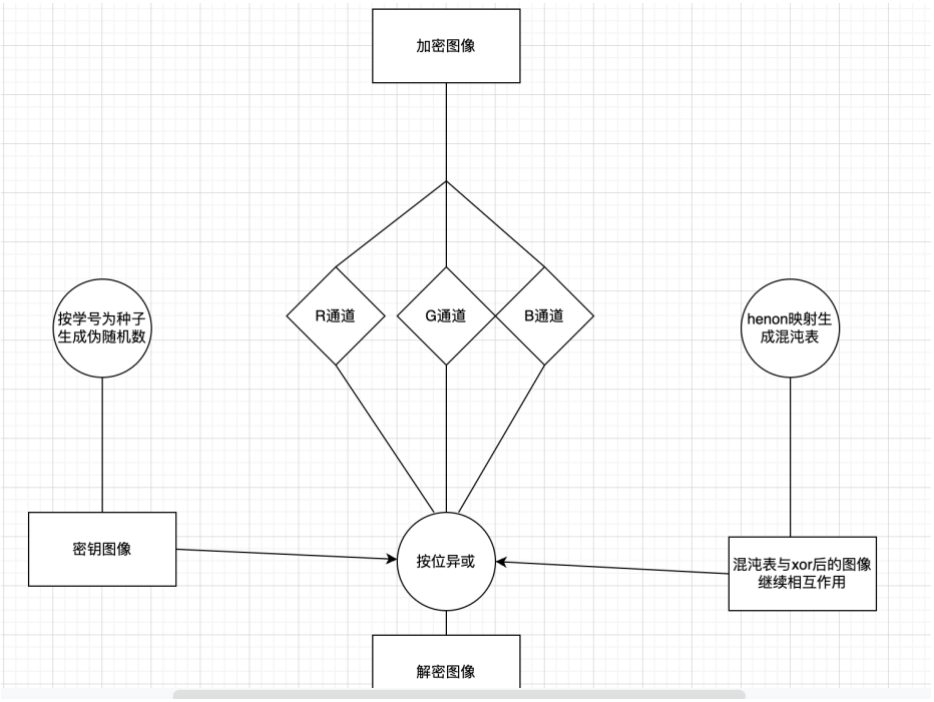


混沌系统与按位异或结合算法:

加密:



解密:

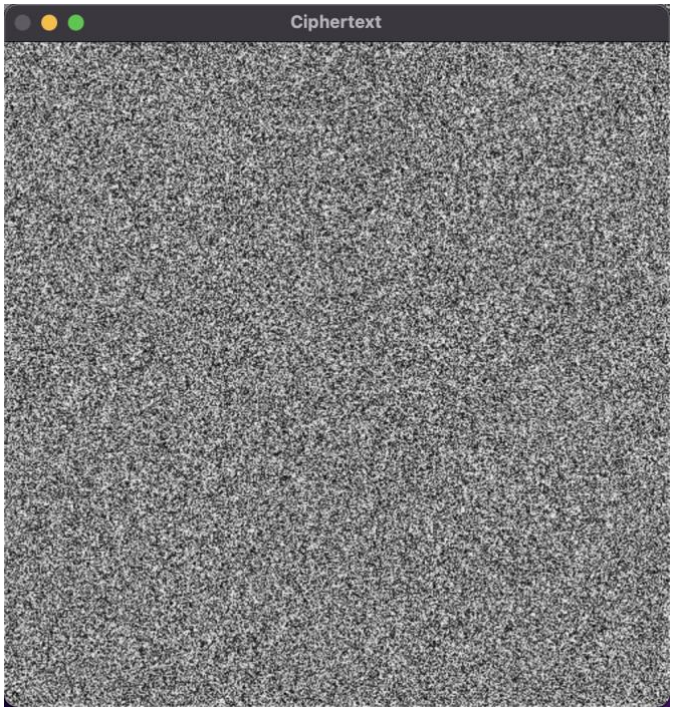


三．方案测试与分析

3.1 加解密正确性测试

按位异或算法:

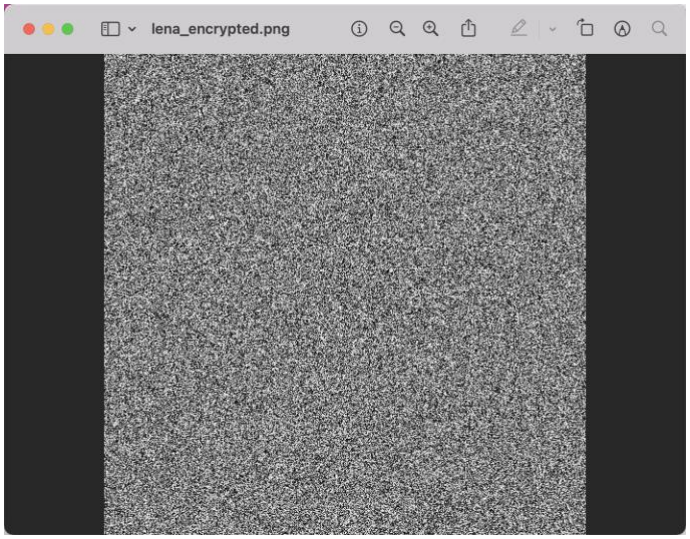
学号为：00000000，用学号对 lena 图像进行加密后的图像为:



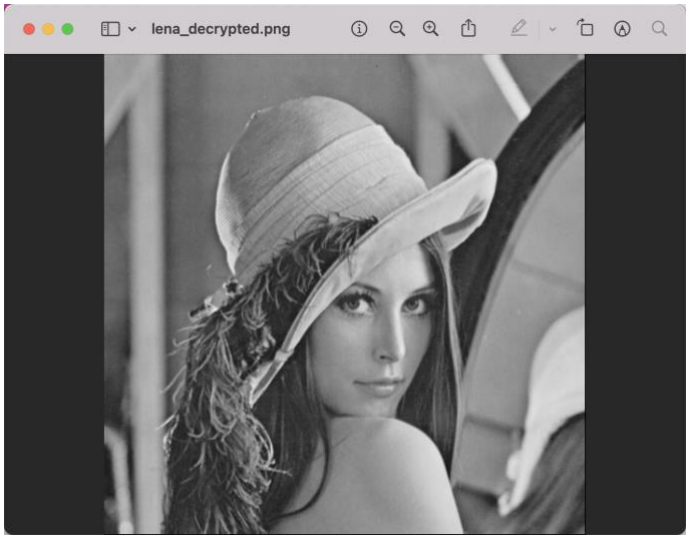
再次用学号作为密钥对图像进行解密得:

混沌系统与按位异或结合算法:

学号为：00000000，用学号对 lena 图像进行加密后的图像为:



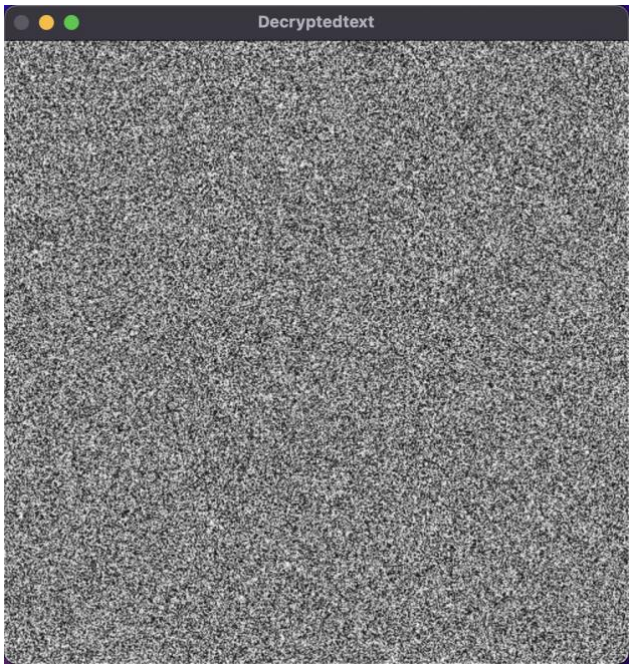
再次用学号进行解密得:



3.2 密钥敏感性测试

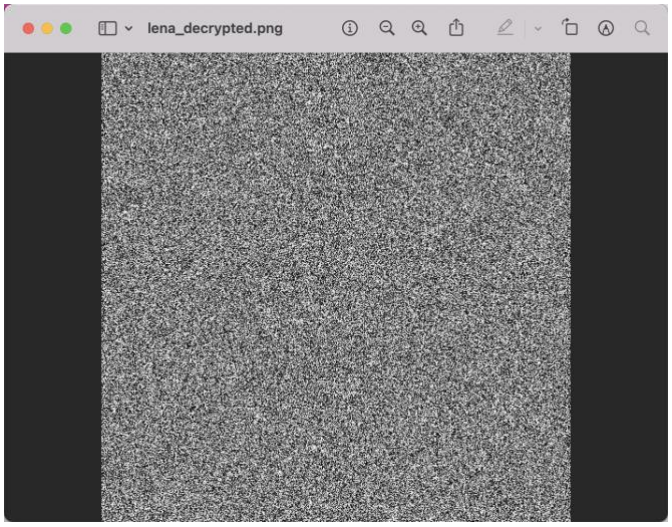
按位异或算法:

学号为 00000000, 用学号加一 00000000 作为解密密钥解密出的图像为:



混沌系统与按位异或结合算法：

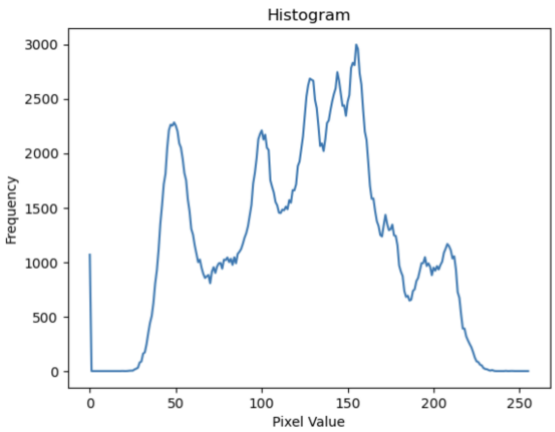
学号为 00000000，用学号加一 0000000000 作为解密密钥解密出的图像为：



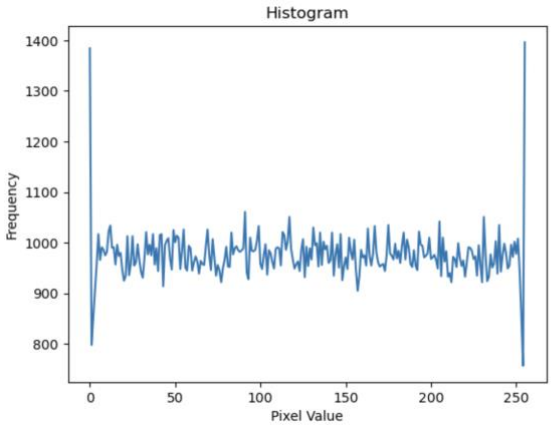
3.3 直方图测试

按位异或算法：

原始图像的直方图：

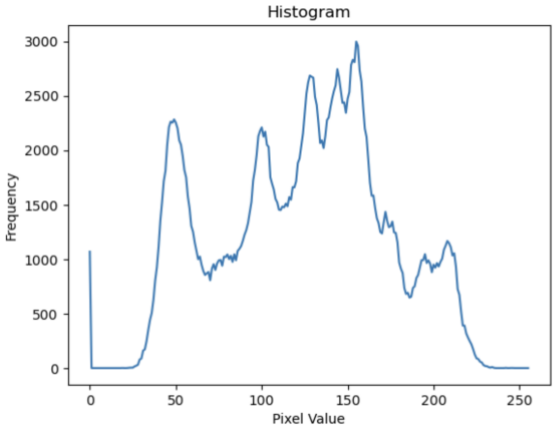


加密后的图像的直方图：

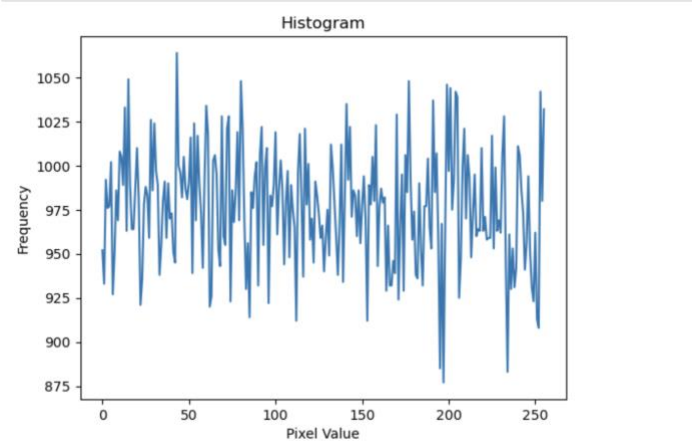


混沌系统与按位异或结合算法：

原始图像的直方图：



加密后图像的直方图：



3.4 算法运行时间分析

使用按位异或算法加密一次图像需要的时间:

```
Encryption Time: 1.4078617095947266 milliseconds
```

使用按位异或算法解密一次图像需要的时间:

```
Decryption Time: 0.14781951904296875 milliseconds
```

使用混沌系统结合按位异或算法加密一次图像需要的时间:

加密时间: 1.45 毫秒

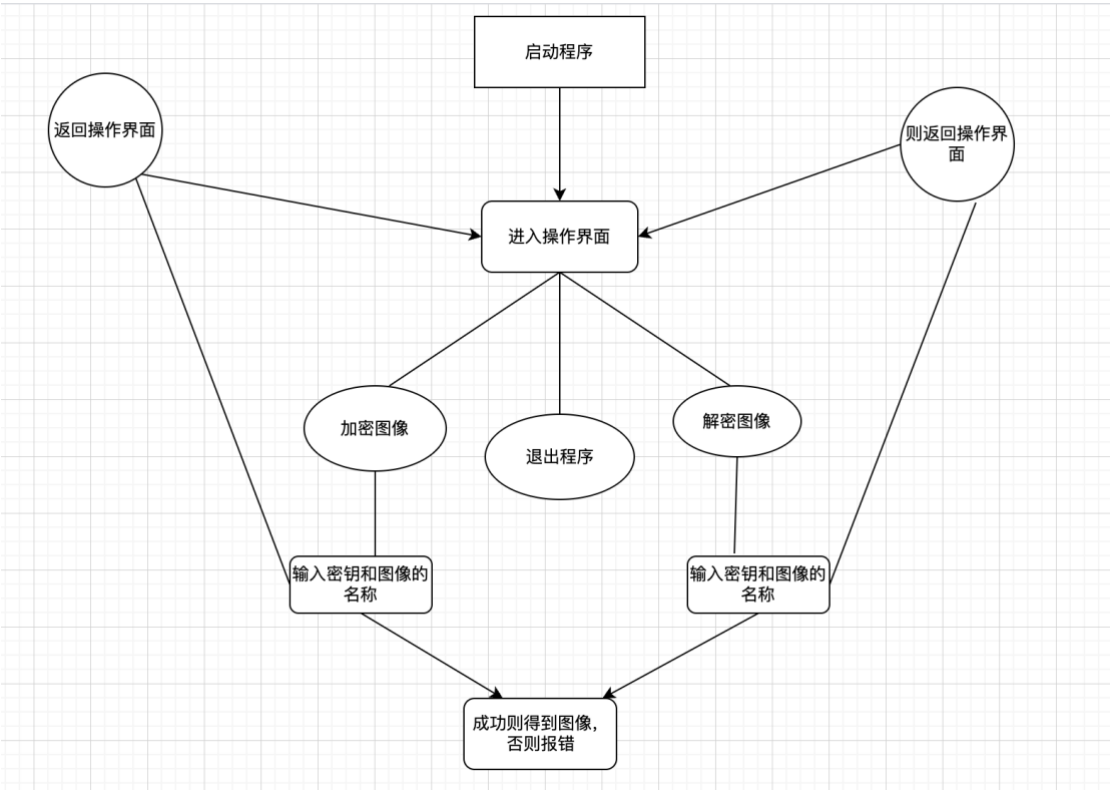
使用混沌系统结合按位异或算法解密一次图像需要的时间:

解密时间: 3.11 毫秒

四．创新性说明

一．图形界面

将 xor 对图像的加密解密程序封装成了具有图形界面的可供用户使用的操作界面，该图像界面的操作逻辑如下图:



程序运行如下：

```
xor
/Users/qianjiamac/PycharmProjects/pythonProject2/ve
********************
* 欢迎使用xor图像加解密系统！ *
********************
请选择：
1. 加密图像
2. 解密图像
0. 退出系统
请输入你的选项(0, 1, 2):
```

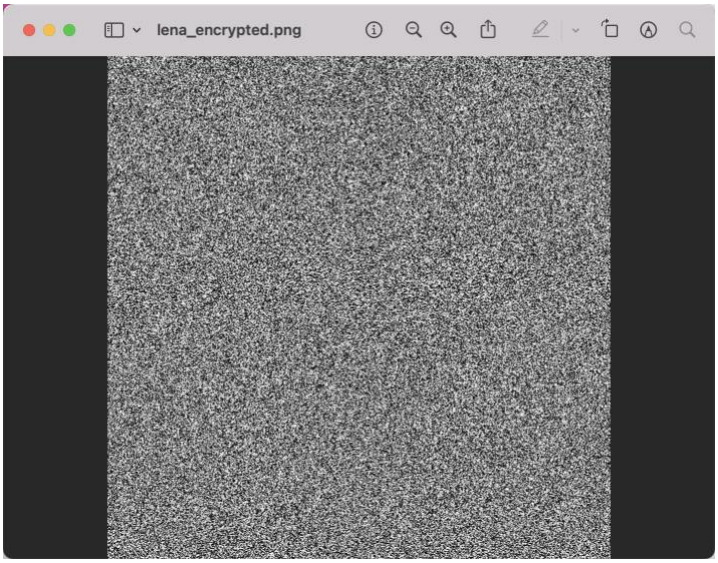
```
请输入你的选项(0, 1, 2): 1
输入密钥: 2020100309
输入图像的名称: lena.png
成功加密图像. 被加密后的图像名称为: lena_encrypted.png
```

```
********************
* 欢迎使用xor图像加解密系统！ *
********************
请选择：
1. 加密图像
2. 解密图像
0. 退出系统
请输入你的选项(0, 1, 2): 2
输入密钥: 2020100309
输入被加密的图像的名称: lena_encrypted.png
成功解密图像，解密后的图像名称为: lena_decrypted.png
```

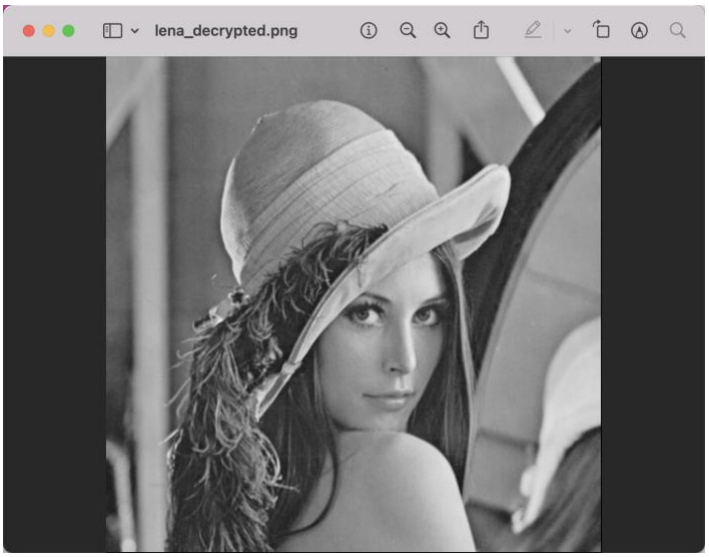
```
********************
* 欢迎使用xor图像加解密系统！ *
********************
请选择：
1. 加密图像
2. 解密图像
0. 退出系统
请输入你的选项(0, 1, 2): 0
********************
* Program Exited *
********************

Process finished with exit code 0
```

得到的加密后的图像和解密后的图像如下：

Lena_encrypted.png:



Lena_decrypted.png:



三． 安全性分析

按位异或是一种较为简单的加密方法，通过在每个像素的不同通道中使用密钥对原始像素进行异或操作，在这个过程中，它是一个线性操作，因此如果密钥是伪随机生成的，且长度合适不可预测，那么攻击者也很难通过简单的分析判断出密钥或者原图像。

代码中使用了 500X500 的随机数种子作为密钥，密钥的长度是固定的，在选择明文攻击中，如果攻击中能够获取足够多的明文-密文对，并且可以知道这些对之间的某种连接，那么可能可以推断出密钥的一部分或者全部。若同一个明文对应的密文被重复使用，攻击者可以通过比较不同的密文来发现某种相关性，所以，确保每个明文使用一个唯一的密钥很重要。

综上，在选择明文攻击中，攻击者可能尝试通过统计分析，图像分析的方法来推断出密钥或者明文，所以，要避免这种情况，使用足够长且随机的密钥，避免明文重用等手段是需要的。

唯密文攻击中，攻击者只拥有密文，虽然理论上按位异或是可逆的操

作，但是针对一个 500X500 的随机数种子作为密钥，攻击者想要通过密文直接破解的难度十分大，在密钥不泄漏的前提下也很难直接破解。

在结合了混沌系统 henon 映射后，加密算法具有了更高的安全性，首先在选择明文攻击中，因为经过了密钥 xor 和混沌映射表相加的操作，加入了混淆密钥的作用，攻击者很难通过部分明文与对应的密文来找到相关规律，因为 henon 映射是非线性的操作。因此该算法具有一定对选择明文攻击的安全性。

同理，在无法获取密钥和参数的情况下，攻击者不可能只通过密文来获取对应的解密结果，因此对唯密文攻击有较好的安全性。

四 . 当前图像加密算法研究现状分析

目前的图像加密算法多种多样，像一般的对称加密或者非对称加密等等已经不再是直接使用的加密算法，而是与其他新技术相结合从而加密，下面介绍几种新的算法：

1. 混沌加密算法：混沌加密算法指的是利用混沌系统的特性来对图像进行加密，混沌系统具有高度的不确定性和敏感性，使得加密过程更加随机和复杂。现有的一些常见的混沌加密算法例如 Arnold 变换，Henon 映射，Logistic 映射等等，这些算法都可以提供很强的安全性。
2. 基于人工智能的加密算法：随着 AI 技术的发展，研究人员也将 AI 技术运用到了图像加密中。这些方法利用深度神经网络来学习图像的表示和特征，并用于加密过程。
3. 多媒体加密标准。为了提高图像加密的标准化和可兼容性，一些多媒体加密标准被提出，比如 JEPG 2000 图像加密标准，它提供了一套统

一的加密方式，这样使得图像的加密更加规范，也有利于图像的加密技术的发展。

五. 设计一种混沌系统 henon 映射与按位异或结合的加密算法

详细说明在加密与解密原理中讲述。

五． 附录

按位异或加解密算法测试代码:

```
import cv2
import numpy as np
import os

def encrypt_image(seed, image_path):
    np.random.seed(seed)

    key = np.random.randint(0, 256, size=(500, 500), dtype=np.uint8)

    img_data = cv2.imread(image_path, -1)
    img_data = cv2.resize(img_data, (500, 500))

    img_data_b = img_data[:, :, 0]
    img_data_g = img_data[:, :, 1]
    img_data_r = img_data[:, :, 2]

    cipher_b = cv2.bitwise_xor(img_data_b, key)
    cipher_g = cv2.bitwise_xor(img_data_g, key)
    cipher_r = cv2.bitwise_xor(img_data_r, key)
    cipher_bgr = cv2.merge([cipher_b, cipher_g, cipher_r])

    encrypted_image_path = image_path.split(".")[0] + "_encrypted.png"
    cv2.imwrite(encrypted_image_path, cipher_bgr)

    return encrypted_image_path

def decrypt_image(seed, image_path):
    np.random.seed(seed)

```

```
    key = np.random.randint(0, 256, size=(500, 500), dtype=np.uint8)
    cv2.imwrite("Key_1.png", key)

    cipher_bgr = cv2.imread(image_path, -1)

    cipher_b = cipher_bgr[:, :, 0]
    cipher_g = cipher_bgr[:, :, 1]
    cipher_r = cipher_bgr[:, :, 2]

    decrypted_b = cv2.bitwise_xor(cipher_b, key)
    decrypted_g = cv2.bitwise_xor(cipher_g, key)
    decrypted_r = cv2.bitwise_xor(cipher_r, key)
    decrypted_bgr = cv2.merge([decrypted_b, decrypted_g, decrypted_r])

    image_name = os.path.splitext(image_path)[0]
    decrypted_image_name = image_name.rsplit("_", 1)[0] + "_decrypted.png"
    cv2.imwrite(decrypted_image_name, decrypted_bgr)

    return decrypted_image_name


seed = 2020100309
image_path = "lena.png"
encrypted_image_path = encrypt_image(seed, image_path)
print("成功加密图像。加密后的图像保存为:", encrypted_image_path)


seed = 2020100309
image_path = encrypted_image_path
decrypted_image_path = decrypt_image(seed, image_path)
print("成功解密图像。解密后的图像保存为:", decrypted_image_path)
```

封装为供用户使用的图像操作界面的加解密程序:

```
import cv2
import numpy as np
import time
import os

# 加密函数
```

```
def encrypt_image(seed, image_path):
    np.random.seed(seed)

    key = np.random.randint(0, 256, size=(500, 500), dtype=np.uint8)

    img_data = cv2.imread(image_path, -1)
    img_data = cv2.resize(img_data, (500, 500))

    img_data_b = img_data[:, :, 0]
    img_data_g = img_data[:, :, 1]
    img_data_r = img_data[:, :, 2]

    cipher_b = cv2.bitwise_xor(img_data_b, key)
    cipher_g = cv2.bitwise_xor(img_data_g, key)
    cipher_r = cv2.bitwise_xor(img_data_r, key)
    cipher_bgr = cv2.merge([cipher_b, cipher_g, cipher_r])

    encrypted_image_path = image_path.split(".")[0] + "_encrypted.png"
    cv2.imwrite(encrypted_image_path, cipher_bgr)

    return encrypted_image_path
2
# 解密函数
def decrypt_image(seed, image_path):
    np.random.seed(seed)

    key = np.random.randint(0, 256, size=(500, 500), dtype=np.uint8)
    cv2.imwrite("Key_1.png", key)

    cipher_bgr = cv2.imread(image_path, -1)

    cipher_b = cipher_bgr[:, :, 0]
    cipher_g = cipher_bgr[:, :, 1]
    cipher_r = cipher_bgr[:, :, 2]

    decrypted_b = cv2.bitwise_xor(cipher_b, key)
    decrypted_g = cv2.bitwise_xor(cipher_g, key)
    decrypted_r = cv2.bitwise_xor(cipher_r, key)
    decrypted_bgr = cv2.merge([decrypted_b, decrypted_g, decrypted_r])

```

```
    image_name = os.path.splitext(image_path)[0]
    decrypted_image_name = image_name.rsplit("_", 1)[0] + "_decrypted.png"
    cv2.imwrite(decrypted_image_name, decrypted_bgr)

    return decrypted_image_name

# 打印边框
def print_border(text):
    border = '*' * (len(text) + 4)
    print(border)
    print(f"* {text} *")
    print(border)

# 欢迎和选择界面
while True:
    print_border("欢迎使用 xor 图像加解密系统！")
    print("请选择:")
    print("1. 加密图像")
    print("2. 解密图像")
    print("0. 退出系统")

    option = input("请输入你的选项(0, 1, 2): ")

    if option == "1":
        seed = int(input("输入密钥: "))
        image_path = input("输入图像的名称: ")
        encrypted_image_path = encrypt_image(seed, image_path)
        print("成功加密图像. 被加密后的图像名称为:", encrypted_image_path)
    elif option == "2":
        seed = int(input("输入密钥: "))
        image_path = input("输入被加密的图像的名称: ")
        decrypted_image_path = decrypt_image(seed, image_path)
        print("成功解密图像，解密后的图像名称为:", decrypted_image_path)
    elif option == "0":
        break
    else:
        print("Invalid option. Please select 0, 1, or 2.")

    print()  # 空行分隔每次操作

print_border("Program Exited")
```

```
88.
```

混沌系统与按位异或结合的加解密算法:

```
1. import cv2
2. import numpy as np
3. import time
4.
5. # Henon 映射混沌系统参数
6. a = 1.4
7. b = 0.3
8.
9. np.random.seed(2020100309)  # 设置随机数种子
10.
11.def generate_key(seed, size):
12.    np.random.seed(seed)
13.    key = np.random.randint(0, 256, size=size, dtype=np.uint8)
14.    return key
15.
16.def henon_map(size):
17.    x = np.zeros(size)
18.    y = np.zeros(size)
19.    for i in range(1, size[0]):
20.        x[i] = 1 - a * x[i-1]**2 + y[i-1]
21.        y[i] = b * x[i-1]
22.    return (x + y) % 256
23.
24.def encrypt_image(seed, image_path):
25.    key = generate_key(seed, (500, 500))
26.
27.    img_data = cv2.imread(image_path, -1)
28.    img_data = cv2.resize(img_data, (500, 500))
29.
30.    img_data_b = img_data[:, :, 0]
31.    img_data_g = img_data[:, :, 1]
32.    img_data_r = img_data[:, :, 2]
33.
34.    henon_table = henon_map((500, 500))
35.    henon_table *= 100
36.    henon_table = henon_table.astype(int)
37.    henon_table = henon_table % 256
38.
39.    cipher_b = (img_data_b ^ key) + henon_table
```

```
    cipher_g = (img_data_g ^ key) + henon_table
    cipher_r = (img_data_r ^ key) + henon_table

    cipher_bgr = cv2.merge([cipher_b % 256, cipher_g % 256, cipher_r % 256])

    encrypted_image_path = image_path.split(".")[0] + "_encrypted.png"
    cv2.imwrite(encrypted_image_path, cipher_bgr)

    return encrypted_image_path

def decrypt_image(seed, image_path):
    key = generate_key(seed, (500, 500))

    cipher_bgr = cv2.imread(image_path, -1)

    cipher_b = cipher_bgr[:, :, 0]
    cipher_g = cipher_bgr[:, :, 1]
    cipher_r = cipher_bgr[:, :, 2]

    henon_table = henon_map((500, 500))
    henon_table *= 100
    henon_table = henon_table.astype(int)
    henon_table = henon_table % 256

    decrypted_b = (cipher_b - henon_table) ^ key
    decrypted_g = (cipher_g - henon_table) ^ key
    decrypted_r = (cipher_r - henon_table) ^ key

    decrypted_bgr = cv2.merge([decrypted_b % 256, decrypted_g % 256, decrypted_r % 256])

    decrypted_image_path = image_path.split("_")[0] + "_decrypted.png"
    cv2.imwrite(decrypted_image_path, decrypted_bgr)

    return decrypted_image_path


seed = 2020100309
image_path = "lena.png"
encrypted_image_path = encrypt_image(seed, image_path)
print("成功加密图像。加密后的图像保存为:", encrypted_image_path)
```

```
80.
81.seed = 2020100309
82.image_path = encrypted_image_path
83.decrypted_image_path = decrypt_image(seed, image_path)
84.print("成功解密图像。解密后的图像保存为:", decrypted_image_path)
```

画直方图:

```
1. import cv2
2. import numpy as np
3. import matplotlib.pyplot as plt
4.
5. def compute_histogram(image):
6.     # 将图像转换为灰度图像
7.     gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
8.
9.     # 计算直方图
10.    histogram = cv2.calcHist([gray_image], [0], None, [256], [0,
   256])
11.
12.    return histogram
13.
14.# 读取图像
15.image_path = "lena.png"
16.image = cv2.imread(image_path)
17.
18.# 计算直方图
19.histogram = compute_histogram(image)
20.
21.# 绘制直方图
22.plt.plot(histogram)
23.plt.title('Histogram')
24.plt.xlabel('Pixel Value')
25.plt.ylabel('Frequency')
26.plt.show()
```