《实验一 加解密算法的实现》实验报告

姓名	李宽宇	年级	21 级
学号	20215279	专业、班级	21 计卓 1 班

实验名称	实验一 信息隐藏实验		
实验时间	2024.4.13	实验地点	DS3305
实验成绩		实验性质	□验证性 □设计性 □综合性

教师评价：

□算法/实验过程正确； □源程序/实验内容提交 □程序结构/实验步骤合理；

□实验结果正确； □语法、语义正确； □报告规范；

评语：

评价教师签名（电子签名）：

一、实验目的

1.学习并掌握图像信息隐藏的基本原理和方法

2.实现基于 LSB 的信息隐藏和提取算法

3.用卡方检测对可疑图像进行 LSB 隐写检测

二、实验项目内容

编程实现基于 LSB 的信息隐藏和提取算法以及卡方检测算法。

1. 使用 LSB 算法在图片中隐藏如下信息：CQUWATERMASKEXP
2. 从被隐藏数据的图片中解析出如上信息，建议使用 Matlab
3. 用卡方检测算法对可疑图像进行检测

三、实验设计（实验原理、原理图）

1. LSB 方法

（1）图像的数字化表示：对图像数据而言，一幅图像的每个象素是以多比特的方式构成的。在灰度图像中，每个象素通常为 8 位；在真彩色图像(RGB 方式)中，每个象素为 24 比特，其中 RGB 三色各为 8 位，每一位的取值为 0 或 1。在数字图像中，每个象素的各个位对图像的贡献是不同的。对图像，可以根据像素的比特为来进行图像分解，对灰度图像，从 LSB(最低有效位 0)到 MSB(最高有效位 7)可以分解为 8 个平面

（2）不同平面的作用：**低位所代表的能量很少，改变低位对图像的质量没有太大的影响**。LSB 方法正是利用这一点在图像低位隐藏入水印信息。在图像中，高位平面对图像感官质量起主要作用，去除图像最低几个位平面并不会造成画面质量的下降。利用这个原理可用秘密信息（或称水印信息）替代载体图像低位平面以实现信息嵌入。LSB 算法选用最低位平面来嵌入信息，最低位平面对图像的视觉效果影响最轻微，因此在视觉上很难察觉。

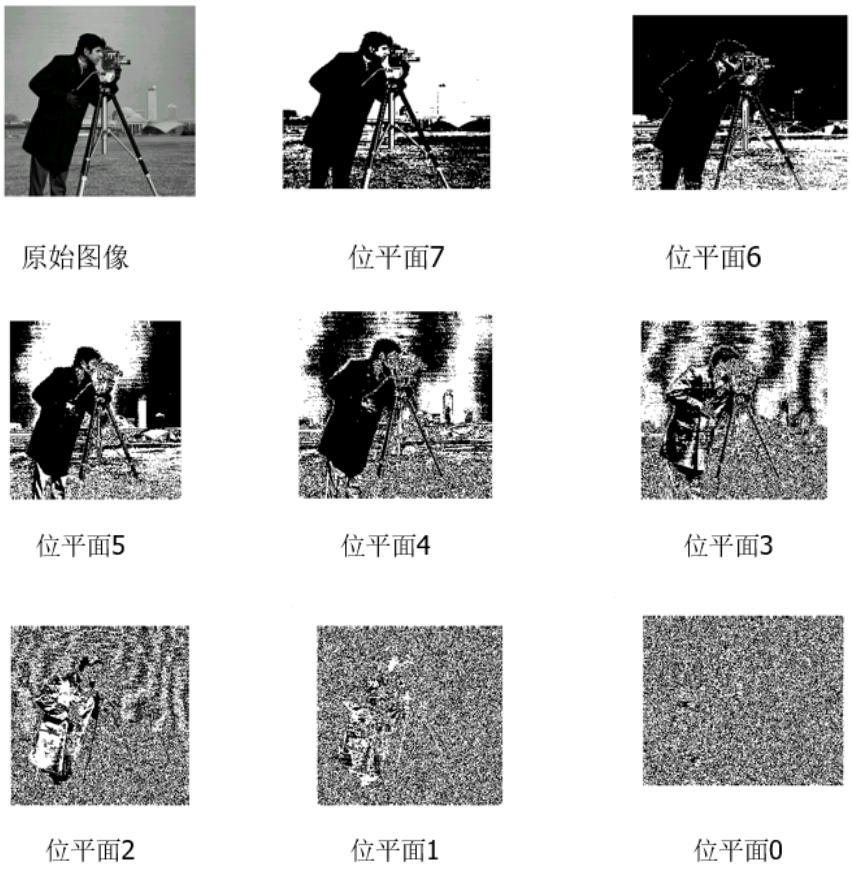


图 1：各个位平面对视觉效果的影响

（3）LSB 主要步骤：

加密：

step1.读入图片

step2.准备待隐藏的信息，将其转换为二进制（需要用加密算法进行加密）

step 3.遍历图像，对像素的最低 1bit 置 0，同时在该比特位写入 1 位二进制表示隐藏的信息

解密：

step 1.预知隐藏信息量（等同于 key）

step 2.提取出像素的最低 1bit，组合成连续 bit 数据，转换为 ASCII 码对比是否与隐藏信息一致

2.卡方检测算法

原理：通过统计样本的实际观测值与理论推断值之间的偏离程度来判断是否存在隐写信息。LSB 算法中，如果秘密信息位与隐藏位置的像素灰度值的最低比特位相同，不改变原始载体，反之，则改变灰度值的最低位

约定：

q：一个像素被选中用于隐藏信息的概率；

$T_c[j], j = 0,1,2,\cdots,255$：载体图像中，值为$j$的像素个数；

$T_s[j], j = 0,1,2,\cdots,255$：隐写图像中，值为$j$的像素个数；

假设：

秘密消息中比特0和1随机分布；

$T_c[2i]$个值为$2i$的像素中，有$\frac{q}{2}T_c[2i]$个像素的最低比特与消息相同，不需要修改；

- 当$q = 1$时：
  - $E\{T_S[2i]\} = E\{T_S[2i+1]\}$
  - $= 0.5\{T_c[2i] + T_c[2i+1]\}$
  - $= 0.5\{T_s[2i] + T_s[2i+1]\}$
  - 即，对于隐写图像来说，
  - 值为$2i$的像素个数的观测值为：$T_s[2i]$
  - 值为$2i$的像素个数的理论值$\overline{T_s}[2i]$为：$0.5\{T_s[2i] + T_s[2i+1]\}$

- 隐写分析：
  - 计算待检测图像统计量s，s的值越小，意味$\overline{T_s}[2i]$与$T_s[2i]$越一致，也就是说待检测图像是隐写图像的概率越高；
  - 反之，s的值越大，意味$\overline{T_s}[2i]$与$T_s[2i]$差异越大，也就是说待检测图像是隐写的概率越低

- 假设：
  - 有$\frac{q}{2}T_c[2i]$个像素最低比特与消息不同，像素值变为$2i+1$；
  - 类似地，值为$2i+1$的像素中（原始像素值最低位为1），有$\frac{q}{2}T_c[2i+1]$个像素最低比特与消息不同，像素值变为$2i$；
- 可得：
  - $E\{T_S[2i]\} = \left(1 - \frac{q}{2}\right)T_c[2i] + \frac{q}{2}T_c[2i+1]$

- 卡方检验：
  - 如果图像LSB隐写，那么$\overline{T_s}[2i]$与$T_s[2i]$一致。
  - 可以用卡方检验来检测$\overline{T_s}[2i]$与$T_s[2i]$的一致性。
  - 由卡方检验原理可知，统计量
  - $s = \sum_{i=1}^{k} \frac{(T_s[2i] - \overline{T_s}[2i])^2}{\overline{T_s}[2i]}$
  - 服从卡方分布（χ^2分布）。

四、实验过程或算法(关键步骤、核心代码注解等)

1.使用 LSB 算法在图片中隐藏如下信息：CQUWATERMASKEXP

(1) 基于实验 1 的代码，用 Feistel 加密,加密信息 CQUWATERMASKEXP,加密后的明文和密钥如下图所示：

```
"""
基于实验1的代码，采用实现Feistel加密解密
keys =
15782, 31857, 16446, 17342, 7731, 10979, 4057, 25173, 16245, 24647, 29548,
 16031, 24278, 2482, 29640, 3573


明文 87, 88, 16, 10, 65, 78, 6, 6, 78, 94, 12, 25, 53, 120, 49, 69, 67, 99,
 48, 73, 84, 111, 39, 88, 80, 22
"""
hidden_assic= [87, 88, 16, 10, 65, 78, 6, 6, 78, 94, 12, 25, 53, 120, 49, 69,
  67, 99, 48, 73, 84, 111, 39, 88, 80, 22]

hidden_binary = ''.join(format(number, '08b') for number in hidden_assic)
```

(2)在重庆大学首页上下载图片

(3)使用 PLL 库加载图片，获取图片大小

```
from PIL import Image
img = Image.open('image/1.jpg')
width, height = img.size
Executed at 2024.04.20 00:21:02 in 51ms
```

(4)打印转换成二进制的密文

```
print(hidden_binary)
Executed at 2024.04.20 00:21:02 in 12ms
```

```
0101011101011000000100000000101001000001010011100000011000000110010011
1001011110000011000001100100110101011110000011000101000101010000011011
0001100110000010010010101010001101111001001110101100001010000000010110
```

(5) 遍历图像，对像素的最低 1bit 置 0，同时在该比特位写入 1 位二进制表示隐藏的信息

```
hidden_index = 0
for y in range(height):
    for x in range(width):
        pixel = list(img.getpixel((x, y)))
        if x == 0 and y == 0:
            print("原始像素点（0, 0）rgb值:", pixel)
        for i in range(3):
            if hidden_index < len(hidden_binary):
                # 最低位置0，可采用“&=~1”
                # ~1 = ~00000001=1111 1110
                #  129=          1000 0001
                pixel[i] &= ~1  # Set the least significant bit to 0
                #  128 = 1000 0000
                # 如果待隐藏信息位是0，rgb某一位要写成0
                # 如果待隐藏信息位是1，rgb某一位要写成1
                # 由于已经置0，“|=隐藏信息位”即可
                pixel[i] |= int(hidden_binary[hidden_index])
                if x == 0 and y == 0:
                    print("待隐藏信息位信息：", hidden_binary[hidden_index])
                hidden_index += 1
        img.putpixel((x, y), tuple(pixel))
        pixel = list(img.getpixel((x, y)))
        if x == 0 and y == 0:
            print(pixel)
Executed at 2024.04.20 00:21:07 in 4s 702ms
```

例如：

```
原始像素点（0, 0）rgb值: [129, 166, 234]
待隐藏信息位信息： 0
待隐藏信息位信息： 1
待隐藏信息位信息： 0
[128, 167, 234]
```

(6)输出隐藏了信息的图片

```
# 输出图像
img.save("encrypted_image.png")
img.show()  # 显示图像
Executed at 2024.04.20 00:21:11 in 4s 443ms
```

难以看出与原图的区别

2.从被隐藏数据的图片中解析出如上信息

(7)获取隐藏的信息，读取图片 encrypted_image.png

```
hidden_index = 0
img = Image.open("encrypted_image.png")
width, height = img.size
```

遍历图片，根据已知的长度提取出像素的最低 1bit

```
de_binary = ""
while hidden_index < len(hidden_binary):
    for y in range(height):
        for x in range(width):
            pixel = list(img.getpixel((x, y)))
            for i in range(3):
                if hidden_index < len(hidden_binary):
                # 提取出像素的最低1bit
                    de_binary += str(pixel[i] & 1)
                    hidden_index+=1
```

Executed at 2024.04.20 00:21:13 in 2s 54ms

打印图片中隐藏的明文

```
# 打印明文
secret_message = []
for i in range(0, len(de_binary), 8):
    byte = de_binary[i:i + 8]
    secret_message.append(int(byte,2))
print(secret_message)
```

Executed at 2024.04.20 00:21:13 in 4ms

```
[87, 88, 16, 10, 65, 78, 6, 6, 78, 94, 12, 25, 53, 120, 49, 69, 67,
99, 48, 73, 84, 111, 39, 88, 80, 22]
```

（8）基于实验 1 的加密算法的解密部分，输入指定的 key=15782,

31857, 16446, 17342, 7731, 10979, 4057, 25173, 16245, 24647, 29548, 16031, 24278, 2482, 29640, 3573

和从图片中获取的信息[87, 88, 16, 10, 65, 78, 6, 6, 78, 94, 12, 25, 53, 120, 49, 69, 67, 99, 48, 73, 84, 111, 39, 88, 80, 22]

```
REn = {87, 88, 16, 10, 65, 78, 6, 6, 78, 94, 12, 25, 53};
LEn = {120, 49, 69, 67, 99, 48, 73, 84, 111, 39, 88, 80, 22};
cout << endl
     << "解密: " << endl;
pair<vector<char>, vector<char>> resultD = feistel(REn, LEn, true);
RDn = resultD.first;
LDn = resultD.second;
for (size_t i = 0; i < RDn.size(); ++i)
{
    cout <<RDn[i];
}
for (size_t i = 0; i < LDn.size(); ++i)
{
    cout << LDn[i];
}
```

解密出 CQUINFORMATIONSECURITYEXP，与密文一致

```
解密:
CQUINFORMATIONSECURITYEXP
(base) PS C:\Users\a3840>
```

3.用卡方检测算法对可疑图像进行检测

使用小组同学朱齐高提供的加密后的图片和原始图像（已将信息循环写入图片，只在 50~80 像素值的位置进行写入，便于卡方分析画图）

需要我判断哪一个图像隐写了信息

首先导入原始图像

```
from PIL import Image
imgOrigin = Image.open('image/1.jpg')
width, height = imgOrigin.size
Executed at 2024.04.22 15:44:13 in 67ms
```

统计各像素值的数目

```
byteArray = [0 for i in range(30)]

for y in range(height):
    for x in range(width):
        pixel = list(imgOrigin.getpixel((x, y)))
        for i in range(3):
            if 50<=pixel[i]<80:
                byteArray[pixel[i]-50] += 1
print(byteArray)
Executed at 2024.04.22 15:44:15 in 2s 273ms
```

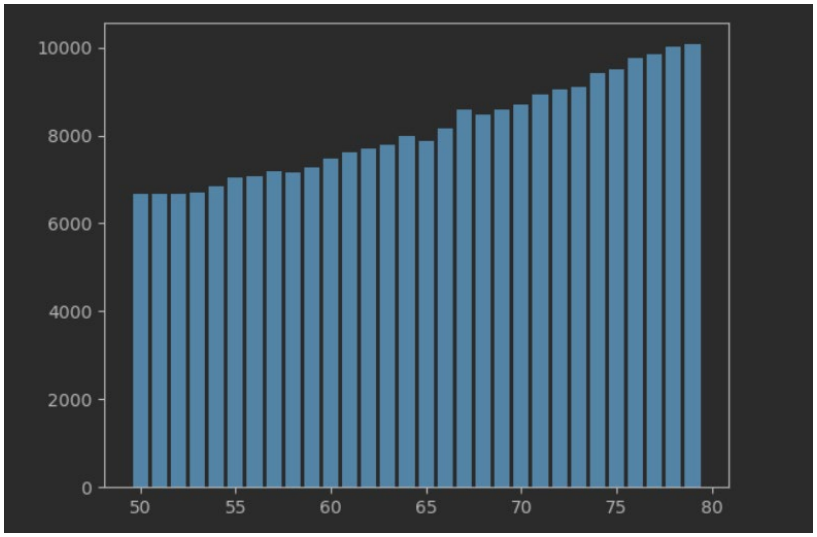
结果：

```
[6676, 6678, 6666, 6703, 6830, 7047, 7065, 7174, 7160, 7284, 7465,
 7606, 7712, 7780, 7974, 7876, 8167, 8588, 8471, 8593, 8713, 8934,
 9056, 9106, 9414, 9501, 9768, 9846, 10026, 10066]
```

绘制原始图片的柱状图

```
import matplotlib.pyplot as plt
plt.bar(range(50,80), byteArray)
plt.show()
Executed at 2024.04.22 15:44:16 in 608ms
```

图像如下，由于实际场景下，图像为渐变的，即相近像素的值会比较接近。



同理再编写代码绘制异常图片

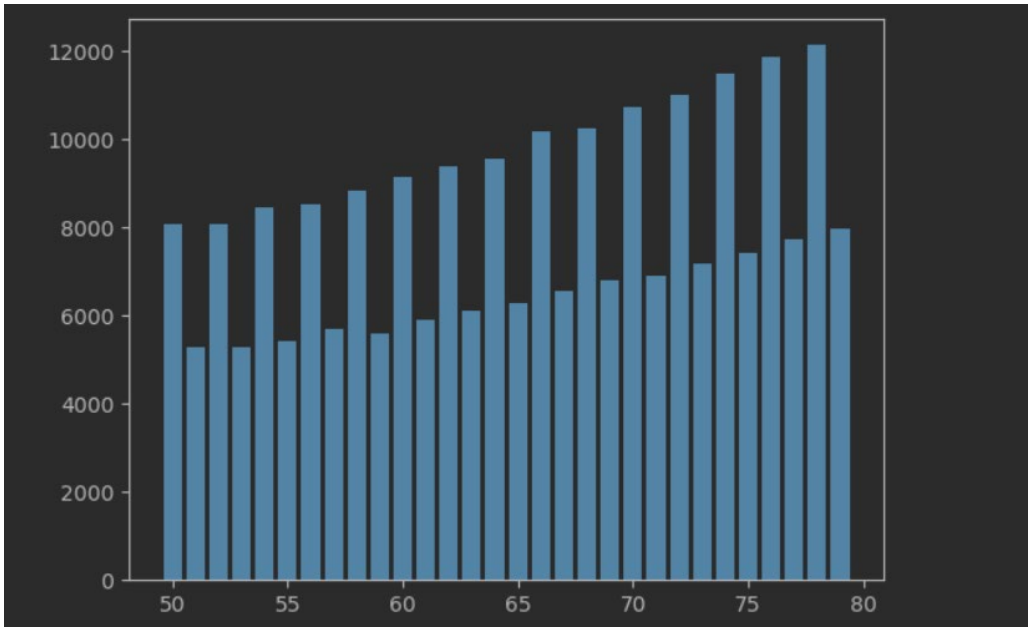
```
from PIL import Image

imgOrigin = Image.open('encrypted_image.png')
width, height = imgOrigin.size
byteArray = [0 for i in range(30)]

for y in range(height):
    for x in range(width):
        pixel = list(imgOrigin.getpixel((x, y)))
        for i in range(3):
            if 50 <= pixel[i] < 80:
                byteArray[pixel[i] - 50] += 1
print(byteArray)
print("像数值50的个数，像数值51的个数{}:{}".format(byteArray[0]/(byteArray[0]+byteArray[1]), byteArray[1]/(byteArray[0]+byteArray[1])))
import matplotlib.pyplot as plt

plt.bar(range(50, 80), byteArray)
plt.show()
```

图像如下：



偶数与奇数的比例是 6:4，例如

```
像数值50的个数：像数值51的个数0.6045379661524637:0.3954620338475363
```

可以看出图像有明显的异常

统计隐藏信息的 01 比例

```
隐藏信息0的数目和1的数目的比例0.6057692307692307:0.3942307692307692
```

计算$f(x)=\sum_{i=1}^{k}\left(\frac{T_{s[2i]}-\overline{T_{s[2i]}}}{\overline{T_{s[2i]}}}\right)^2$

其中 k:[25, 40)，$\overline{T_{s[2i]}}=0.605769*T_{c[2i]}+0.605769T_{c[2i+1]}$

计算$\overline{T_{s[2i]}}$：

```
ETs = [0 for i in range(15)]
for i in range(15):
    ETs[i] = sum0 / (sum1 + sum0) * byteArray[2 * i] + sum0 / (sum1 + sum0) *
    byteArray[2 * i + 1]
Executed at 2024.04.22 16:55:47 in 1s 954ms
```

计算$T_{s[2i]}$：

这里的 byteArray 是隐写了信息的图像，上面的是原始图像对应的

```
Ts = [0 for i in range(15)]
for i in range(15):
    Ts[i] = byteArray[2 * i]
```

计算卡方

```
f = 0
print(Ts)
print(ETs)
for i in range(15):
    f += ((Ts[i] - ETs[i])) ** 2 / ETs[i]
print(f)
```

结果

```
[8073, 8070, 8454, 8526, 8843, 9158, 9371, 9567, 10189, 10265, 10736,
10991, 11506, 11868, 12135]
[8089.442307692308, 8098.528846153846, 8406.259615384613, 8625
.548076923076, 8749.73076923077, 9129.548076923076, 9384.576923076922,
9601.442307692307, 10149.663461538461, 10336.846153846152, 10690
.009615384613, 11001.98076923077, 11458.125, 11881.557692307691,
12171.115384615383]
3.963326552149944
```

卡方值为 3.9633，n 为 15（因为只在[50,80)范围内写信息）

查表知，4.601 对应 0.995，3.9633<4.601

15	4.601	5.229	6.262	7.261	8.547	11.037	18.245	22.307

所以有超过 0.995 的把握认为第二张图像隐写了该信息

五、实验过程中遇到的问题及解决情况(主要问题及解决情况)

1.主要问题：C++图片处理不是很熟悉

解决情况：转变思路，用 python 处理图片，用 DataSpell 展示中间过程

2.主要问题：二进制数最低位置 0

解决情况：

```
# 最低位置0，可采用“&=~1”
# ~1 = ~00000001=1111 1110
#  129=          1000 0001
pixel[i] &= ~1
```

3. 主要问题：需要加密算法

解决情况：使用实验一实现的加密算法

六、实验结果及分析和（或）源程序调试过程

1.使用 LSB 算法在图片中隐藏如下信息：CQUWATERMASKEXP

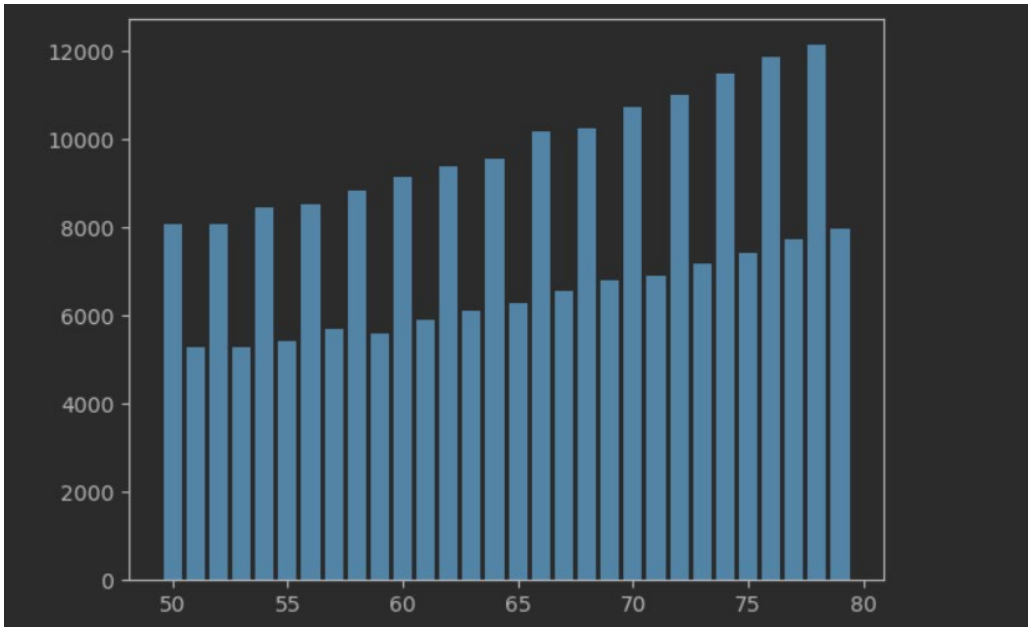
写入后的图像

2. 从被隐藏数据的图片中解析出如上信息，

解密出 CQUINFORMATIONSECURITYEXP，与密文一致

3. 用卡方检测算法对可疑图像进行检测

异常图像的直方图：



卡方值为 3.9633，n 为 15（因为只在[50,80)范围内写信息）

查表知，4.601 对应 0.995，3.9633<4.601

所以有超过 0.995 的把握认为第二张图像隐写了该信息