南开大学
Nankai University

南　　开　　大　　学

网络空间安全学院

信息隐藏技术实验报告

实验 7：奇偶校验位隐藏法

姓名：2113203 付政烨

年级：2021 级

专业：信息安全、法学

指导教师：李朝晖

2024 年 4 月 17 日

目录

一、 实验内容

(一) 实验目的

- **隐藏**：利用奇偶校验位隐藏法，实现将秘密图像嵌入到位图中；
- **提取**：将秘密图像提取出来。

(二) 实验环境

- 运行系统：Windows11
- 实验工具：Matlab2022a
- 数据：PNG 格式图像

(三) 实验要求

- 在 MATLAB 中调试完成
- 编写实验代码和报告，并给出截图
- QQ 群提交作业

二、 实验原理

(一) 方法一

该方法特点是翻转最低位，影响不大。把载体划分成几个不相重叠的区域，在一个载体区域中存储一比特信息。

1. 嵌入

选择 $L(m)$ 个不相重叠区域，计算出每一区域 I 的所有最低比特的奇偶校验位（即“1”的个数奇偶性），$b_i(i = 1, 2, ..., n)$。

$$b_i = \sum_{j \in I} LSB(c_j) \mod 2$$

嵌入信息时，在对应区域的奇偶校验位上嵌入信息比特 m_i，如果奇偶校验位 b_i 与 m_i 不匹配，则将该区域中所有元素的最低比特位进行翻转，使得奇偶校验位与 m_i 相同，即 $b_i = m_i$。

例如一个区域内所有像素的最低比特有偶数个 1，计算得奇偶校验位 $b_i = 0$。如果要嵌入的秘密信息比特为 1，即 $m_i = 1$，要想满足 $b_i = m_i$ 则需要翻转所有像素的最低比特位，使得该区域的最低有效位有奇数个 1，即 $b_i = 1$，从而满足 $b_i = m_i$。

2. 提取

在接收端，收方与发方拥有共同的伪装密钥作为种子，可以伪随机地构造载体区域。收方从载体区域中计算出奇偶校验位，排列起来就可以重构秘密信息。

(二) 方法二

该方法特点是翻转像素少。把载体划分成几个不相重叠的区域，在一个载体区域中存储一比特信息。

1. 嵌入

选择 $L(m)$ 个不相重叠区域，计算出每一区域 I 的所有最低比特的奇偶校验位 $b_i(i = 1, 2, ..., n)$。

$$b_i = \sum_{j \in I} LSB(c_j) \mod 2$$

区域 I 隐藏一个信息比特。若 b_i 与 m_i 不同，则将该区域中某个像素的最低比特位进行翻转，使得奇偶校验位与 m_i 相同，即 $b_i = m_i$。例如一个区域内所有像素的最低比特位有偶数个 1，计算得奇偶校验位 $b_i = 0$。如果要嵌入的秘密信息比特为 1，即 $m_i = 1$，要想满足 $b_i = m_i$ 则需要翻转某个像素的最低比特位，使得该区域的最低有效位有奇数个 1，即 $b_i = 1$，从而满足 $b_i = m_i$。

2. 提取

用同样的方法划分载体区域，计算出奇偶校验位，构成秘密信息

三、 实验步骤

本实验采用了第二种部分翻转法。主要有如下四个步骤:

1. 主函数调用，实现整个实验。
2. 奇偶检验位函数。
3. 加密函数。
4. 解密函数。

(一) HideAndExtract 主函数

```
function HideAndExtract()
    x=imread('Lena.bmp'); %载体图像
    y=imread('lion.bmp'); %秘密信息图像  是灰度图像，长宽均为载体图像的一半
    y=imbinarize(y);
    [m, n]= size(y);

    subplot(2, 2, 1);
    imshow(x) ; title('原始图像');

    subplot(2, 2, 2);
    imshow(y) ; title('水印图像');

    x=Hide(x,m,n,y);
    subplot(2, 2, 3);
    imshow(x ,[]) ; title('伪装图像');

```

```
    t=Extract();
    subplot(2,2,4);
    imshow(t,[]); title("提取出的水印图像");
end
```

该函数为主入口，负责调用其他函数执行图像的载入、信息隐藏、信息提取，并显示结果。使用 imread 加载载体图像 Lena.bmp 和秘密信息图像 lion.bmp。秘密信息图像为灰度图，尺寸是载体图像的一半，并且通过 imbinarize 函数将其二值化。通过四个子图显示原始载体图像、二值化的秘密信息图像、隐藏信息后的伪装图像和提取出的水印图像。调用 Hide 函数将秘密信息嵌入载体图像中，并通过 Extract 函数提取隐藏的信息。

(二) checksum 函数

```
function out = checksum (x, i, j)
   temp= zeros(1, 4);
   temp(1) = bitget(x(2*i-1,2*j-1), 1);
   temp(2) = bitget(x(2*i-1,2*j), 1);
   temp(3) = bitget(x(2*i, 2*j-1), 1);
   temp(4) = bitget(x(2*i, 2*j ), 1);
   out=rem(sum(temp), 2);
end
```

checksum 函数用于计算指定区域的最低位校验和（奇偶性）。输入坐标 (i, j)，针对此坐标指定的小区域（2x2 像素块），计算最低位的奇偶性。使用 bitget 函数获取像素的最低位，并计算这四个像素的最低位之和。利用 rem 函数确定总和的奇偶性，以决定该区域的校验和。

(三) Hide 函数

```
function result=Hide(x,m,n,y)
    for i =1:m
        for j =1:n
            if checksum(x, i, j) ~= y(i, j)
                random= int8(rand()*3);
                switch random
                                    case 0
                                            x(2*i-1,2*j-1)= bitset(x(2*i-1,2*j-1), 1, ~
                                                bitget(x(2*i-1,2*j-1), 1));
                                    case 1
                                            x(2*i-1,2*j)= bitset(x(2*i-1,2*j) , 1 , ~ bitget
                                                (x(2*i-1,2*j), 1));
                                    case 2
                                            x(2*i, 2*j-1)= bitset(x(2*i, 2*j-1) ,1 ,~ bitget
                                                (x(2*i , 2*j-1) , 1));
                    case 3
                                            x(2*i , 2*j)= bitset(x(2*i , 2*j) , 1 , ~ bitget
                                                (x(2*i , 2*j) , 1));
                end
            end
```

```
        end
    end
    imwrite(x , 'watermarkedImage.bmp');
    result=x;
end
```

Hide 函数旨在从加密的载体图像中提取隐藏的信息。读取 watermarkedImage.bmp 文件，对每个小区域使用 checksum 函数提取其奇偶性。根据奇偶性重构原始的秘密信息图像，即提取出的水印图像。整体来看，该代码利用图像的最低有效位的微小变化来嵌入秘密信息，这种方法对图像的视觉影响较小，同时能有效隐藏信息。

(四) Extract 函数

```
function out=Extract()
    c=imread('watermarkedImage.bmp');
    [m, n]= size(c);
    secret = zeros(m/2 , n/2);
    for i =1:m/2
        for j =1: n/2
            secret(i, j)= checksum(c, i, j);
        end
    end
    out=secret;
end
```

Extract 函数用于从加密的载体图像中提取隐藏的信息。读取 watermarkedImage.bmp 文件，对每个小区域使用 checksum 函数提取其奇偶性。根据奇偶性重构原始的秘密信息图像，即提取出的水印图像。该代码利用图像的最低有效位的微小变化来嵌入秘密信息，这种方法对图像的视觉影响较小，同时能有效隐藏信息。

(五) 实验结果

图 1: 实验结果

四、 扩展实验-多位 LSB 嵌入增强隐蔽性与鲁棒性

(一) 原理简介

(1) 数字图像表示

数字图像通常以像素阵列的形式表示，每个像素有一个或多个数值表示其颜色。在灰度图像中，每个像素通常用一个 8 位的整数表示，范围从 0（黑色）到 255（白色）。在彩色图像中，每个像素通常由三个 8 位整数表示，分别对应红色、绿色和蓝色（RGB）通道。

(2) 最低有效位（LSB）

一个 8 位的数字可以表示为从最低位到最高位的二进制数。例如，数字 150 可以表示为二进制 `10010110`。其中，最右边的位（这里是 `0`）是最低有效位。因为在二进制数中，每一位的权重是指数级增长的，最低位的权重最小，修改这一位对数值的影响最小，通常也最不容易在视觉上被察觉。

(3) 多位 LSB 嵌入

设 p 是一个 8 位的像素值，p 可以表示为：

$$p = b_8 2^7 + b_7 2^6 + \ldots + b_1 2^0$$

其中，b_1 是最低有效位，b_8 是最高位。

如果我们要修改最低的 2 位来嵌入信息，假设嵌入的信息是两位二进制数 d_1 和 d_2，我们可以重新设置 p 的最低两位为 d_1 和 d_2，计算公式为：

$$p' = (p \wedge 252) + 2d_2 + d_1$$

其中，‘252’ 是二进制的 ‘11111100’，该操作保留了 p 的最高六位，而最低两位被设置为 0。

(4) 信息提取

提取信息时，可以直接读取像素值的最低 2 位：

$$d_1 = p' \wedge 1$$

$$d_2 = (p' \wedge 2) >> 1$$

其中，`>> 1` 表示右移一位，即除以 2。

(5) 图像质量的数学评估

峰值信噪比 PSNR 是一种常用的评估图像质量的方法，特别是在图像处理中用于评估原始图像和处理后图像之间的差异。PSNR 定义为：

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\text{MAX}^2}{\text{MSE}} \right)$$

其中，MAX 是可能的最大像素值（对于 8 位图像，为 255），MSE 是均方误差，计算公式为：

$$\text{MSE} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} (I(i,j) - K(i,j))^2$$

I 是原图像，K 是处理后的图像，m 和 n 是图像的宽度和高度。

(二) 代码实现

1. 主函数 HideAndExtractMultiLSB

首先，函数通过 imread 读取载体图像 Lena.bmp 和秘密信息图像 lion.bmp。秘密信息图像随后通过 imbinarize 函数进行二值化处理，这一步是必要的因为多位 LSB 嵌入需要处理的是二进制数据。

将二值化后的图像 y 转换为一维二进制序列。这一步是将图像数据重新格式化，使其可以逐位嵌入到载体图像的 LSB 中。然后，使用 MATLAB 的 subplot 函数来安排四个图像显示区域，其中第一个显示原始的载体图像，第二个显示秘密信息图像。

```
secretData = reshape(y, 1, numel(y));
figure;
subplot(2, 2, 1);
imshow(x); title('原始图像');
subplot(2, 2, 2);
imshow(y); title('秘密信息图像');
```

调用自定义的 HideMultiLSB 函数将二进制序列嵌入到载体图像的最低两位（由第二个参数 2 指定）。这个函数将遍历载体图像的每个像素，并将秘密数据的位依次嵌入到像素的 LSB 中。在图像处理窗口的第三个位置，显示经过信息隐藏处理后的图像。这个图像外观上与原始图像非常接近，但已经包含了隐藏的信息。

```
modifiedImage = HideMultiLSB(x, 2, secretData);
subplot(2, 2, 3);
imshow(modifiedImage, []); title('伪装图像');
```

调用 ExtractMultiLSB 函数从伪装图像中提取隐藏的信息。提取的数据将与原始秘密信息的二进制序列匹配。将提取出的数据重新构造为与原始秘密信息图像相同的二维结构，然后显示在图像处理窗口的第四个位置。

```
extractedData = ExtractMultiLSB(modifiedImage, 2, numel(y));
extractedImage = reshape(extractedData, size(y));
subplot(2, 2, 4);
imshow(extractedImage, []); title('提取出的信息图像');
```

2. 信息隐藏函数 HideMultiLSB

HideMultiLSB 函数是一个 MATLAB 函数，旨在将二进制数据嵌入到载体图像的多个最低有效位（LSB）中。这个函数采用了详尽的逐像素和逐位方法来确保数据的正确嵌入。

(1) 函数定义和参数初始化

```
function modifiedImage = HideMultiLSB(x, numLSBBits, data)
idx = 1;
[m, n, p] = size(x);
```

- x：输入的载体图像，可以是灰度或彩色图像。
- numLSBBits：要用于信息嵌入的 LSB 位的数量。例如，2 表示使用像素值的最低两位。

- data：要嵌入的二进制数据序列。
- idx：这是一个索引变量，用于追踪当前嵌入的数据位在 data 数组中的位置。
- m, n, p：这些变量分别代表图像的行数、列数和颜色通道数。对于彩色图像，p 通常为 3（RGB）；对于灰度图像，p 为 1。

嵌入过程

代码外两层循环遍历图像的所有像素，内层循环遍历每个像素的颜色通道。变量 pixel 存储了当前处理的单个像素值。对于每个像素的每个颜色通道，内部的循环根据 numLSBBits 的值决定要修改的 LSB 位数。例如，如果 numLSBBits 是 2，则修改最低的两位。此 MATLAB 函数用于设置（修改）特定位的值。在此代码中，它用于将像素的特定 LSB 设置为 data 中的相应值。idx 用于确保只有当有数据可嵌入时才进行位设置操作。一旦 data 中的所有数据都已经嵌入，就不再修改像素位。

```
for i = 1:m
    for j = 1:n
        for k = 1:p
            pixel = x(i, j, k);
            for bit = 1:numLSBBits
                if idx <= length(data)
                    pixel = bitset(pixel, bit, data(idx));
                    idx = idx + 1;
                end
            end
            x(i, j, k) = pixel;
        end
    end
end
```

这种方法的主要优点是它允许在图像中灵活地嵌入大量数据，同时提供了直接控制嵌入深度的能力。不过，因为涉及多重循环和位操作，它在处理大型图像或大量数据时可能会有性能考虑。

3. 信息提取函数 ExtractMultiLSB

ExtractMultiLSB 函数通过逐像素和逐位地访问图像中的数据，确保能够准确地提取出之前嵌入的信息。此函数与嵌入函数配合使用，可以实现数据的隐秘传输和恢复。该技术的关键优点是能够在保持图像视觉不变性的同时，隐藏大量的数据。

```
function data = ExtractMultiLSB(x, numLSBBits, numDataBits)
    idx = 1;
    data = zeros(1, numDataBits);
    [m, n, p] = size(x);
    for i = 1:m
        for j = 1:n
            for k = 1:p
                pixel = x(i, j, k);
                for bit = 1:numLSBBits
                    if idx <= numDataBits
                        data(idx) = bitget(pixel, bit);
```

```
                        idx = idx + 1;
                    end
                end
            end
        end
    end
end
```

上述代码遍历图像的每一个像素及其颜色通道。这确保了所有可能包含隐藏数据的部分都被检查。pixel 变量存储了当前正在处理的像素值。内层循环处理每个像素的指定 LSB 位数（由 numLSBBits 定义）。这些位是在嵌入数据时被修改的。这个 MATLAB 函数用来获取特定位的值。在这里，它被用来从每个像素的指定 LSB 中提取数据。通过 idx 来确保数据提取过程不会超出预期的数据长度。一旦达到 numDataBits，循环将不再提取新的数据。

(三) 结果展示

图 2: 实验结果

五、 实验心得体会

在完成本次信息隐藏技术实验后，我深刻体会到了数字图像处理和信息安全领域中技术细节的复杂性与重要性。本实验主要通过 MATLAB 实现了奇偶校验位隐藏法，并对多位 LSB 嵌入方法进行了探索和实验，增强了我在实践中解决问题和应用理论知识的能力。通过实验，我能够将课堂上学习的理论知识与实际编程技能结合起来，具体体现在如何将秘密信息有效地隐藏在图像文件中，以及如何从已修改的图像文件中提取这些信息。这不仅仅是编程技能的锻炼，更是对信息隐藏理论的深入理解。在扩展实验中，我尝试了多位 LSB 嵌入方法，这比传统的单一 LSB 方法可以隐藏更多的数据，同时对图像的影响仍然非常小。通过修改像素的最低两位，我成功地增加了嵌入数据的容量，同时保持了图像质量的高度保真。这种方法在提高数据隐藏容量的同时，也增加了嵌入数据的安全性，因为多位的变化比单一位更难通过肉眼观察到。

通过这次实验，我不仅学到了多位 LSB 嵌入技术的具体实施方法，还对信息隐藏领域的应用和重要性有了更深的认识。实验不仅提升了我的技术能力，也激发了我对信息安全技术深入研究的兴趣。未来我希望能将这些知识应用于更广泛的领域中，探索更多的可能性。