



南開大學
Nankai University

网络空间安全学院
信息隐藏技术实验报告

图像的 **LSB** 隐藏法实验

姓名：孙璐

学号：2112060

专业：信息安全

2024 年 4 月 14 日

目录

| | |
|-------------------------------------|-----------|
| 1 实验要求 | 2 |
| 2 实验原理 | 2 |
| 2.1 LSB 方法 | 2 |
| 2.2 流载体的 LSB 方法 | 2 |
| 2.3 隐藏位置子集的方法 | 2 |
| 2.3.1 顺序选取 | 2 |
| 2.3.2 随机间隔法 | 3 |
| 2.3.3 伪随机置换 | 3 |
| 2.4 提高 LSB 方法安全性的措施 | 3 |
| 3 二值图像嵌入到位图 | 3 |
| 3.1 实验原理 | 3 |
| 3.2 Matlab 源码 | 4 |
| 3.2.1 Matlab 中的 bitget 函数 | 5 |
| 3.2.2 Matlab 中的 bitset 函数 | 5 |
| 3.3 实验结果展示 | 6 |
| 4 将学号（一个整数）嵌入到位图 | 7 |
| 4.1 实验原理 | 7 |
| 4.2 Matlab 源码 | 7 |
| 4.2.1 Matlab 中的 bitget 函数 | 8 |
| 4.2.2 Matlab 中的 bitset 函数 | 9 |
| 4.3 实验结果展示 | 9 |
| 5 实验心得体会 | 10 |

1 实验要求

实验 6: 图像的 LSB 隐藏法实验

内容:

- 1、实现将二值图像嵌入到位图中;
- 2、实现将学号 (一个整数) 嵌入到位图中。

2 实验原理

2.1 LSB 方法

LSB(Least Significant Bit) 最低有效位/最低有效位方法: 用秘密信息 (bit) 替换掉最低有效位的数据。

对空域的 LSB 做替换, 用来替换 LSB 的序列就是需要加入的水印信息、水印的数字摘要或者由水印生成的伪随机序列。由于水印信息嵌入的位置是 LSB, 为了满足水印的不可见性, 允许嵌入的水印强度不可能太高。然而针对空域的各种处理, 如游程编码前的预处理, 会对不显著分量进行一定的压缩, 所以 LSB 算法对这些操作很敏感。

符号约定

- c_i : 载体对象的取样值
- $L(c)$: 载体的长度
- $L(m)$: 秘密信息的长度
- s_i : 伪装对象的取样值
- k : 伪装密钥

2.2 流载体的 LSB 方法

- 嵌入

选择一个载体元素的子集 $j_1, j_2, \dots, j_{L(m)}$, 其中共有 $L(m)$ 个元素, 用以隐藏秘密信息的 $L(m)$ 个比特。然后在这个子集上执行替换操作, 把 C_{j_i} 的最低比特用 m_i 来替换。

- 提取

找到嵌入信息的伪装元素的子集 $j_1, j_2, \dots, j_{L(m)}$, 从这些伪装对象 S_{j_i} , 中抽出它们的最低比特位, 排列之后组成秘密信息。

2.3 隐藏位置子集的方法

2.3.1 顺序选取

发送者从载体的第一个元素开始, 顺序选取 $L(m)$ 个元素作为隐藏的子集。

但载体的已修改部分和未修改部分, 具有不同的统计特性。针对这一缺点, 有以下解决办法

- 嵌入秘密信息后, 继续嵌入伪随机序列

- 重复嵌入秘密信息，直到载体结束
- 随机间隔法：双方利用相同的伪随机数发生器和伪装密钥，确定隐藏位置

2.3.2 随机间隔法

用伪装密钥 k 作为伪随机数发生器的种子，生成一个伪随机序列 $k_1, k_2, \dots, k_{L(m)}$ ，嵌入位置为

$$\begin{cases} j_1 = k_1 \\ j_i = j_{i-1} + k_i \quad i \geq 2 \end{cases}$$

2.3.3 伪随机置换

能够得到载体的所有元素 (例如：载体为图像)，就可以从整个载体考虑，把秘密信息比特随机地分散在整个载体中

- 嵌入
用种子密钥产生一个伪随机索引序 $j_1, j_2, \dots, j_{L(m)}$ ，将第 k 个秘密消息比特隐藏在索引为 j_k 的载体元素的最低比特位中
- 提取
用同样的种子密钥产生同样的伪随机索引序列，从这些位置最低比特位中提取信息。

伪随机置换产生的索引值可能产生碰撞；用一个集合记录所有已使用过的载体索引值，当再次出现同样的索引值时，则放弃这个索引值，再选择下一个元素；提取时做同样处理。

2.4 提高 LSB 方法安全性的措施

LSB 方法的优点是简单，易于实现，容量大。

但缺点是安全性不高，不能抵抗叠加噪声、有损压缩等破坏

提高 LSB 方法的安全性的措施如下：

- 对秘密信息先加密后再隐藏
- 多次重复嵌入
- 引入纠错编码技术。先进性纠错编码，再进行隐藏

3 二值图像嵌入到位图

3.1 实验原理

主要分为以下步骤：

- 将秘密信息转为二进制
- 将原始图像的像素值转为二进制
- 将二进制值秘密信息中的每一 bit 信息替换与之相对应的载体数据的最低有效位

- 将得到的含秘密信息的二进制数据转换为十进制像素值，从而获得含秘密信息的数字图像
- 接收方通过提取数字图像的最低有效位，即可获取二进制的秘密信息，将其转为原始信息并获取秘密信息

3.2 Matlab 源码

载体图像为 boy.bmp，水印图像为 snowman.bmp。Embedding 将水印图像嵌入到载体图像中，Decode 函数从嵌入后的图像中提取出原始水印图像。

在 Embedding 函数中，载体图像的行数和列数分别为 Mc 和 Nc，WaterMarked 是一个全零矩阵，用于存储嵌入水印后的图像。遍历载体图像的每个像素点，通过 bitset 函数将水印图像每个像素点最低有效位替换载体图像对应像素点的最低有效位。

在 Decode 函数中，嵌入水印后的图像的行数和列数分别是 Mc 和 Nc，WaterMark 是一个全零矩阵，用于存储提取出的水印图像。遍历 embedding 后生成的嵌入水印后的图像的每个像素点，使用 bitget 函数获取嵌入水印后的图像对应像素点的最低有效位，存储在 WaterMark 的对应位置。

Listing 1: 二值图像嵌入到位图

```

1
2 function ImageHiding()
3     x=imread("boy.bmp"); %载体图像
4     m=imread("snowman.bmp"); %水印图像
5     imshow(x,[])
6     title("boy_origin.bmp")
7     imshow(m,[]);
8     title("snowman_origin.bmp")
9     WaterMarked=Embedding(x,m);% 嵌入水印
10    watermark=Decode(WaterMarked);
11 end
12
13 function WaterMarked = Embedding(origin,watermark)
14     [Mc,Nc]=size(origin);
15     % 创建一个与载体图像大小一致的全0矩阵
16     WaterMarked=uint8(zeros(size(origin)));
17
18     for i=1:Mc
19         for j=1:Nc
20             % 将水印图像的每个像素点的最低有效位
21             % 嵌入到载体图像中对应像素点的最低有效位
22             WaterMarked(i,j)=bitset(origin(i,j),1,watermark(i,j));
23         end
24     end
25
26     imwrite(WaterMarked,'lsb_embedded_image_watermark.bmp','bmp');
```

```
27     figure;
28     imshow(WaterMarked, []);
29     title("The embedded watermarked image");
30 end
31
32 % 提取伪装图像的最低位平面，恢复隐藏的图像
33 function WaterMark=Decode(WaterMarked)
34     [Mc,Nc]=size(WaterMarked);
35     % 创建一个与嵌入水印后的图像大小一致的全0矩阵
36     WaterMark=uint8(zeros(size(WaterMarked)));
37
38     for i=1:Mc
39         for j=1:Nc
40             %
41             % 提取每个像素点的最低有效位，并存储到watermaek矩阵的对应位置中
42             WaterMark(i,j)=bitget(WaterMarked(i,j),1);
43         end
44     end
45
46     imwrite(WaterMark, 'lsb_extracted_image_watermark.bmp', 'bmp');
47     figure;
48     imshow(WaterMark, []);
49     % title("The decoded watermark image")
50 end
```

3.2.1 Matlab 中的 bitget 函数

- 语法

b = bitget(A,bit)

b = bitget(A,bit,assumedtype)

- 说明

b = bitget(A,bit) 返回整数数组 A 中位于位置 bit 的位值。

b = bitget(A,bit,assumedtype) 假定 A 属于 assumedtype。

tmp(i,j) = bitget(X(i,j),k);

bitget 函数首先将 X(i,j) 处灰度值分解为二进制串，然后取第 k 位

3.2.2 Matlab 中的 bitset 函数

bitset 函数的功能是设置指定位置的位。

- 语法

```
intout = bitset(A,bit)
```

```
intout = bitset(A,bit,assumedtype)
```

```
intout = bitset(A,bit,V)
```

```
intout = bitset(A,bit,V,assumedtype)
```

- 说明

`intout = bitset(A,bit)` 返回 A 的值, 并将位置 bit 设置为 1 (打开)。

`intout = bitset(A,bit, assumedtype)` 假定 A 的类型为 `assumedtype`。

`intout = bitset(A,bit,V)` 返回 A, 并将位置 bit 设置为 V 的值。

a) 如果 V 为零, 则位的位置 bit 将设置为 0 (关闭)。

b) 如果 V 为非零, 则位的位置 bit 将设置为 1 (打开)。

`intout = bitset(A,bit,V, assumedtype)` 假定 A 的类型为 `assumedtype`。

3.3 实验结果展示

原始图像:



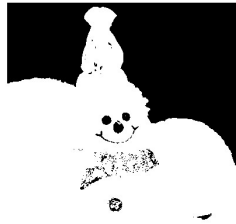
水印图像:



将水印图像的每个像素点最低有效位嵌入到载体图像对应像素点最低位



恢复隐藏的图像



4 将学号（一个整数）嵌入到位图

4.1 实验原理

主要分为以下步骤：

- 将秘密数字信息转为二进制
- 将原始图像的像素值转为二进制
- 使用 bitset 将要嵌入的整数的二进制嵌入到载体图像像素点的最低有效位中
- 将得到的含秘密信息的二进制数据转换为十进制像素值，从而获得含秘密信息的数字图像
- 接收方通过提取数字图像的最低有效位，即可获取二进制数字的秘密信息，再将其转为原始信息即可获取秘密信息

4.2 Matlab 源码

嵌入的整数信息为 2112060，使用 Embedding 函数将整数信息嵌入到载体图像中，再使用 Decode 从嵌入信息中提取出原始的整数信息。针对整数 2112060，将整数的二进制前 22 位嵌入即可。

在 Embedding 函数中，载体图像的行数和列数分别为 Mc 和 Nc，WaterMarked 是一个全零矩阵，用于存储嵌入水印后的图像。遍历载体图像的每个像素点，通过 bitset 函数将水印图像每个像素点最低有效位替换载体图像对应像素点的最低有效位。

在 Decode 函数中，message 初始化为 0 用于存储提取出的整数。遍历 embedding 后生成的嵌入水印后的图像的每个像素点，使用 bitget 函数获取嵌入水印后的图像对应像素点的最低有效位，存储在 message 的对应二进制位。

Listing 2: 将学号（一个整数）嵌入到位图

```
1
2 function IntHiding()
3     x=imread("boy.bmp"); %载体图像
4     m=2112060
5     imshow(x,[])
6     WaterMarked=Embedding(x,m);
7     message=Decode(WaterMarked)
8 end
```



```
9
10 function WaterMarked = Embedding(origin, watermark)
11     [Mc, Nc] = size(origin);
12     % 创建一个与载体图像大小一致的全0矩阵
13     WaterMarked = uint8(zeros(size(origin)));
14
15     for i = 1:Mc
16         for j = 1:Nc
17             if i == 1 && j <= 22
18                 % 将嵌入整数对应二进制位
19                 % 嵌入到载体图像中对应像素点的最低有效位
20                 bit = bitget(watermark, j);
21                 WaterMarked(i, j) = bitset(origin(i, j), 1, bit);
22             else
23                 WaterMarked(i, j) = origin(i, j);
24             end
25         end
26     end
27
28     imwrite(WaterMarked, 'lsb_embedded_int_watermarked.bmp', 'bmp');
29     figure;
30     imshow(WaterMarked, []);
31     title("The embedded watermarked image");
32 end
33
34 function message = Decode(WaterMarked)
35     message = 0;
36     for j = 1:22
37         bit = bitget(WaterMarked(1, j), 1);
38         message = bitset(message, j, bit);
39     end
40 end
```

4.2.1 Matlab 中的 bitget 函数

- 语法

b = bitget(A, bit)

b = bitget(A, bit, assumedtype)

- 说明

b = bitget(A, bit) 返回整数数组 A 中位于位置 bit 的位值。

b = bitget(A, bit, assumedtype) 假定 A 属于 assumedtype。

```
tmp(i,j) = bitget(X(i,j),k);
```

bitget 函数首先将 $X(i,j)$ 处灰度值分解为二进制串，然后取第 k 位

4.2.2 Matlab 中的 bitset 函数

bitset 函数的功能是设置指定位置的位。

- 语法

```
intout = bitset(A,bit)
```

```
intout = bitset(A,bit,assumedtype)
```

```
intout = bitset(A,bit,V)
```

```
intout = bitset(A,bit,V,assumedtype)
```

- 说明

`intout = bitset(A,bit)` 返回 A 的值，并将位置 bit 设置为 1（打开）。

`intout = bitset(A,bit, assumedtype)` 假定 A 的类型为 `assumedtype`。

`intout = bitset(A,bit,V)` 返回 A ，并将位置 bit 设置为 V 的值。

a) 如果 V 为零，则位的位置 bit 将设置为 0（关闭）。

b) 如果 V 为非零，则位的位置 bit 将设置为 1（打开）。

`intout = bitset(A,bit,V, assumedtype)` 假定 A 的类型为 `assumedtype`。

4.3 实验结果展示

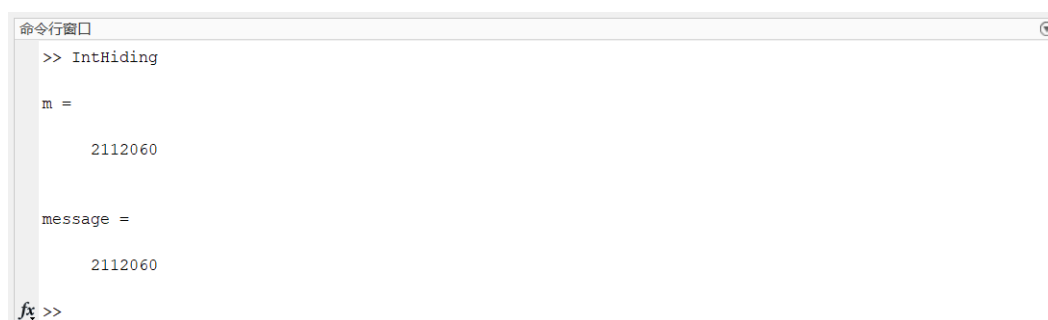
原始图像：



嵌入秘密信息后的图像



提取出的秘密信息



```
命令窗口
>> IntHiding

m =

    2112060

message =

    2112060

fx >>
```

5 实验心得体会

本次实验主要涉及了数字图像隐写术中的 LSB 方法，通过将秘密信息嵌入到图像的最低有效位中实现信息隐藏。实验中分别实现了将二值图像和一个整数（学号）嵌入到位图中，并成功提取出隐藏的信息。通过实验，对 LSB 算法的原理和实现方法有了更深入的了解。

本次实验中使用 MATLAB 来处理数字图像，练习使用了 MATLAB 的基本函数，如 `bitset()` 和 `bitget()`，并且理解了 LSB 算法的基本步骤。LSB 算法虽然简单易懂，但在实际应用中需要注意安全性问题，因为它容易受到攻击者的检测，需要结合其他隐写术和加密算法来提高信息的安全性。

本次实验不仅帮助我加深了对数字图像处理和隐写术的理解，还提高了我在 MATLAB 编程方面的实践能力。通过实验，我深刻体会到了理论知识与实践操作的结合对于知识的巩固和应用的重要性，也更加深刻地认识到了信息安全在数字化时代的重要性。