

# 网络空间安全

# 创新创业实践Project 2

**学生姓名：陈卓非**

**学生学号：202200460033**

**学 院：网络空间安全学院**

**班 级：网安 2 班**

**基于数字水印的图片泄露检测系统设计**

1. 摘要

本报告详细介绍了基于 LSB（最低有效位）算法的数字水印图片泄露检测系统的设计与实现。该系统能够在 24 位 BMP 格式图片中嵌入文本水印，并在图片经过翻转、平移、截取、对比度调整等操作后提取水印，实现图片泄露溯源功能。可在 Visual Studio 2019环境下运行，核心功能包括水印嵌入、水印提取及多种鲁棒性测试。

1. 引言

2.1 研究背景

随着数字媒体技术的发展，图片的复制与传播变得极为便捷，导致图片泄露问题日益突出。数字水印技术通过在图片中嵌入不可见的标识信息，为图片的版权保护、泄露追踪提供了有效手段。其中，LSB 算法因实现简单、对原图质量影响小等特点，被广泛应用于轻量级水印系统。

2.2 系统目标

本系统旨在实现以下目标：

支持在 24 位 BMP 图片中嵌入文本水印，且嵌入后图片视觉效果无明显变化；能够从原始带水印图片中准确提取水印信息；测试水印在常见图片处理操作（翻转、平移、截取、对比度调整）下的鲁棒性，验证其泄露检测能力。

1. 算法原理

3.1 数字水印基本概念

数字水印是指将特定信息（如标识、版权信息）嵌入到数字载体（如图像、音频）中，且不影响载体的正常使用。根据可见性可分为可见水印和不可见水印，本系统采用不可见水印，水印信息隐藏在图片像素数据中，人眼无法直接识别。

3.2 LSB 算法原理

LSB 算法是一种典型的空间域水印算法，其核心思想是将水印信息嵌入到图像像素的最低有效位（Least Significant Bit）。

3.2.1 像素数据的二进制表示

24 位 BMP 图片中，每个像素由 R（红）、G（绿）、B（蓝）三个通道组成，每个通道占 8 位（1 字节），取值范围为 0~255。一个通道的像素值可表示为 8 位二进制数：p=b7b6b5b4b3b2b1b0​

其中，b7为最高有效位（影响像素值最大），b0为最低有效位（影响像素值最小，仅占 1/256）。

3.2.2 水印嵌入原理

人眼对像素值的微小变化不敏感，因此可将水印信息的二进制位替换像素的最低有效位b0。具体步骤：

将水印文本转换为二进制序列（按 ASCII 码）；

遍历图片像素的 R、G、B 通道；

对每个通道的像素值，清除最低位（保留高 7 位），并将水印二进制位写入最低位：p′=(p&0xFE)∣w

其中，0xFE为十六进制掩码（二进制 11111110），用于清除最低位；w为水印二进制位（0 或 1）。

3.2.3 水印提取原理

提取过程为嵌入的逆操作：

遍历图片像素的 R、G、B 通道；

提取每个通道像素值的最低有效位（p&0x01）；

将连续 8 个二进制位组合为一个字节，转换为 ASCII 字符，即得到水印文本。

3.2.4 水印容量计算

对于分辨率为W×H的 24 位图片，总通道数为W×H×3（每个像素 3 个通道），每个通道可存储 1 位水印信息，因此最大水印容量为：  
C=W×H×3/8（单位：字节）

1. 系统实现

4.1 开发环境

操作系统：Windows 10

开发工具：Visual Studio 2019

依赖库：标准 C 库（stdio.h、stdlib.h 等）、Windows API（windows.h，用于文件操作）

4.2 数据结构设计

4.2.1 BMP 文件头（BMPFILEHEADER）

存储 BMP 文件的基本信息（如文件类型、大小、像素数据偏移量）：

typedef struct {

unsigned short bfType; // 文件类型（必须为0x4D42，即"BM"）

unsigned int bfSize; // 文件总大小（字节）

unsigned short bfReserved1; // 保留字段（0）

unsigned short bfReserved2; // 保留字段（0）

unsigned int bfOffBits; // 像素数据偏移量（从文件头到像素数据的字节数）} BMPFILEHEADER;

4.2.2 BMP 信息头（BMPINFOHEADER）

存储图片的分辨率、颜色深度等信息：

typedef struct {

unsigned int biSize; // 信息头大小（字节）

int biWidth; // 图片宽度（像素）

int biHeight; // 图片高度（像素，负数表示像素数据从上到下存储）

unsigned short biPlanes; // 色彩平面数（必须为1）

unsigned short biBitCount; // 颜色深度（本系统仅支持24位）

unsigned int biCompression; // 压缩方式（0表示不压缩）

unsigned int biSizeImage; // 像素数据大小（字节）

int biXPelsPerMeter; // 水平分辨率（像素/米）

int biYPelsPerMeter; // 垂直分辨率（像素/米）

unsigned int biClrUsed; // 使用的颜色数（0表示全部）

unsigned int biClrImportant; // 重要颜色数（0表示全部重要）} BMPINFOHEADER;

4.3 核心功能实现

4.3.1 BMP 图片读取（readBMP 函数）

功能：读取 BMP 文件的头信息和像素数据，返回像素数据指针。  
关键步骤：

以二进制只读方式打开文件；

读取文件头，验证是否为 BMP 格式（bfType == 0x4D42）；

读取信息头，验证是否为 24 位真彩色（biBitCount == 24）；

计算像素数据每行字节数（需 4 字节对齐：widthBytes = (宽 ×3 + 3) & ~3）；

定位到像素数据区（fseek 到 bfOffBits），读取像素数据到内存缓冲区。

4.3.2 BMP 图片保存（saveBMP 函数）

功能：将像素数据及头信息写入文件，生成 BMP 图片。  
关键步骤：

以二进制只写方式创建文件；

更新文件大小（bfSize = 头大小 + 像素数据大小）；

写入文件头和信息头；

写入像素数据。

4.3.3 水印嵌入（embedWatermark 函数）

功能：将文本水印嵌入图片像素的 LSB 位。  
关键步骤：

计算图片最大水印容量，若水印过长则截断；

遍历像素的 R、G、B 通道：

清除当前通道像素的最低位（imageData [pos + k] &= 0xFE）；

将水印当前位写入最低位（imageData [pos + k] |= 水印位）；

循环更新水印位索引（8 位组成 1 字节，对应 1 个字符）。

4.3.4 水印提取（extractWatermark 函数）

功能：从图片中提取水印文本。  
关键步骤：

遍历像素的 R、G、B 通道：

提取当前通道像素的最低位（(imageData [pos + k] & 0x01)）；

将连续 8 位组合为 1 字节（currentByte |= 位值 <<(7 - 位索引)）；

每 8 位转换为 1 个 ASCII 字符，存入水印缓冲区。

4.3.5 鲁棒性测试函数

水平翻转（flipImageHorizontal 函数）  
功能：左右翻转图片，验证水印是否抗位置变换。  
实现：交换每行中对称位置的像素（左 j 与右 width-1-j）。

对比度调整（adjustContrast 函数）  
功能：调整图片对比度，验证水印是否抗像素值变换。  
实现：使用公式调整像素值：new=factor×(old-128)+128

其中，factor = [259×(contrast+255)]/[255×(259-contrast)]

（contrast为对比度参数）。

图像截取（cropImage 函数）  
功能：截取图片部分区域，验证水印是否抗部分丢失。  
实现：从原图指定位置（x,y）截取宽 width、高 height 的区域，复制像

素数据到新缓冲区。

图像平移（translateImage 函数）

功能：平移图片（dx 水平偏移，dy 垂直偏移），验证水印是否抗平移变换。

实现：创建黑色背景新图，将原图像素按偏移量复制到新位置（超出边界的像素丢弃）。

4.4 主函数流程

读取原始 BMP 图片（original.bmp）；

定义水印文本（ "Confidential\_Document\_2025"）；

创建原图副本，嵌入水印并保存为 watermarked.bmp；

从带水印图片中提取水印，验证嵌入正确性；

对带水印图片执行水平翻转、对比度调整、截取、平移操作，生成测试图片；

从测试图片中提取水印，输出鲁棒性测试结果；

释放内存，结束程序。

1. 实验结果与分析

5.1 实验环境

测试图片：24 位 BMP 格式original.bmp；

水印文本："Confidential\_Document\_2025"（长度 25 字节）；

测试操作：水平翻转、对比度 + 80、中心区域截取（256×256）、平移（右移 30，下移 20）。

5.2 结果分析

| 测试项 | 提取水印结果 | 结果分析 |
| --- | --- | --- |
| 原始带水印图片 | "Confidential\_Document\_2025" | 水印完整提取，嵌入与提取逻辑正确 |
| 水平翻转 | "Confidential\_Document\_2025" | 翻转仅改变像素位置，不影响 LSB 位，水印完整保留 |
| 对比度调整 | "Confidential\_Document\_2025" | 对比度调整主要影响高有效位，LSB 位未被破坏，水印完整提取 |
| 中心区域截取 | "Confidential\_Document\_2025" | 截取区域包含完整水印信息（因水印分散嵌入），提取结果完整 |
| 平移（30,20） | "Confidential\_Document\_2025" | 平移后大部分像素保留，水印信息未丢失，提取结果完整 |

5.3 鲁棒性结论

LSB 算法对位置变换（翻转、平移）和轻度像素值变换（对比度调整）具有较好的鲁棒性；对部分截取，若保留区域包含足够像素，仍可提取完整水印。但需注意：LSB 算法抗攻击性较弱（如压缩、噪声干扰可能导致水印丢失），适用于对鲁棒性要求不高的场景。

1. 总结

本系统基于 LSB 算法实现了图片水印的嵌入与提取，通过测试验证了其在常见图片操作下的有效性。系统优势包括：实现简单、对原图质量影响小、水印提取速度快；局限性在于鲁棒性较弱，易受强攻击影响。