华中科技大学

2023

网络空间安全概论　　结题报告

题　　目：	信息隐藏研究
专　　业：	计算机科学与技术
班　　级：	CS2205
学　　号：	U202215494
姓　　名：	廖奎
指导教师：	郝义学
邮　　件：	2967845704@qq.com

目　录

1 引言

1.1 研究背景

随着信息技术的飞速发展，数据的传输、存储和处理变得越来越普遍，但同时也带来了数据泄露、篡改和非法访问等安全问题，数字内容的可复制性和易传播性使得版权保护问题尤为突出。传统的加密技术虽然能够有效保护数据的机密性，但在某些情况下，加密本身可能会引起攻击者的注意，从而增加被破解的风险。此外，加密技术无法解决数字内容的版权保护问题，因为加密后的数据在解密后仍然可以被轻易复制和传播。

因此信息隐藏技术应运而生，通过在数字媒体中嵌入不可见的信息，如版权标识、用户身份等，确保内容创作者的权益得以保护。这种嵌入的信息可以在不影响载体正常使用的情况下，提供额外的安全保障。例如，在图像、音频或视频中嵌入水印，不仅能够验证内容的来源和完整性，还能防止未经授权的使用和分发。此外，信息隐藏技术还可以用于隐蔽通信，通过将秘密信息嵌入到看似无害的数据中，实现安全的信息传输，从而在信息安全、版权保护和隐私维护等方面带来有效的解决方案

1.2 研究问题与应用前景

信息隐藏技术主要解决的是信息安全、版权保护和隐私保护等方面的问题，具体应用于以下几个方面：

(1) 版权保护： 通过在数字媒体（如图像、音频、视频）中嵌入不可见的水印，信息隐藏技术可以验证内容的来源和完整性，防止盗版和非法复制，从而保护内容创作者的权益。

(2) 隐蔽通信： 信息隐藏技术可以在看似无害的数据中嵌入秘密信息，实现安全的信息传输。这种隐蔽通信方式在军事和情报领域尤为重要，能够确保敏感信息不被察觉和拦截。

(3) 数据认证： 通过在数据中嵌入特定的标识或签名，信息隐藏技术可以验证数据的完整性和来源，防止数据被篡改。这对于确保数据的真实性和可靠性非常关

键。

(4) 隐私保护： 在社交媒体和大数据分析中，信息隐藏技术可以用于保护用户的个人隐私。例如，在用户上传的照片或视频中嵌入匿名标识，以防止个人信息泄露。

(5) 数字取证： 信息隐藏技术可以帮助在电子证据中嵌入时间戳或其他标识，确保证据的真实性和不可篡改性。这在法律和执法领域非常重要，有助于追踪和验证电子证据。

(6) 多媒体内容管理： 通过在多媒体内容中嵌入元数据，信息隐藏技术可以提供额外的内容管理和检索功能。例如，在视频中嵌入时间戳或地理位置信息，便于后续的搜索和分析。

(7) 防伪技术： 信息隐藏技术可以用于防伪标签和产品包装中，通过嵌入唯一的标识符，确保产品的真伪。消费者可以通过扫描这些标识来验证产品的真实性。

2 信息隐藏原理

2.1 信息隐藏定义

信息隐藏，就是在一些载体信息中将需要保密传递的信息隐藏进去，而载体本身并没有太大的变化，不会引起怀疑。其核心思想是利用人类感知系统的局限性和数据冗余性来实现信息的隐蔽传输（根据郝义学老师《网络空间安全概论》课程的第四章课件内容安全的内容）。

信息隐藏由以下几个部分组成：

（1）传递消息双方：A 和 B

（2）载体对象：C（从随机消息源中任意选取）

（3）秘密消息：m

（4）伪装对象：C'

信息隐藏的原理框图如图 2.1 所示。

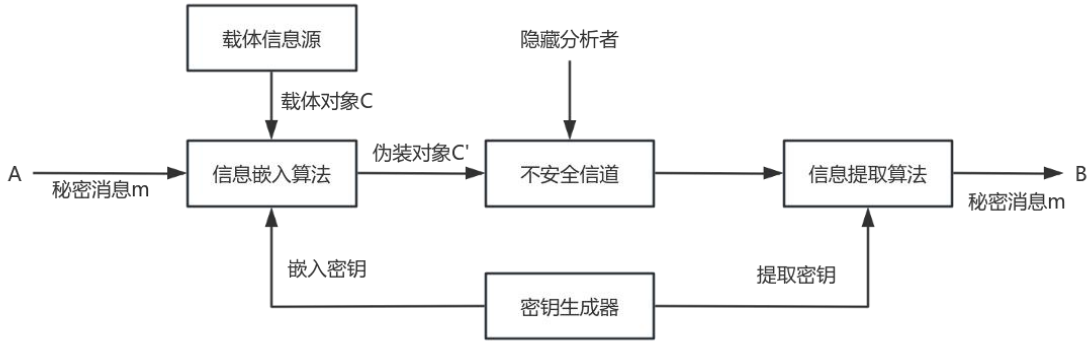

图 2.1 信息隐藏原理框图

C 与 C’在感官上是不可区分。当秘密信息 m 嵌入到载体对象后，伪装对象的视觉感观、听觉感观，或者一般的计算机统计分析都不会发现伪装对象与载体对象有什么区别，这样就实现了信息的隐蔽传输，实现了信息的安全传递。信息隐藏的安全性取决于第三方有没有能力将载体对象和伪装对象区别开来。

2.2 信息隐藏算法的分类及特性

几种常用的信息隐藏算法：

（1）基于空间域的信息隐藏算法

最低有效位算法(LSB)；像素值差分算法。

（2）基于变换域的信息隐藏算法

离散余弦变换(DCT)；离散小波变换(DWT)；离散傅里叶变换(DFT)。

（3）基于扩频技术的信息隐藏算法

扩频水印算法。

（4）基于密码学和混沌理论的信息隐藏算法

混沌映射。

信息隐藏算法的几种性能指标：

（1）透明性（不可感知性）：指嵌入的秘密信息导致隐写载体信号质量变化的程度，是信息隐藏的首要特性。在被保护信息嵌入数字水印后，不会引起隐写载体质量的显著下降和视听效果的明显变化，不能影响隐写载体的正常使用。

（2）鲁棒性（稳健性）：指隐藏的秘密信息抵抗各种信号处理和攻击的能力，鲁棒性水印通常不会因常见的信号处理和攻击而丢失隐藏的水印信息。

（3）隐藏容量：指在单位时间或一副作品中能嵌入水印的比特数。

信息隐藏算法这三个性能指标之间相互制约，没有一个算法能让这三个性能指标同时达到最优。

2.3 信息隐藏的攻击手段

信息隐藏的几种攻击：

（1）被动攻击：监视和破译隐藏的秘密信息。

（2）主动攻击：破坏隐藏的秘密信息；篡改秘密信息。

（3）非恶意修改：压缩编码，信号处理技术，格式转换等。

伪装载体受到某种攻击后，仍然能够从中提取出隐藏信息，称为算法对这种攻击是健壮的，算法的隐藏容量、安全性、健壮性相互制约。

在本课题中我选取最低有效位算法(LSB)、离散余弦变换(DCT)算法两者进行对比，分析不同的信息隐藏算法的各自特性，并尝试进行改进。

3 信息隐藏算法设计与实现

3.1 最低有效位算法

3.1.1 算法原理

考虑到所有图像及音频对象均可数字化为$\{c_i\}$序列，由若干二进制位组成，最低两位反映的基本上是噪声，去掉最低 2 位甚至最低 4 位对图像的整体视觉效果没有太大影响，去掉最低 1 位对图像的统计特性影响不明显，因此可以利用载体对象的二进制的最低几位来进行秘密信息的隐藏，这就是最低有效位算法（Least Significant Bits）。

利用最低有效位算法隐藏信息，可嵌入的信息量由使用的最低有效位的位数决定，使用的位数越小，则嵌入信息对原图像的影响就越小，但可嵌入数据容量也就越小，因此嵌入的数据容量要根据载体大小调整，保证在不引起隐藏分析者注意的情况下，尽量嵌入更多的信息。

3.1.2 详细设计

不同载体对象的最低有效位算法有所差异，我在这里对三种类型的载体对象的嵌入策略和信息容量进行讨论。

（1）当载体对象为彩色图片时，可以嵌入的信息较多。RGB 三色通道各自有 8 位二进制数，改变最低一位对原数值影响为 1，改变最低第 2 位对原数值影响为 2，第 3 位对原数值影响为 4，依此类推，只需要利用 RGB 中三个通道的最低一位或者两位隐藏秘密消息，就可使对像素值影响较小。

令原图像点阵图大小为：宽 W 像素，高 D 像素，颜色分量数为 3，每一颜色分量用 8 位二进制数表示，使用最低 X 位用于嵌入信息时，可嵌入图像的信息量占比为 X/8，总的二进制位数 C：

$$C=3\times X\times W\times D$$

（2）当载体对象为灰度图像时，人眼不能分辨全部 256 个灰度等级，4 个左右灰度等级的差异人眼是不能区别的，而当对比度比较小时人眼的分辨能力更差。同样将嵌入对象的数据存放到最低的 X 位中来隐藏信息，可嵌入图像的信息量占比仍

为 X/8，但总的二进制位数较彩色图片没有乘以倍数 3。

（3）当载体对象为音频时（如 wav），每个音频样本通常是由多个字节（通常是 2 个字节，即 16 位）表示的，且每个样本表示的是声音的幅度（即音量）在某一时刻的数值。在 16 位音频样本中，最低位也是一个可以修改的位，用于嵌入信息，这样改变值仅为 1，不会显著改变音频的质量。单声道音频每个音频样本在 16 位深度的音频文件中占 2 字节。对于立体声（2 个通道：左、右），每个样本的大小是 4 字节。假设音频文件的时长为 T 秒，采样率为 S（例如 44100Hz），音频是单声道或立体声，每个样本的位数为 16 位，嵌入 1 位信息，嵌入的信息量占比为 1/16，对于单声道音频，嵌入总的二进制位数：S*T，对于立体声音频，嵌入总的二进制位数：2*S*T。

对于图像，数据嵌入的总量则依赖于图像的分辨率以及使用的颜色通道位数。而对于音频文件，数据嵌入的总量主要依赖于音频的时长、采样率、以及选择的最低有效位数。

3.1.3 算法实现

（1）RGB 图片中嵌入黑白图片水印

此算法主要分为三个模块：根据输入的字符串生成白字黑底的黑白水印图片；嵌入水印到彩色图片的 B 通道的最低有效位；提取嵌入信息后的彩色图片中的水印信息。考虑到彩色图片可嵌入容量大，输入字符串不长，因此只需选取三通道之一的 B 通道最低一位即可实现较好的信息隐藏，并且人眼基本看不到区别，因此未实现分别嵌入三个通道的低两位。如若需要隐藏大量信息，则代码需要进行修改，可以不生成黑白水印直接将字符串转换成二进制数，存到三个通道的低两位。此代码使用黑白水印图片而不是字符串是为了后续进行攻击时方便观察。

生成黑白水印图片模块中首先要考虑的问题是图片大小，有两种方案，一是根据输入的字符串长度动态调整图片大小，然后将图片大小保存下来，嵌入、提取信息时用作参数。二是使黑白图片与彩色图片大小一致，方便嵌入时像素点一一对应，在此代码中我使用的是第二种方案。主要过程是根据彩色图片大小创建黑色图片，处理输入的字符串得到文本的位置信息，利用 draw.text 函数绘制白色文字。

嵌入模块首先读取彩色图片并转为 RGB 模式，生成三维数组，读取黑白水印转换为灰度图像，将像素值转换成 0 或 1，嵌入彩色图片对应的 B 通道最低位，保

存嵌入后的图片。

提取模块读取嵌入后的图片，获取 B 通道的最低有效位信息，创建彩色图片大小的黑色图片，使用 putdata 方法将隐藏信息放入图片中，保存水印图片。

算法流程图如图 3.1 所示。

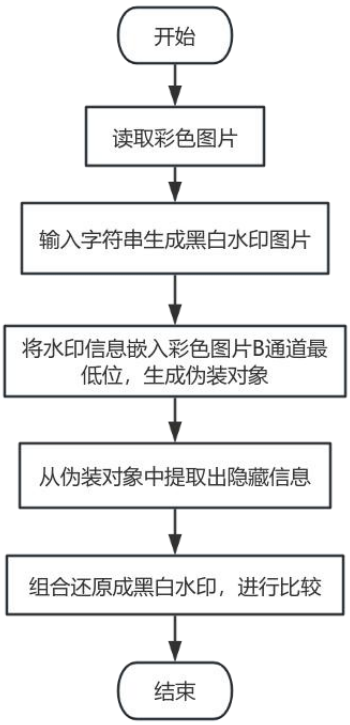

图 3.1 使用 LSB 算法在彩图中隐藏黑白水印流程图

关键代码如下。

水印嵌入：

```
for i in range(color_data.shape[0]):
        for j in range(color_data.shape[1]):
            r, g, b = color_data[i, j]
            b = (b & 0b11111110) | watermark_data[i, j]
            color_data[i, j] = (r, g, b)
```

水印提取：

```
watermark_data = image_data[:, :, 2] & 1
watermark_image = Image.new('1', (width, height))
watermark_image.putdata(watermark_data.flatten())
```

（2）wav 音频中嵌入字符串

此算法与上一算法类似，区别在于输入的字符串不需要转换成黑白水印图片，而是直接转换成二进制字符串；嵌入策略也有所不同，读取音频文件将其转换成一维数组，每个数组元素为一帧音频信息，即 16 位二进制串，将待隐藏的信息当前一位保存进一帧的最低位。提取时直接提取出载体音频每帧最低位，这里注意结束标志，人为地加入不易和原二进制消息冲突的结束标志，一段特殊的不易重复的字符串 1011011101111011111011110111011010，将结束标志之前的二进制转换成字符串即可。算法流程图如图 3.2 所示。

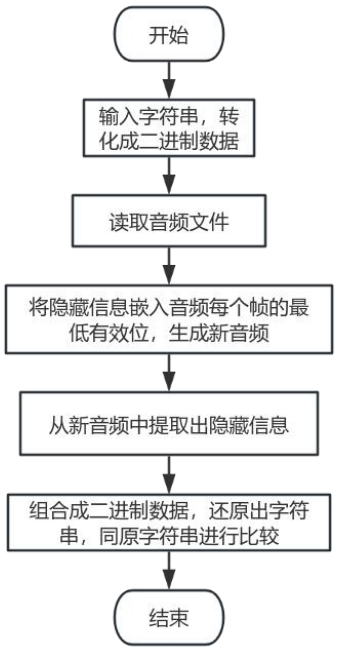

图 3.2 使用 LSB 算法在音频中隐藏字符串流程图

关键代码如下。

数据嵌入：

```
data_bin = string_to_bin(data)#转换成二进制数据
data_length = len(data_bin)
for i in range(data_length):#将每位数据存进每个帧最低位
    bit = int(data_bin[i])
    audio_array[i] = (audio_array[i] & ~1) | bit
```

数据提取：

```
audio_array = np.frombuffer(frames, dtype=np.int16)
```

```
extracted_bits = []
for sample in audio_array:#提取音频样本的最低有效位
    extracted_bits.append(sample & 1)
```

3.1.4 运行测试

运行程序 LSB_image.py 测试在彩图中隐藏黑白水印，输入隐藏消息“Meet at the park”，命令行输出消息如图 3.3 所示，载体图片、生成的水印如图 3.4 所示。伪装图片和提取的水印如图 3.5 所示。

```
(venv) PS D:\NetworkSecurity\code> python LSB_image.py
请输入水印字符串(英文)(例如 Meet at the park): Meet at the park
黑白水印图片已保存到 D:/NetworkSecurity/resources/LSB/LSB_create_watermark.png
嵌入水印后彩色图片已保存到 D:/NetworkSecurity/resources/LSB/LSB_embedded.png
提取的水印图片已保存到 D:/NetworkSecurity/resources/LSB/LSB_get_watermark.png
PSNR值： 55.9108064445004444 dB
SSIM值： 0.9999871235472145
操作完成！
```

图 3.3 LSB_image.py 测试图

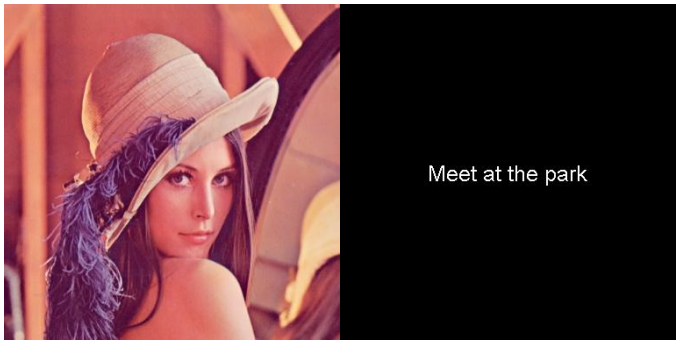

图 3.4 载体图片、生成的水印图

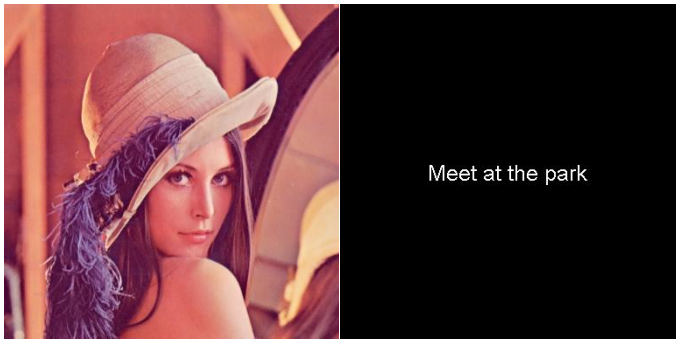

图 3.5 伪装图片、提取的水印图

由图 3.3 可知伪装图片峰值信噪比约为 55.91，可知与原图像相比仅有些许非常小的误差，结构相似度为 99.9987%，区别非常小，同时人眼也看不出彩色图片间的

区别，提取出的水印完好，和初始水印图片相同，算法功能实现。

运行程序 LSB_audio.py 测试在 wav 格式音频中隐藏字符串，输入隐藏消息“1.huster。2.大鹏一日同风起，扶摇直上九万里。3.青青子衿，悠悠我心。”，生成音频保存在 resources 目录下，不便于在报告中展示，命令行输出测试结果如图 3.6 所示。

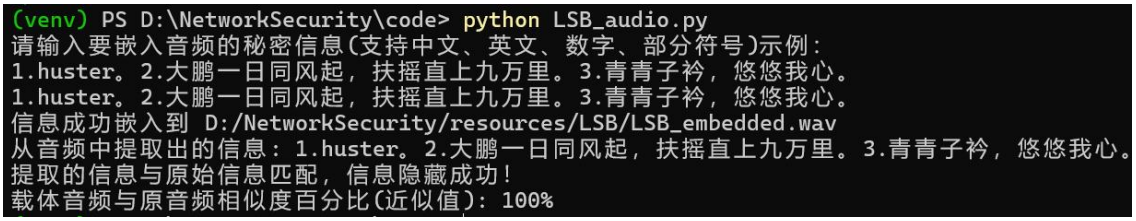

图 3.6 LSB_audio.py 代码测试结果

可见提取的字符串和原隐藏字符串相同，进入文件目录中听取生成音频，未发现和原音频的区别，相似度百分比近似为 1，算法功能实现。

3.2 离散余弦变换算法

3.2.1 算法原理

离散余弦变换算法（Discrete Cosine Transform）是一种线性变换，能够将信号从时域转换到频域，得到信号中包含的频率成分以及各频率分量的大小。它与离散傅里叶变换类似，但是只使用实数。离散余弦逆变换将信号从频域转换到时域，将不同大小的频率分量合成时域信号。

在自然界中，大多数信号的能量都集中在余弦变换后的低频部分。图像的 DCT 系数按频率分布，低频系数代表图像的基本结构（大致的亮度、对比度等），而高频系数代表细节和纹理。中频系数通常包含了图像的细节信息，但不至于过于细微，并且人眼对于细节信息并不敏感，因此中频系数是常用的嵌入信息的位置，使用 DCT 算法进行信息隐藏就是在这个特点之上进行的。

利用 DCT 水印算法对图像进行水印嵌入时，需要先将原始图像数据分成 8×8 数据单元矩阵，然后对数据单元进行离散余弦变换，得到 1 个 8×8 的系数矩阵。为了将隐藏信息与载体图像的视觉重要部分绑定，通常将隐藏信息嵌入系数矩阵的中频系数，达到既不引起视觉变化又不会被轻易破坏的目的。

由于每个数字图像可看作一个实数矩阵，同时数字图像一般为二维，且二维离

散余弦变换一般采用行列分离法来实现，因此对数字图像进行二维离散余弦变换的公式如下：

$$F(u,v)=\frac{2}{N}c(u)c(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)\cos(\frac{\pi u(2x+1)}{2N})\cos(\frac{\pi v(2y+1)}{2N})$$

$$x，y=0，1，\ldots，N-1$$

$$其中\ c(x)=\begin{cases}\frac{1}{\sqrt{2}} & ,当\ x=0\\ 1 & ,其它\end{cases}$$

反 DCT 变换公式：

$$f(x,y)=\frac{2}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}c(u)c(v)F(u,v)\cos(\frac{\pi u(2x+1)}{2N})\cos(\frac{\pi v(2y+1)}{2N})$$

$$u，v=0，1，\ldots，N-1$$

3.2.2 详细设计

此算法中我实现的是在灰度图像 DCT.bmp 中隐藏字符串信息，生成载体图片，在其中提取出字符串与原字符串进行比较，输出峰值信噪比和结构相似度。

算法开始先把图像分成 8×8 的像素块，把待隐藏的字符串信息转换成二进制字符串，随机地选择一个图像块 b_i，它经二维 DCT 变换后得到系数矩阵 B_i，用它对二进制串中第 i 个消息比特 k_i 进行编码。关于如何利用 8×8 系数矩阵 B_i 隐藏一个比特，我这里使用的是比较特定两个位置系数值的大小来承载信息。比如用（u_1,v_1），（u_2,v_2）代表所选定的两个系数的坐标，如果 $B_i(u_1,v_1)>B_i(u_2,v_2)$，就代表这个块隐藏信息为 1，$B_i(u_1,v_1)<B_i(u_2,v_2)$代表隐藏 0，在嵌入时如果大小表示的信息和要隐藏的比特相反，则交换这两个位置的值，并进行微调，增大差距，避免系数过于接近导致提取时出错。嵌入完成后做逆 DCT 变换，将图像变回空间域，保存图像。提取信息时首先要得到嵌入过程中比较的两个系数值的位置，根据大小得出此处隐藏的二进制值，组合转换回字符串，与原始字符串进行比较。

此算法在 8*8 的矩阵中嵌入 1 个比特，数据嵌入比例为 1/64（假设图像像素矩阵大小是 8*8 矩阵的整数倍），假设图像大小为 m*n，则可嵌入数据总量为(m/8)*(n/8)，容量相较 LSB 算法小很多。

3.2.3 算法实现

此算法中最关键部分在于如何选取图像块隐藏比特信息，如果不是随机选取，易导致修改集中，使图片质量下降，随机选取又如何记录或是复现，传递给提取函数使用。在这里我使用的是随机数种子，一是只需传入相同 key 值便可使结果复现，方便提取函数使用；二是生成的位置非固定，随机性使得选取的块分散，对图像质量影响小，不易被观察者发现。函数 randrc 输入包括的行矩阵个数、列矩阵个数、待隐藏比特数以及 key 值，生成随机选取的矩阵行列数组 row、col，嵌入和提取时根据行列数组确定比较的两个数位置。同时要注意到以下两个问题：如果一个块被选择了多次，那么嵌入时会对同一位置进行多次操作，有可能导致数据被篡改或遗失，为了避免这种情况，我在算法中定义了一个集合，每次生成的随机块在集合中查找，重复则丢弃，否则存进集合中；如果图片太小不够存储隐藏字符串，那么需要重新输入字符串。

嵌入操作关键代码如下：

```
k1, k2 = randrc(row, col, count, 12)
for i in range(0, count):
        r = (k1[i] - 1) * 8  # 恢复行索引
        c = (k2[i] - 1) * 8  # 恢复列索引
        if msg[i] == '0':
            if Y[r + 4, c + 1] > Y[r + 3, c + 2]:
                Y[r + 4, c + 1], Y[r + 3, c + 2] = swap(Y[r +
4, c + 1], Y[r + 3, c + 2])
        else:
            if Y[r + 4, c + 1] < Y[r + 3, c + 2]:
                Y[r + 4, c + 1], Y[r + 3, c + 2] = swap(Y[r +
4, c + 1], Y[r + 3, c + 2])
```

提取操作关键代码如下：

```
k1, k2 = randrc(row, col, count, 12)
b = ""
for i in range(0, count):
```

```
r = (k1[i]-1) * 8  # 恢复行索引
c = (k2[i]-1) * 8  # 恢复列索引
if Y[r + 4, c + 1] < Y[r + 3, c + 2]:
    b += '0'
else:
    b += '1'
```

算法流程图如图 3.7 所示。

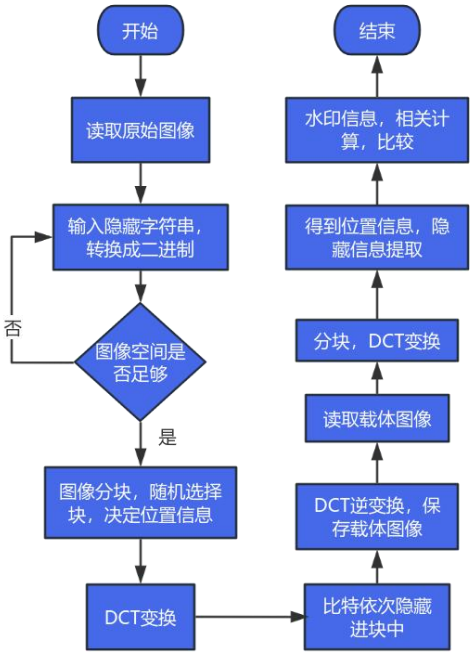

图 3.7 DCT 算法流程图

3.2.4 运行测试

运行 DCT.py 代码，输入字符串“1 人生若只如初见，何事秋风悲画扇。2 众里寻他千百度，蓦然回首，那人却在，灯火阑珊处。3Fight”，运行结果如图 3.8 所示，原始图像和载体图像如图 3.9 所示。

```
(venv) PS D:\NetworkSecurity\code> python DCT.py
读取载体图像D:/NetworkSecurity/resources/DCT/DCT.bmp
请输入要嵌入音频的秘密信息(支持中文、英文，数字)例如:
1人生若只如初见，何事秋风悲画扇。2众里寻他千百度，蓦然回首，那人却在，灯火阑珊处。3Fight
1人生若只如初见，何事秋风悲画扇。2众里寻他千百度，蓦然回首，那人却在，灯火阑珊处。3Fight
图像隐写已完成,隐写后的图像保存为D:/NetworkSecurity/resources/DCT/DCT_embedded.bmp
开始提取，隐写图像路径：D:/NetworkSecurity/resources/DCT/DCT_embedded.bmp
提取的文本信息：1人生若只如初见，何事秋风悲画扇。2众里寻他千百度，蓦然回首，那人却在，灯火阑珊处。3Fight
提取的信息与原始信息匹配，信息隐藏成功！
峰值信噪比PSNR: 44.21002175700125 dB
结构相似度SSIM: 0.9946887620881244
```

图 3.8 DCT.py 测试图

图 3.9 原始图像和载体图像

由上图可知提取的字符串与原始字符串相同，数据隐藏成功。生成的载体图片峰值信噪比为 44.21，说明生成图像与原图像仅有很小的误差，结构相似度 99.469%也说明了这一点，人眼也看不出差别，函数功能实现。

3.3 算法性能评估

3.3.1 鲁棒性 (Robustness)

本报告着重探究实现算法的鲁棒性，即算法对载体的各种攻击或修改（如遮挡、压缩、裁剪等）的抵抗能力。

运行程序 LSB_image_cover.py，对载体图片进行遮挡，生成图片 LSB_cover.png，从其中提取水印，保存为 LSB_coverd_watermark.png，和未遮挡时提取的水印图片进行比较。载体图片遮挡前后比较图如图 3.10 所示，遮挡前后提取水印比较图如图 3.11 所示。

图 3.10 载体图片遮挡前后比较图

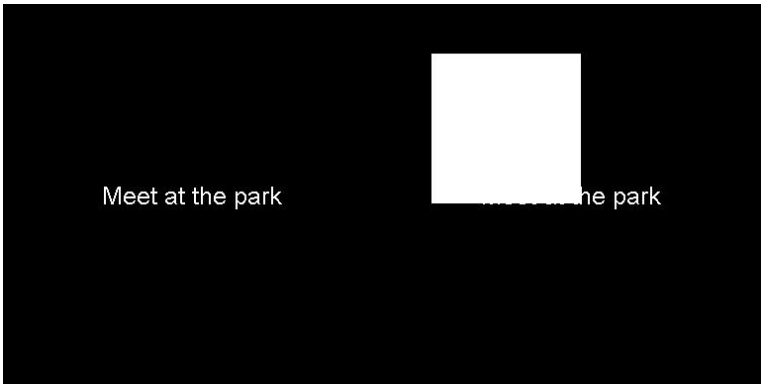

图 3.11 遮挡前后提取水印比较图

由上图可见对 LSB 算法生成的载体图片进行遮挡后，提取出的水印图片对应位置也会变化，显然这是因为遮挡后把对应点像素值变为了 255，最低位变成了 1，提取 1 后还原时自然显示白色。故 LSB 算法对遮挡抵抗性很差。

接着运行程序 LSB_image_compress.py 程序探究 LSB 算法对图像压缩的抵抗性。对载体图片进行压缩，压缩成图片 LSB_compressed.jpg，从其中提取水印，保存为 LSB_compressed_watermark.png，和未压缩时提取的水印图片进行比较。载体图片压缩前后比较图如图 3.12 所示，压缩前后提取水印比较图如图 3.13 所示。

图 3.12 载体图片压缩前后比较图

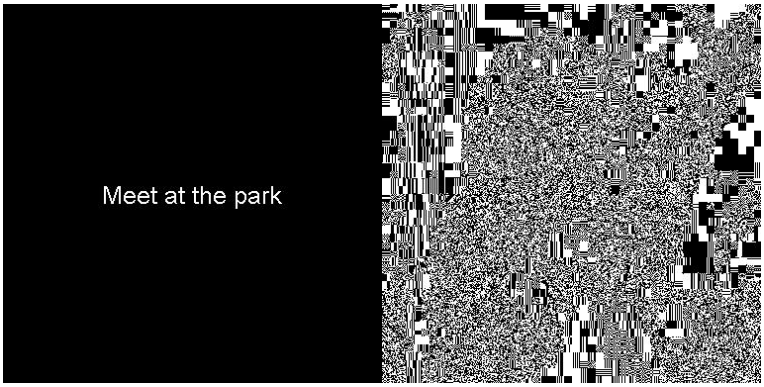

图 3.13 压缩前后提取水印比较图

由上图可见对 LSB 算法生成的载体图片进行压缩后，提取出的水印图片显示杂乱，完全看不出有用信息，这是由于压缩算法通常会去除图像中的冗余信息，以减小文件大小，而 LSB 算法嵌入的信息往往被视为这种冗余信息的一部分。并且在 JPEG 等有损压缩算法中，压缩时会对图像数据进行量化和重新编码，这个过程会改变像素值的原始表示，从而破坏 LSB 算法嵌入的信息。故 LSB 算法对压缩抵抗性也很差。

运行 LSB_audio_comp_trim.py 程序研究压缩和裁剪对以音频为载体进行信息隐写的 LSB 的影响。测试结果如图 3.14 所示。

```
(venv) PS D:\NetworkSecurity\code> python LSB_audio_comp_trim.py
读取载体音频LSB_embedded.wav，其中加密信息：
1.huster。2.大鹏一日同风起，扶摇直上九万里。3.青青子衿，悠悠我心。

压缩音频中
音频已压缩并保存为 D:/NetworkSecurity/resources/LSB/LSB_compressed.wav
解码失败，无法显示解码结果。
压缩后提取的信息与原始信息不匹配，信息隐藏失败！

裁剪音频中
成功裁剪载体音频，将前1/3部分保存到D:/NetworkSecurity/resources/LSB/LSB_trimmed.wav
从裁剪后的音频中提取出的信息：1.huster。2.大鹏一日同风起，扶摇直上九万里。3.青青子衿，悠悠我心。
裁剪后提取的信息与原始信息匹配，信息隐藏成功！
```

图 3.14 LSB_audio_comp_trim.py 程序测试图

由上图可见对音频进行压缩后隐藏信息被破坏，这通常是由于量化效应、压缩过程中的数据损失以及编码方式的改变造成的。

裁剪后隐藏数据并未丢失，这是因为保留了原音频的前 1/3，信息嵌入过程是按照音频帧顺序逐帧嵌入，因此隐藏信息是从音频开头顺序存放，并且未达到总长的 1/3，故而能完整保存下来，一旦裁剪是发生在隐藏信息帧段中间，将信息截断，或者舍弃的音频包含隐写信息，那么就会发生数据丢失，同理裁剪图片时如果隐藏信息所在的像素位置都被保留了下来，那么信息也不会丢失。故 LSB 算法对裁剪具有一定的抵抗性。

再运行 DCT_robustness.py 程序研究 DCT 算法对图像的遮盖、压缩和旋转的抵抗性。测试结果如图 3.15 所示。

```
(venv) PS D:\NetworkSecurity\code> python DCT_robustness.py
以下是DCT算法抗攻击性检测
读取载体图像D:/NetworkSecurity/resources/DCT/DCT.bmp
请输入要嵌入音频的秘密信息(支持中文、英文，数字)例如：
1人生若只如初见，何事秋风悲画扇。2众里寻他千百度，蓦然回首，那人却在，灯火阑珊处。3Fight
1人生若只如初见，何事秋风悲画扇。2众里寻他千百度，蓦然回首，那人却在，灯火阑珊处。3Fight
图像隐写已完成,隐写后的图像保存为D:/NetworkSecurity/resources/DCT/DCT_embedded.bmp
把隐写后的图像压缩，压缩质量80%，保存为D:/NetworkSecurity/resources/DCT/DCT_embedded_compressed.jpg

从未经处理的载体图像DCT_embedded.bmp中提取隐藏信息
开始提取，隐写图像路径：D:/NetworkSecurity/resources/DCT/DCT_embedded.bmp
提取的文本信息：1人生若只如初见，何事秋风悲画扇。2众里寻他千百度，蓦然回首，那人却在，灯火阑珊处。3Fight
提取信息与原始信息的相似度：1.0000
提取的信息与原始信息匹配，信息隐藏成功！
```

```
DCT_embedded.bmp遮挡成功，生成图片DCT_cover.bmp
开始提取，隐写图像路径：D:/NetworkSecurity/resources/DCT/DCT_cover.bmp
隐藏信息被更改，解码失败，无法显示解码结果。
提取信息与原始信息的相似度：0.9062
提取的信息与原始信息不匹配，信息隐藏失败！

从压缩后的图像DCT_embedded_compressed.jpg中提取隐藏信息
开始提取，隐写图像路径：D:/NetworkSecurity/resources/DCT/DCT_embedded_compressed.jpg
隐藏信息被更改，解码失败，无法显示解码结果。
提取信息与原始信息的相似度：0.6299
提取的信息与原始信息不匹配，信息隐藏失败！

正在旋转图像 1°
从旋转后的图像D:/NetworkSecurity/resources/DCT/DCT_embedded_rotated.bmp中提取隐藏信息
开始提取，隐写图像路径：D:/NetworkSecurity/resources/DCT/DCT_embedded_rotated.bmp
隐藏信息被更改，解码失败，无法显示解码结果。
提取信息与原始信息的相似度：0.4922
提取的信息与原始信息不匹配，信息隐藏失败！
```

图 3.15 DCT_robustness.py 程序测试图

由上图知遮挡后隐藏信息被篡改了，提取出来的信息和原信息相似度接近 90%，基本类似，而查看遮盖后的图像 DCT_cover.bmp 后发现遮盖面积占总面积的 1/9（2002/6002），如果嵌入信息随机分布于 8*8 的块中，那么接近 1/9 信息会被修改，8/9 会保留下来，即 88.89%，和 0.9062 很接近，因此可知信息嵌入较均匀。

压缩后的图像中提取出来的信息同原信息相似度只有 0.6299，丢失较多。这是因为在图片压缩过程中，尤其是使用 JPEG 等有损压缩方法时，会对 DCT 变换后的系数进行量化处理。量化步骤会减少数据的精度，将一些较小的、对视觉影响不大的系数设置为零或接近零的值，从而减小文件大小。压缩过程还包括对量化后的系数进行编码和重构。在解码端，通过反量化和逆 DCT 变换来恢复图像。然而，由于量化和编码过程中引入的误差，重构后的图像与原始图像在像素级别上可能存在差异，这些差异会反映到隐藏信息上，造成信息的改变。

旋转图片 1 度就导致信息丢失率高达半数之多，原因一是旋转图片会导致像素位置的改变。在此算法中，隐藏信息是基于特定位置的 DCT 系数大小进行嵌入的。旋转操作后，这些特定位置可能不再对应原有的 DCT 系数，从而导致隐藏信息的提取出现错误。二是插值效应，即旋转过程中，为了保持图像的连续性，通常会对像素进行插值处理，这会引入新的像素值，改变原始像素之间的相关性，进而影响到 DCT 变换后的系数和隐藏信息的完整性。

从以上可见我实现的 DCT 算法中所使用的策略对图像的遮挡、压缩和旋转操作鲁棒性较差。但一般情况下 DCT 算法对压缩和其他图像处理操作的鲁棒性是较强的。

3.3.2 嵌入容量 (Embedding Capacity)

LSB 算法在图像中嵌入数据时，嵌入比例为 1/8~3/8，嵌入数据总量由图像的分

辨率、比例以及使用的颜色通道位数决定。比例越高，数据量越多，图像质量越差；LSB 算法在音频中嵌入数据比例为 1/16~3/16，总量由音频的时长、采样率、嵌入比例决定。比例越高，数据量越多，音频质量越差。

DCT 算法在图像中嵌入数据时，嵌入比例为 1/16~3/16，嵌入数据总量也是由图像的分辨率、嵌入比例以及使用的颜色通道位数决定。相比于 LSB，DCT 算法通常嵌入的信息量较少，因为它在图像的频域中操作，通常仅修改一些较小的频率分量。但可以根据 DCT 系数的重要性动态调整嵌入的数量和质量。

3.3.3 图像质量 (Image Quality)

图像质量使用峰值信噪比（PSNR）和结构相似性（SSIM）来衡量。相关结果可见报告前述图片，由于本报告中 LSB、DCT 两种算法并未对同种图片进行处理，嵌入数据大小也不相同，比较相关数值无说服力。在此给出一般性结论：

由于 LSB 直接修改图像的最低有效位，因此在嵌入信息后图像可能会产生较大的失真，导致 PSNR 较低。虽然对于一些人眼难以察觉的变化，SSIM 值可能还保持较高，但它仍然会影响图像质量。

DCT：由于 DCT 在频域中进行操作，它通常能够保持较高的图像质量，尤其是在较低的嵌入容量情况下。通过选择重要的 DCT 系数来嵌入信息，可以减少对图像质量的影响，因此通常能获得较高的 PSNR 和 SSIM 值。

3.3.4 隐蔽性 (Invisibility)

由于 LSB 隐写直接影响图像的最低有效位，较容易被检测出来，尤其是在高压缩或高噪声环境下，其隐蔽性较差，当嵌入大量信息时，容易产生明显的伪影或噪点。DCT 隐写在频域中操作，通常对图像的视觉效果影响较小，尤其是当仅在较高频率的 DCT 系数中嵌入信息时。其隐蔽性较好，特别是在低容量嵌入时，嵌入的信息难以被察觉。

3.3.5 计算复杂度 (Computational Complexity)

LSB 算法的计算复杂度相对较低，主要是位操作和简单的像素修改，因此执行速度较快。DCT 算法的计算复杂度相对较高，需要对图像进行频域转换，通常需要更多的计算时间和内存资源。

3.3.6 可扩展性 (Scalability)

LSB 算法通常具有良好的可扩展性，因为它直接操作图像的像素值，对图像的尺寸或类型变化影响较小。DCT 隐写的可扩展性稍差一些，因为它依赖于频域变换，且图像的频域特性随着尺寸或内容的变化可能有所不同。

3.3.7 安全性 (Security)

LSB 在安全性方面较弱，因为其嵌入的数据容易被攻击者发现，且提取过程简单。DCT 隐写的安全性较高，因为它对频域系数进行了修改，这些系数难以通过简单的视觉或统计分析检测出来，尤其是当信息嵌入在较高频率区域时。

LSB 和 DCT 算法性能评估对比表如下表 3.1 所示。

表 3.1 LSB 和 DCT 算法性能评估对比表

性能评估指标	LSB	DCT
鲁棒性	较低，容易被压缩、噪声等破坏	高，对压缩和噪声更鲁棒
嵌入容量	高，能够嵌入大量信息	较低，适合嵌入较少信息
图像质量	较低，容易引入视觉失真	高，图像质量保留较好
隐蔽性	较差，容易被检测到	较好，视觉上几乎不易察觉
计算复杂度	低，快速	较高，需要 DCT 变换
可扩展性	良好	较差，受图像尺寸和类型影响
安全性	较弱	较强，难以检测和提取

4 信息隐藏算法总结与心得

4.1 信息隐藏算法总结

在本次课程实践中，我主要完成了以下工作，深入地探索了信息隐藏相关的理论和算法应用：

1、根据理论描述和具体问题设计了能在彩色图片中隐写黑白水印图片的最低有效位（LSB）算法，能在 wav 格式音频中隐写字符串信息的 LSB 算法，以及在灰度图像中隐写字符串数据的离散余弦变换（DCT）算法。

2、实现了对输入字符串的多种转换，例如生成黑白水印、二进制流等，并对嵌入后的载体进行比较评估和信息提取，对提取出的信息同原信息进行比较，评估算法的质量。

3、完成了对已实现算法的鲁棒性的功能测试和嵌入容量的分析，对生成的载体进行多种操作，然后提取出信息，计算分析相关数据。

4.2 信息隐藏算法心得

在这次课程实践中，我遇到了不少问题，查阅相关文献并多次调试后得以解决，部分问题及解决如下：

1、彩图中隐写黑白水印图片的算法实现过程中，我将字符串转化成黑白水印图片，随之出现的问题是黑白水印应该多大，其中字符串字体多大，所处位置在哪合适等等，黑白水印过大时彩图 8 比特存储 1 位存不下，黑白水印较小时又需要传递其大小到提取函数，否则提取函数不知道提取区间界限。我纠结于黑白水印大小是随输入字符串长度变化还是固定，后者又需要考虑字符串是否存得下。最终我决定将水印大小设定为彩图大小，方便嵌入和提取，输入字符串字体大小根据其长度动态调整，位置经过计算使其居中。

2、在音频中隐写字符串信息的过程中，我初始认为输入字符串会有结束符'\0'，转换成二进制即为 00000000，我便以此为结束标志，嵌入后提取出所有音频帧的最低位组成二进制串，查找结束标志，在这之前的即为隐写的信息。但测试后发现字符串转换成二进制后并没有结束符 00000000，我便在字符串转换成二进制后的末尾

加入 00000000，即人为的加入结束标志。但我经过测试发现我输入的字符串有时可以正常提取转换，有时却报错，特别是字符串越长，越有可能报错。经过多次尝试发现，问题出在结束符上，设置为 00000000，那么正常以 10 结尾的隐藏信息加入 00000000 后，读取时就把正常的 0 当做了结束符一部分，导致解码失败。故而我把结束符特殊化，例如 10110111011110111110111101110111011010，避免隐藏信息被误认为结束标志情况的发生。

3、在 DCT 算法实现过程中，我使用 randrc 函数生成随机的 8*8 像素块行列位置，用于记录嵌入信息的 DCT 系数位置，我这个函数输出的是 8*8 像素块行列索引，即是原图像行列大小的 1/8，最初我并未注意到这个问题，在修改 Y 矩阵时使用的是矩阵块的行列索引，能运行成功，但生成的图像有很明显的黑影，可在文件中查看 DCT_wrong.bmp，这是因为我所做修改都集中在一小块区域，行列值为原图形的 1/8，而且容易发生冲突，多次修改了同一块的 2 个点，导致解码时出错。发现问题后我将行列索引乘 8，即从矩阵块的索引恢复到了原图像矩阵索引中来，这些问题就都得到了解决，解码时不会出错，生成的图片也看不出区别。

本次报告中我也有部分内容没有实现，例如使用 LSB 算法在灰度图像中隐写字符串，好与 DCT 算法形成对比，在同样的原始图片、同样的隐藏信息情况下进行算法性能的评估，测试鲁棒性等相关特性。音频隐写中对于相似度的计算还有待改进，可以增加对 LSB 载体图像旋转、加噪声、裁剪的测试，

参考文献

[1]安波,许宪东,王亚东.基于最低有效位的数字水印技术[J].黑龙江工程学院学报,2005,(01):30-33.DOI:10.19352/j.cnki.issn1671-4679.2005.01.011.

[2]尹浩, 林闯, 邱锋, 等. 数字水印技术综述[J]. 计算机研究与发展, 2005, 42(7): 1093.

[3]刘涤.基于离散余弦变换的数字水印算法研究[J].电脑编程技巧与维护,2023,(09):138-140.DOI:10.16184/j.cnki.comprg.2023.09.019.

[4]skyerhxx.Visual-watermarking-system-based-on-digital-image.2020.GitHub.Available at:https://github.com/skyerhxx/Visual-watermarking-system-based-on-digital-image

• 指导教师评定意见 •

一、原创性声明

本人郑重声明本报告内容，是由作者本人独立完成的。有关观点、方法、数据和文献等的引用已在文中指出。除文中已注明引用的内容外，本报告不包含任何其他个人或集体已经公开发表的作品成果，不存在剽窃、抄袭行为。

特此声明！

作者签字: