信息隐藏技术

实验四：媒体文件格式剖析

2111454-李潇逸
2111876-梅骏逸
2110049-张刘明
2112474-王志远
2111252-李佳豪

一、实验目的

1. 熟悉wav媒体文件格式
2. 掌握音频文件wav信息隐藏方法
3. 用matlab实现wav文件的信息隐藏过程

二、实验原理

- wav文件格式
  i. wav文件

  WAV即波形声音文件格式 (Waveform Audio File Format，简称WAVE，因后缀为*.wav故简称WAV文件)，其采用RIFF（Resource Interchange File Format，资源互换文件格式）结构，并符合（**RIFF**）规范，用于保存Windows平台的音频信息资源，被Windows平台及其应用程序所广泛支持。Wave格式支持MSADPCM、CCITT A律、CCITT μ律和其他压缩算法，支持多种音频位数、采样频率和声道，是PC机上最为流行的声音文件格式；但由于“无损”的特点，WAV文件格式所占用的磁盘空间相对较大（每分钟的音乐大约需要12MB磁盘空间），故此文件格式多用于存储简短的声音片段。同时WAV文件格式通常用来保存PCM格式的原始音频数据，所以通常被称为无损音频。但是严格意义上来讲，WAV也可以存储其它压缩格式的音频数据。

  ii. 音频文件参数

  对于一般的音频文件，其蕴含的文件参数包括：
  - 采样率：声音信号在“模→数”转换过程中单位时间内采样的次数。
  - 采样值（采样精度）：每一次采样周期内声音模拟信号的积分值。同时，每个采样数据记录的是振幅, 而采样精度取决于储存空间的大小。

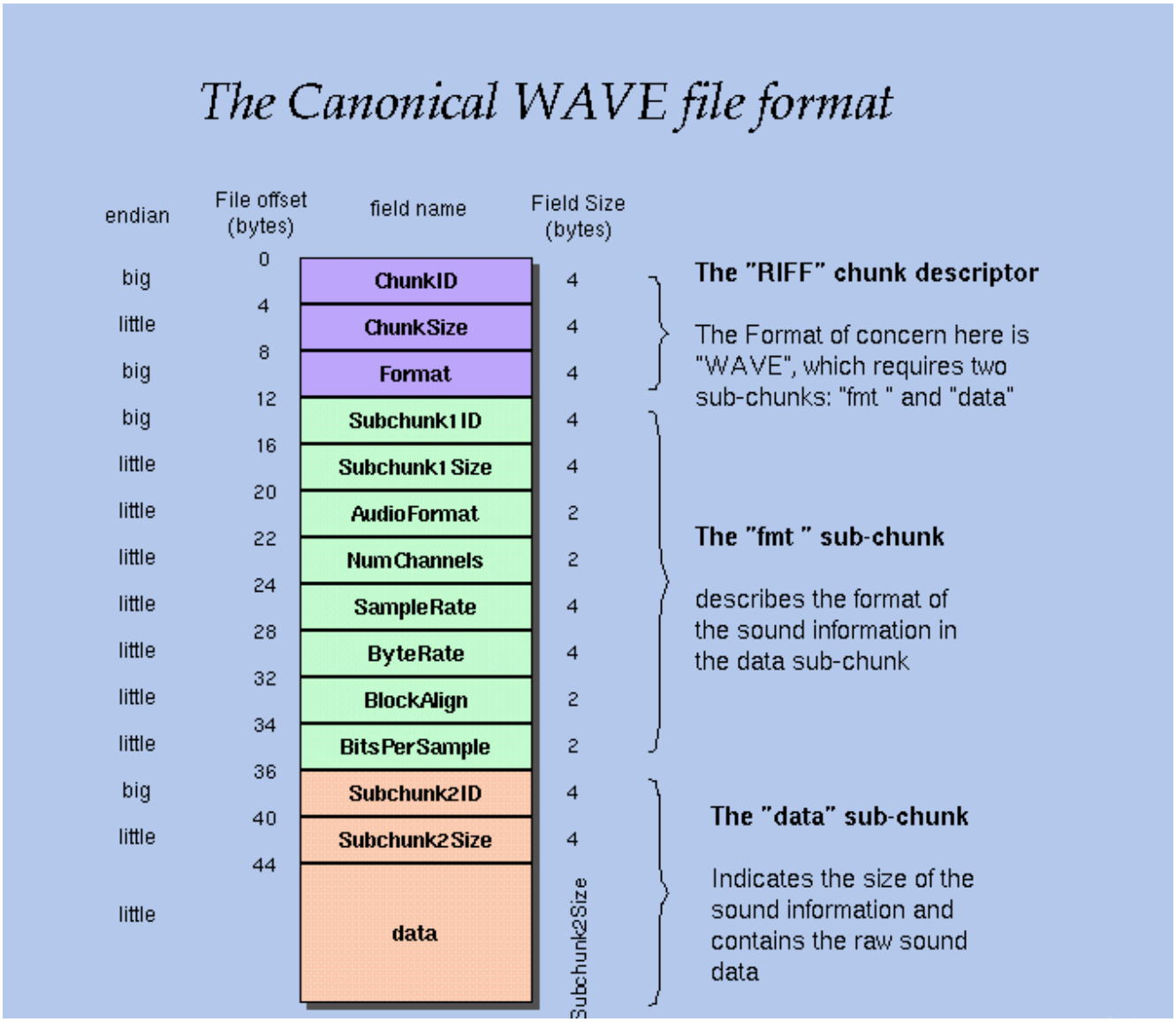
    对于单声道（mono）文件，采样数据为8位的短整数，同时其采样精度有：
  - 1 字节（8bit） 只能记录 256 个数, 也就是只能将振幅划分成 256 个等级；
  - 2 字节（16bit） 可以细到 65536 个数, 即为 CD 标准；

- 4 字节（32bit） 能把振幅细分到 4294967296 个等级

  对于双声道立体声（stereo）文件，每次采样数据为一个16位的整数（int），且采样是双份的，也为单声道文件的两倍。采样数据中高八位(左声道)和低八位(右声道)分别代表两个声道。

iii. wav文件结构

WAV文件遵循RIFF规则，其数据体以区块（Chunk）为最小单位进行存储，而整个文件则以文件头进行标识。其结构示意如下图：



a. wav文件头

WAV（Waveform Audio File Format）文件的头部包含了文件的基本信息，其中包括文件格式、音频数据的编码方式、采样率、声道数等。WAV文件头的结构如下：

- 文件标识符（4字节）：WAV文件的头部通常以字符串"RIFF"开始，表示文件的格式。
- 文件大小（4字节）：以字节为单位表示整个WAV文件的大小，包括头部和数据部分。
- 文件格式（4字节）：以字符串"WAVE"表示该文件是一个WAV文件。

- 格式块标识符（4字节）：以字符串"fmt "表示格式块的开始。
- 格式块大小（4字节）：表示格式块的大小，通常为16字节。
- 音频格式（2字节）：表示音频数据的编码格式，一般为PCM（脉冲编码调制）编码。
- 声道数（2字节）：表示音频数据的声道数，常见的值有单声道（Mono）和立体声（Stereo）。
- 采样率（4字节）：表示每秒采样的样本数，以赫兹（Hz）为单位。
- 数据传输速率（4字节）：表示每秒传输的数据量，以字节为单位，通常等于采样率乘以采样大小除以8。
- 数据块对齐（2字节）：表示每个采样的字节数，通常为声道数乘以采样大小除以8。
- 位深度（2字节）：表示每个样本的位数，即每个采样的量化位数，通常为8位、16位、24位或32位。
- 数据块标识符（4字节）：以字符串"data"表示数据块的开始。
- 数据块大小（4字节）：表示数据块的大小，即音频数据的长度，以字节为单位。

b. wav数据体

WAV文件的数据体区块一般由3个区块组成：RIFF Chunk、Format Chunk和Data Chunk。另外，文件中还可能包含一些可选的区块，如：Fact Chunk、Cue Points Chunk、Playlist Chunk、Associated Data List Chunk等

- 信息隐藏方法

目前主要的音频信息隐藏技术分为时域和变换域音频信息隐藏方法两类。

经典时域音频信息隐藏技术：到目前为止，公认比较成熟的时域音频信息隐藏技术有四种：最不重要位法、回声隐藏法、相位编码法、扩频法。

i. LSB方法：

最不重要位（LSB）的隐藏算法是最早应用于音频信息隐藏领域的算法。它的基本思想是用秘密数据替换原始音频信号采样值的最低几个比特位，达到隐藏的目的。LSB 算法具有计算复杂度低、可实时实现及通用性等优点，但其抵抗攻击的能力较弱。

ii. 回声隐藏法：

回声隐藏法是通过引入回声来将秘密信息嵌入到载体中。与其他方法不同，回声隐藏法对载体音频信号的改变，考虑的是环境条件而不是随机噪声的特性，因而具有较强的抵抗主动攻击的能力。但信道噪声、任何形式的篡改都会直接影响算法的正确提取率。

iii. 相位编码法：

相位编码是利用 HAS 对人耳对绝对相位的不敏感性及对相对相位的敏感性，将代表秘密信息的参考相位替换原始音频段的绝对相位，并对其他音频段进行相应调整，以保持各段之间的相对相位不变。

iv. 扩频法：

扩频法的基本思想是利用扩频调制技术将秘密信息扩展到整个可听频谱范围内，再将扩频后 的秘密信息叠加到原始的音频信号中完成隐藏。

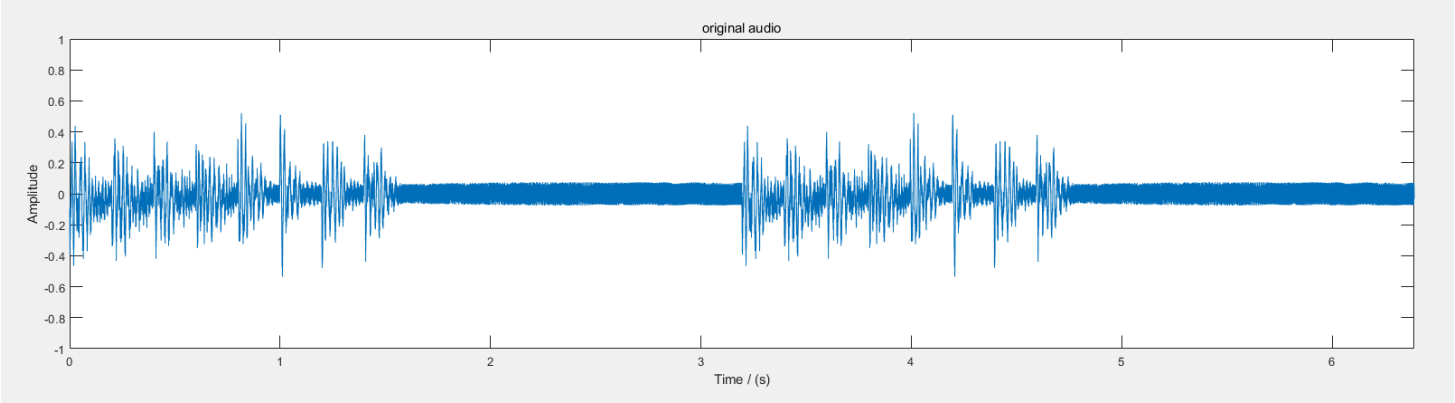
本次实验中，将采用LSB方法进行实验。

三、实验过程

1. 载体音频和秘密信息

原载体音频波形图：

input.wav



大小:	550 KB (563,756 字节)
占用空间:	552 KB (565,248 字节)

秘密信息图像：

watermark.png

大小:	1.93 KB (1,979 字节)
占用空间:	4.00 KB (4,096 字节)

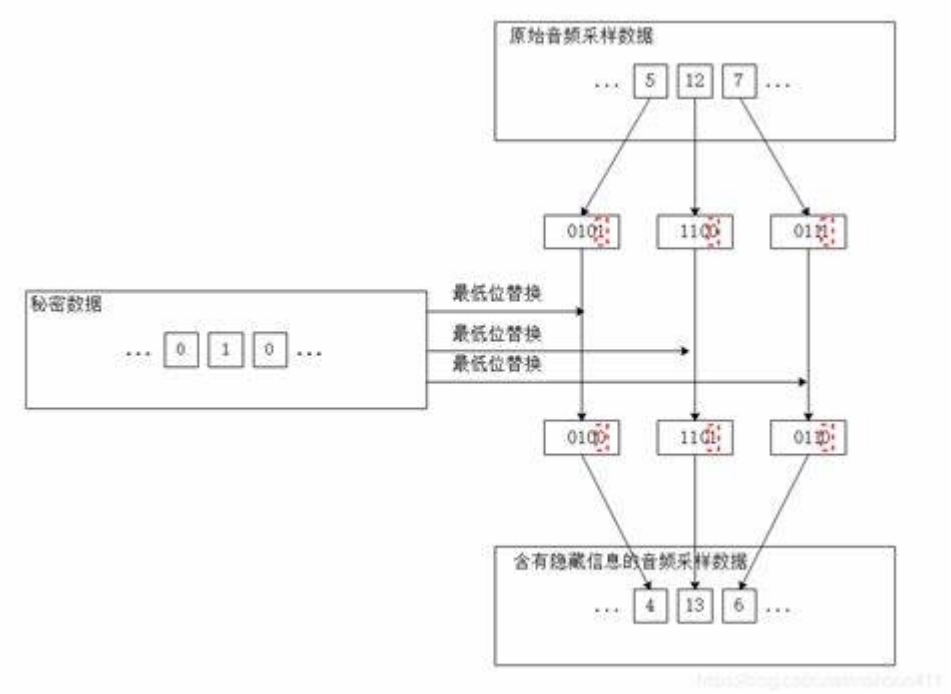
利用代码获取秘密图像的长和宽：

```
% 获取图像的长和宽
[height, width, ~] = size(watermark);

disp(['图像的长为：', num2str(height)]);
disp(['图像的宽为：', num2str(width)]);
```

```
图像的长为：506
图像的宽为：888
```

2. 信息隐藏



主函数：hide.m

```
% 读取音频文件
audio = readAudio('input.wav');

% 读取水印图像
watermark = readWatermark('watermark.png');

% 获取图像的长和宽
[height, width, ~] = size(watermark);

disp(['图像的长为：', num2str(height)]);
disp(['图像的宽为：', num2str(width)]);

% 将水印嵌入音频文件
marked_audio = embedWatermark(audio, watermark);

% 将带有水印的音频文件写入新文件
fid = fopen('output.wav', 'wb');
fwrite(fid, marked_audio, 'uint8');
fclose(fid);

% 绘制波形图
[y1, Fs] = audioread('input.wav');
[y2, Fs] = audioread('output.wav');
plotWaveform(y1, y2, Fs);
```

实现音频水印隐藏的过程，并对隐藏水印后的音频文件进行了保存和可视化。具体流程如下：

1. 通过 readAudio 函数读取输入的音频文件 'input.wav'，并将其存储在变量 audio 中。
2. 通过 readWatermark 函数读取水印图像文件 'watermark.png'，并将其存储在变量 watermark 中。
3. 使用 size 函数获取水印图像的长和宽，并将其打印输出。
4. 调用 embedWatermark 函数将水印嵌入到音频文件中。该函数将音频文件中的部分字节用来存储水印信息。
5. 使用 fopen 函数创建一个新的文件 'output.wav'，并使用 fwrite 函数将带有水印的音频数据写入该文件中。最后使用 fclose 函数关闭文件。
6. 使用 audioread 函数读取原始音频文件 'input.wav' 和带有水印的音频文件 'output.wav'，并将它们的波形数据分别存储在变量 y1 和 y2 中，采样率存储在变量 Fs 中。
7. 使用 plotWaveform 函数绘制原始音频文件和带有水印的音频文件的波形图，以便比较它们的波形特征。

当要隐藏的秘密信息过大时：

```
>> hide
图像的长为：1640
图像的宽为：2360
错误使用 embedWatermark (第 6 行)
音频载体太小，请更换载体

出错 hide (第 14 行)
marked_audio = embedWatermark(audio, watermark);
```

各功能函数如下：

readAudio.m

```
function audio = readAudio(filename)
    fid = fopen(filename, 'rb');
    audio = fread(fid, inf, 'uint8');
    fclose(fid);
end
```

readWatermark.m

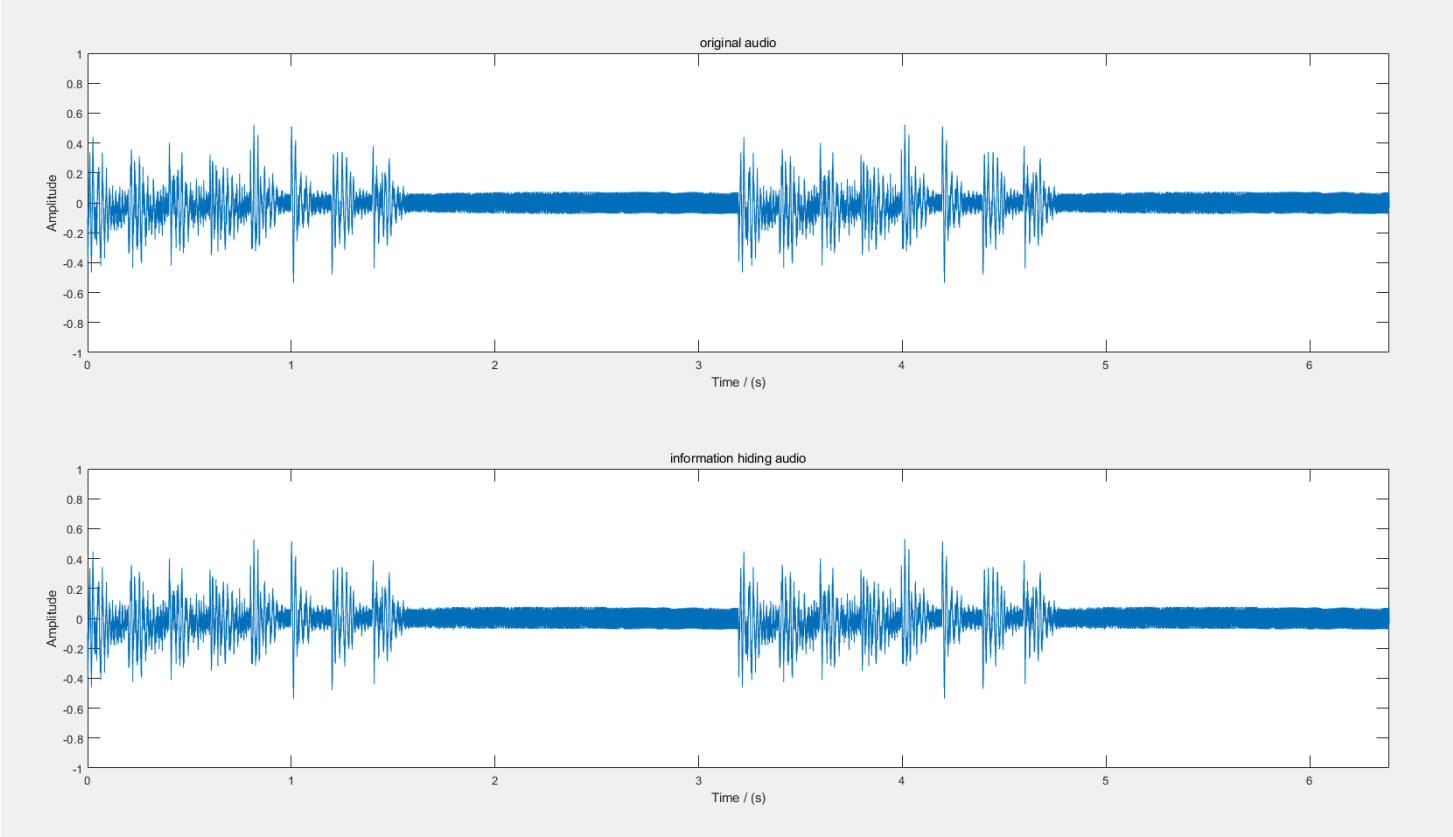
```
function watermark = readWatermark(filename)
    watermark = imread(filename);
end
```

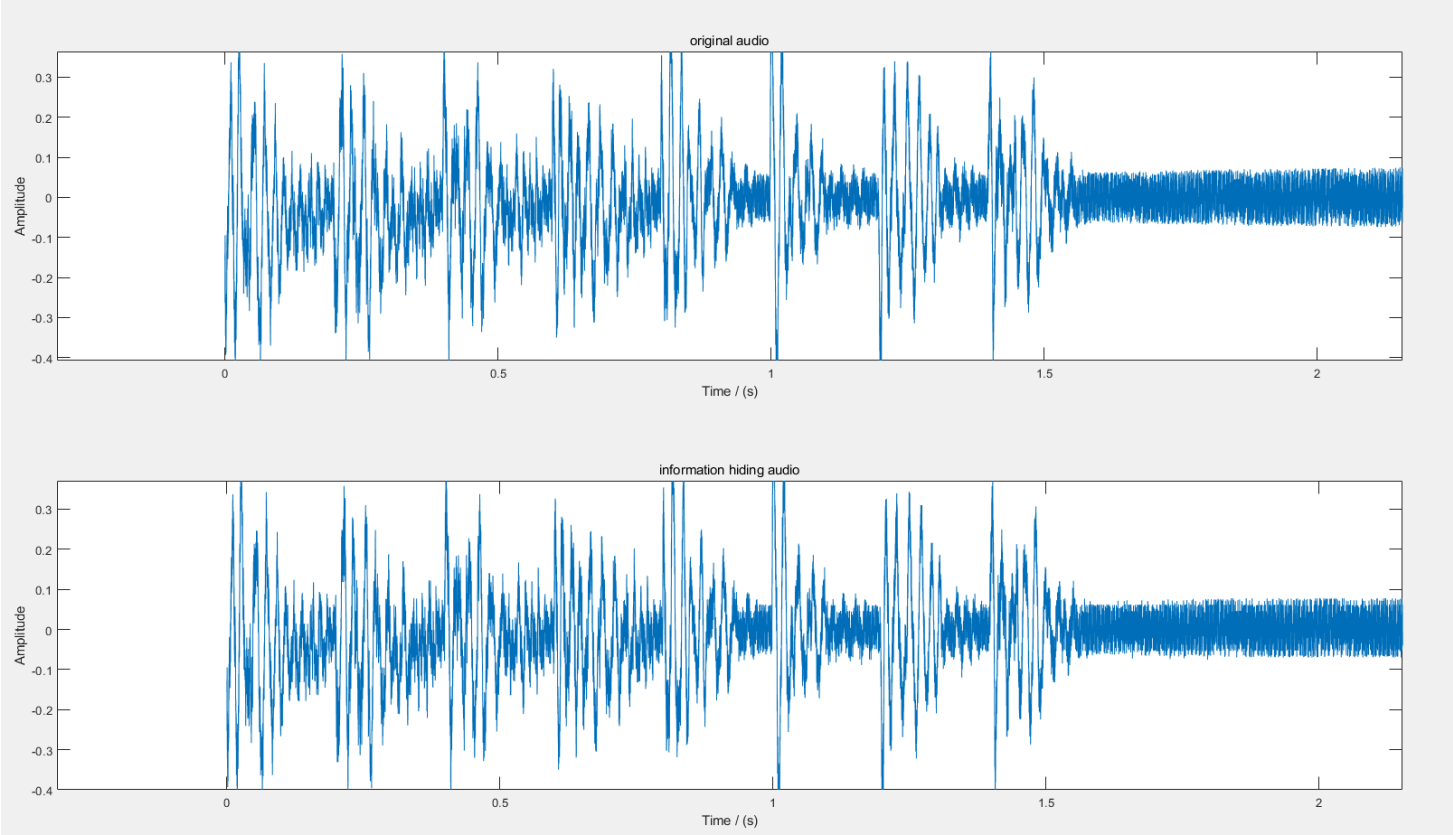
embedWatermark.m

```
function marked_audio = embedWatermark(audio, watermark)
    n = length(audio) - 44;
    [row, col] = size(watermark);
    wi = watermark(:);
    if row * col > n
        error('音频载体太小，请更换载体');
    end
    marked_audio = audio;
    for k = 1 : row * col
        marked_audio(44 + k) = bitset(marked_audio(44 + k), 1, wi(k));
    end
end
```

运行结果：

隐藏秘密信息前后的波形图对比：





隐藏秘密信息前后的内存对比：

大小:	550 KB (563,756 字节)	大小:	550 KB (563,756 字节)
占用空间:	552 KB (565,248 字节)	占用空间:	552 KB (565,248 字节)

3. 信息提取

主函数：extract.m

```
% 读取音频文件
audio = readAudio('output.wav');

% 提取水印信息
watermark = extractWatermark(audio, 449328);

% 将提取的水印信息转换为图像
extracted_image = watermarkToImage(watermark);

% 读取原始图像
orig_image = imread('watermark.png');

% 显示原始图像和提取的水印图像
displayImages(orig_image, extracted_image);
```

实现音频水印提取的过程，并将提取的水印信息转换为图像进行显示。具体流程如下：

1. 通过 readAudio 函数读取带有水印的音频文件 'output.wav'，并将其存储在变量 audio 中。
2. 调用 extractWatermark 函数从音频文件中提取水印信息。在该函数中，根据已知的水印信息长度（此处为 449328）来提取水印数据。
3. 使用 watermarkToImage 函数将提取的水印信息转换为图像。该函数将水印信息按照图像的长和宽重新排列，并返回提取的水印图像。
4. 通过 imread 函数读取原始水印图像文件 'watermark.png'，并将其存储在变量 orig_image 中。
5. 使用 displayImages 函数显示原始水印图像和提取的水印图像。这里可以对比两幅图像，验证提取水印的准确性。

功能函数如下：
extractWatermark.m

```
function watermark = extractWatermark(audio, n)
    watermark_bits = zeros(n, 1);
    for i = 1 : n
        watermark_bits(i) = bitget(audio(44 + i), 1);
    end
    watermark = reshape(watermark_bits, 506, 888);
end
```

watermarkToImage.m

```
function image = watermarkToImage(watermark)
    image = uint8(watermark * 255);

end
```

displayImages.m

```
function displayImages(orig_image, extracted_image)
    figure;
    subplot(1, 2, 1); imshow(orig_image); title('Original Picture');
    subplot(1, 2, 2); imshow(extracted_image); title('Extracted Picture');
end
```

实验结果：

提取秘密信息图像与原始图像对比：