## 实验四：媒体文件格式剖析

1. **实验要求**

依次完成以下三种有关媒体文件格式剖析的实验：

1、任选一种媒体文件，进行格式剖析（建议用UltraEdit）；

2、针对该类型的文件，讨论可能的隐藏位置和隐藏方法；

3、实现秘密信息的隐藏和提取（选做）。

本次实验，我选择对MP3（MPEG Audio Layer 3）进行介绍，所使用的mp3文件来自https://www.aigei.com/，在网站中的名称为“船舶, 汽车渡轮, 17,400 吨, Sir”。

1. **实验原理**
2. **MP3文件格式**

MP3全称为MPEG Audio Layer 3，它是一种高效的计算机音频编码方案，它以较大的压缩比将音频文件转换成较小的扩展名为.mp3的文件，基本保持源文件的音质，MP3是ISO/MPEG标准的一部分。

1. 编码和解码

MP3音频压缩包含编码和解码两部分，编码是将原始信号转换成电平信号的过程，解码即是逆过程，MP3 采用了感知音频编码（PerceptualAudio Coding）这一失真算法。人耳感受声音的频率范围是20Hz-20kHz，MP3截掉了大量的冗余信号和无关的信号，编码器通过混合滤波器组将原始声音变换到频率域，利用心理声学模型，估算刚好能被察觉到的噪声水平，再经过量化，转换成Huffman编码，形成MP3位流。解码器要简单得多，它的任务是从编码后的谱线成分中，经过反量化和逆变换，提取出声音信号。

1. MP3文件结构

MP3文件数据由多个帧组成，帧是MP3文件最小组成单位。每个帧又由帧头、附加信息和声音数据组成。每个帧播放时间是0.026秒，其长度随位率的不同而不等。有些MP3文件末尾有些额外字节存放非声音数据的说明信息。

MP3文件大体上分为三个部分：ID3V2，音频数据，ID3V1。

|  |  |
| --- | --- |
| 名称 | 说明 |
| ID3V2 | 位于 mp3 文件的首部，包含作者、作曲、专辑、等信息、长度不固定，扩展了 ID3V1 的信息量 |
| 音频数据 | 1. 音频数据由一系列的数据帧构成 2. 每个帧的长度可能固定，也可能不固定，由位率决定 3. 每个帧都包括帧头、数据实体两部分 4. 帧头记录着 mp3 的位率、采样率、版本等信息 5. 如果启用 CRC 校验,则帧头后跟随2 字节 CRC 校验之后可能存在 32 字节的附加信息，也可能无附加信息 |
| ID3V1 | Mp3 文件末尾的固定 128 字节，包含作者作曲专辑等 |
| 说明信息 | 有些mp3文件可能会有一些额外的说明信息（不主要） |

1. ID3V2解析

ID3V2通常记录在文件的首部，由一个标签头和若干个标签帧或者一个扩展标签头组成，至少要有一个标签帧，每一个标签帧记录一种信息，例如作曲、标题等。

* 1. 标签头

位于文件开始出，长度为10字节，结构如下：

* char Header[3]; 必须为“ID3”，否则认为标签不存在
* char Ver; 版本号ID3V2.4就记录4
* char Revision; 副版本号此版本记录为0
* char Flag; 标志字节，只使用高三位，其它位为0
* char Size[4]; 标签大小

如下图所示，深蓝色的即为标签头：

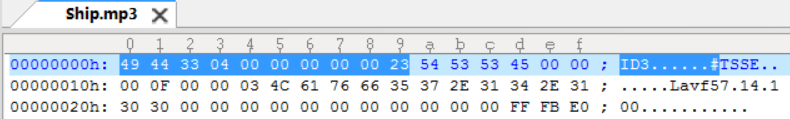


图 1 标签头

对应内容为：

|  |  |  |  |
| --- | --- | --- | --- |
| 地址 | 字节 | 值 | 说明 |
| 0H-2H | 3 | 49 44 33 | ID3 ASCII码 |
| 3H | 1 | 04 | ID3版本号，ID3V2.4 |
| 4H | 1 | 00 | 副版本号，此版本为0 |
| 5H | 1 | 00 | 标志字节，均为0 |
| 6H-9H | 4 | 00 00 00 23 | 标签大小 |

* 1. 标签帧

每个标签帧都有10个字节的帧头(和标签头不是一个东西，虽然他们刚好都是10字节，标签头只有一个，每个标签帧都有一个帧头)和至少一个字节的内容构成，标签帧与标签头/其他标签帧无特殊字节分割，只能通过帧头信息来确定帧内容的大小。

* char ID[4]; 标识帧，说明其内容，例如作者/标题等
* char Size[4]; 帧内容的大小，不包括帧头，不得小于1
* char Flag[2]; 标志帧，只定义了6 位

如下图为TSSE标签帧：

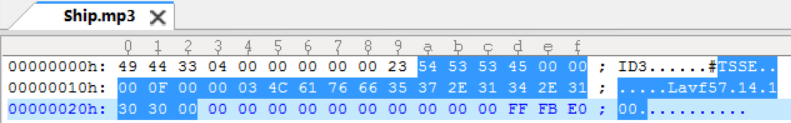


图 2 TSSE标签帧

对应内容为：

|  |  |  |  |
| --- | --- | --- | --- |
| 地址 | 字节 | 值 | 说明 |
| AH-DH | 4 | 54 53 53 45 | TSSE帧标识 |
| EH-11H | 4 | 00 00 00 0F | 帧内容大小：15字节 |
| 12H-13H | 2 | 00 00 | 标识，均为0 |
| 14H-22H | 13 | 00 | 帧内容 |

1. 音频数据解析

每个帧都有一个帧头，长度是四个字节，帧后面可能有2字节的CRC校验，取决于帧头的第16位，为0则无校验，为1则有校验，后面是可变长度的附加信息，对于标准的MP3文件来说，其长度是32字节，紧接其后的是压缩的声音数据，当解码器读到此处时就进行解码了。

|  |  |  |
| --- | --- | --- |
| 名称 | 长度（bytes） | 属性 |
| 帧头 | 4 | 必存在 |
| CRC | 2 | 可能不存在 |
| 通道信息 | 32 | 必存在 |
| 声音数据 | N | 必存在 |

如图所示，为第一帧音频数据：

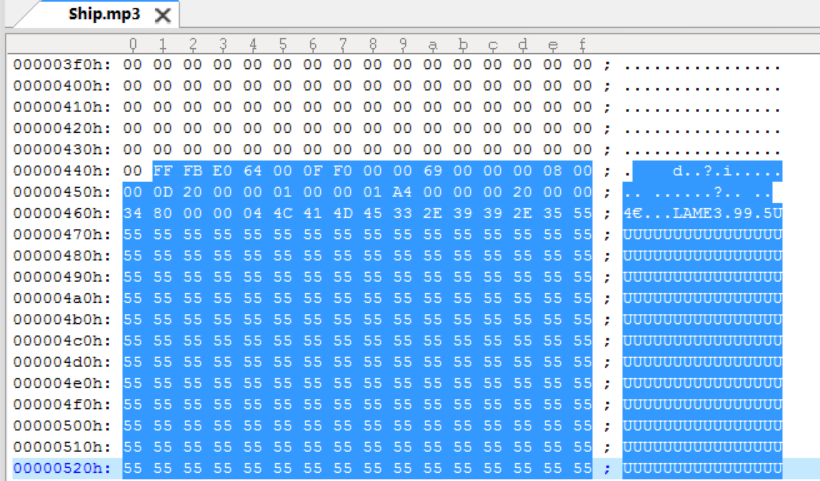


图 3 音频数据

解释如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 地址 | 字节 | 值 | 说明 |
| 441H-444H | 4 | FF FB E0 64 | 帧头 |
| 447H-466H | 32 | F0 00 ... | 通道信息 |
| 467H-857H | 1008 | 4D ... | 由帧头得知：采样数为1152；位率为320kbps；采样频率为44.1K。故帧长度为 （1152/8\*320）/44.1 = 1044 |

1. ID3V1解析

ID3 V1.0标准并不周全，存放的信息少，无法存放歌词，无法录入专辑封面、图片等。V2.0是一个相当完备的标准，但给编写软件带来困难，虽然赞成此格式的人很多，在软件中真正实现的却极少。绝大多数MP3仍使用ID3 V1.0标准。此标准是将MP3文件尾的最后128个字节用来存放ID3信息。

1. **流载体的LSB信息隐藏方法**

LSB（Least Significant Bit，最不重要位/最低有效位）方法是一种隐写术方法，可以被用于在音频等流载体中隐藏秘密信息。该方法通过在流载体的最低有效位（即最不重要位）中嵌入秘密信息，以实现信息隐藏的目的。在LSB方法中，将秘密信息的比特逐一替换到流载体的最低有效位中，这样可以在听觉上不影响原始音频的质量，同时又能够隐藏秘密信息。在提取秘密信息时，只需将音频的最低有效位提取出来即可恢复隐藏的信息。

LSB方法的优点是简单，易实现，容量大；缺点是安全性不高，不能抵抗叠加噪声，有损压缩等破坏。我们可以使用一些方法来提高安全性：对秘密信息先加密后隐藏，多次重复嵌入或引入纠错编码技术（先进行纠错编码，再进行隐藏）。

要隐藏的数据为“我是爱南开的; I Love NKU !”，它的Unicode编码共32\*8=256 Bit。

1. 嵌入

选择一个载体元素的子集，其中共有个元素，用以隐藏秘密信息的个比特。然后在这个子集上执行替换操作，把的最低比特用来替换。

根据Ship.mp3文件讲解如何进行基于LSB方法的隐藏信息嵌入：前文提到，第一个数据帧声音数据起始Byte位置为467H（467H的十进制1127，但由于UltraEdit的下标从0开始，而MATLAB的下标从1开始，故467H对应MATLAB中的下标为1128），声音数据终止Byte位置为857H（857H的十进制为2135，对应MATLAB中的下标为2136），因此第1128-2136 Byte可以供我们隐藏数据，共1008 Byte，由于使用LSB信息隐藏方法，故每个Byte隐藏一个Bit数据，共可以隐藏1008 Bit的数据，而我们要隐藏的信息为256 Bit，空间很充足。

例如，“我”字的Unicode编码为230，二进制形式为11100110，需要8 Byte的声音数据来隐藏。在本例中，隐藏前的声音数据的二进制形式为（1128-1135）：

01001101 0100 0101 00110011 00101110

00111001 0011 1001 00101110 00110101

将“我”的Unicode编码使用LSB方法隐藏，隐藏之后的声音数据的二进制形式为：

0100110**1** 0100 010**1** 0011001**1** 0010111**0**

0011100**0** 0011 100**1** 0010111**1** 0011010**0**

1. 提取

找到嵌入信息的伪装元素的子集，把这些伪装对象抽出它们的最低比特位，排列之后组成秘密信息。

该过程为上一步骤的逆过程，在此不再赘述。

1. **信息隐藏及信息提取**

首先，以二进制形式读取音频文件，得到对应数据data：

|  |
| --- |
| %% 读取文件  fid\_read = fopen("audio\Ship.mp3", 'rb'); % 打开要读取的二进制文件  data = fread(fid\_read, 'uint8'); % 以uint8格式读取文件内容  fclose(fid\_read); % 关闭文件  start\_index = 1128; % 第一个数据帧的声音数据起始位置 |

1. **信息隐藏**

将“我是爱南开的; I Love NKU !”的这几个字隐藏进音频数据中，使用unicode2native将Unicode编码的字符串转化为Unicode码，使用 bitget 获取inf\_uni(i)的第j比特，bitset将data(index)的最低比特位设置为相应的数据。

|  |
| --- |
| %% 隐藏信息  information = "我是爱南开的; I Love NKU !";  inf\_uni = unicode2native(information);% 将字符串转为unicode编码; 此处为 1x32 uint8  % native2unicode(inf\_uni)  [dim,inf\_len] = size(inf\_uni);  index = start\_index;  for i=1:inf\_len  for j=1:8  % 使用 bitget 获取inf\_uni(i)的第j比特  hidden = bitget(inf\_uni(i), j) ;  % LSB设置为对应的数据  data(index) = bitset(data(index),1,hidden);  index = index + 1;  end  end |

1. **信息提取**

从隐藏信息的data中恢复出原始的信息，使用bitget提取data(index)的最低比特位，使用bitset设置inf\_uni\_recover(i)的第j位为对应的比特，使用native2unicode将Unicode码转为其对应的Unicode字符。

|  |
| --- |
| %% 信息提取  inf\_uni\_recover = zeros(1,32, "uint8");  index = start\_index;  for i=1:inf\_len  for j=1:8  hidden = bitget(data(index),1);% 提取LSB数据  index = index + 1;  inf\_uni\_recover(i) = bitset(inf\_uni\_recover(i),j,hidden);%赋值  end  end  inf\_recover = native2unicode(inf\_uni\_recover);% 提取到的信息 |

最终结果如下所示，inf\_recover即为提取得到的信息：



图 4 提取结果

此外，原始音频文件的波形图为：

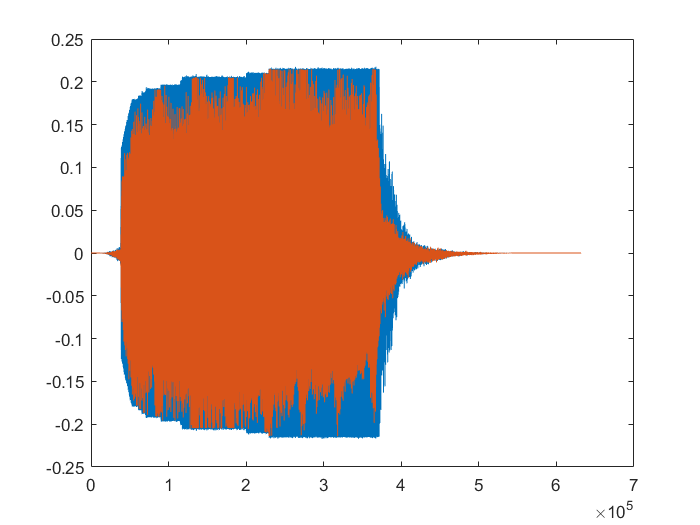


图 5 原始音频

隐藏信息后的音频文件为：

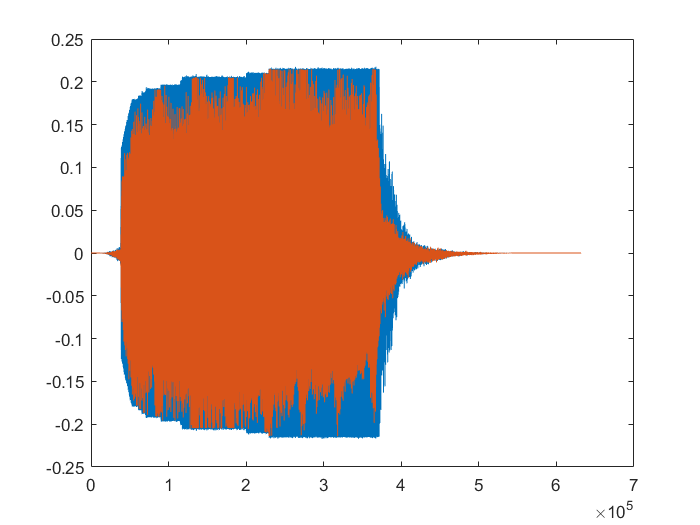


图 6 隐藏信息的音频

可以看出，两者几乎没有区别，可以起到很好的信息隐藏效果。

1. **心得体会**

经过本次实验，我对MP3文件格式有了一定的了解，并找到了MP3中可以用于信息隐藏的数据部分，然后尝试在MP3文件的数据帧的声音数据部分隐藏一个字符串信息“我是爱南开的; I Love NKU !”，并取得成功。结果表明，隐藏信息前后的音频几乎没有区别，可以起到很好的信息隐藏效果。