南開大學
Nankai University

网络空间安全学院

信息隐藏实验报告

第二次大作业：媒体文件格式剖析

姓名：魏伯繁

学号：2011395

专业：信息安全

2023 年 4 月 20 日

目录

媒体文件格式剖析

【摘要】本次实验报告主题为媒体文件格式剖析并分析可能的信息隐藏位置并实践，旨在研究不同媒体文件格式（如图像、音频、视频等）中的信息隐藏可能性及方法，并通过实践验证不同隐藏技术的有效性。实验从媒体文件的结构入手，对其进行剖析，并深入探讨各种信息隐藏技术（如 LSB、DCT、SVD 等）的具体实现原理和应用场景，进一步探索它们在媒体文件中的应用情况以及可能的信息隐藏位置。通过实践部分，实验报告将展示 LSB 信息隐藏技术的实际应用过程，并提供可以直接运行的 py 文件供读者验证

关键字： ogg 文件 信息隐藏 LSB 法 文件格式

1 实验要求

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）。

2 实验原理

2.1 技术原理

流媒体信息隐藏技术是一种在音视频流中嵌入隐藏信息的技术，主要应用于数字版权保护、数字水印等领域。它将隐藏信息嵌入到流媒体中，使得信息能够随着音视频流的传输而传输。流媒体信息隐藏技术对信息本身的安全性要求较高，一方面要确保嵌入的信息不易被攻击者发现和修改，并且对流媒体的质量、容量、传输速率等方面的要求也比较高。

流媒体信息隐藏技术一般分为基于时域的技术、基于频域的技术和基于多媒体格式的技术三种。

基于时域的技术是在原始音视频数据上进行信息隐藏，如通过改变采样率、量化值、帧率等方式，将隐藏信息嵌入到这些参数中。但是这种方法需要对音视频数据进行调整，可能会导致音视频质量下降，不利于视频的播放和传输。

基于频域的技术则是在音视频数据的傅里叶变换域中进行信息嵌入，即将信息嵌入到音视频的特定频率区间中，同时需要对音视频数据进行调整，以便在傅里叶变换域中嵌入信息。这种方法对音视频的影响较小，不会影响音视频的质量，但需要对音视频数据进行较为复杂的处理。

基于多媒体格式的技术则是利用视频的元数据信息进行隐藏，例如视频的拍摄位置、时间戳等元数据信息，这种方法不需要对音视频数据进行调整，嵌入的信息不会影响音视频的质量，但是无法嵌入大量的信息。

流媒体信息隐藏技术可以应用于多个领域，例如数字版权保护、数字水印、信息追踪等，能保护信息的安全性和完整性，但也存在着信息隐藏和检测方面的问题和挑战，需要进一步的改进和研究。

2.2 技术方法

常用的流媒体信息隐藏技术有：

1. 整数离散余弦变换（Integer Discrete Cosine Transform，IDCT）：该技术通过将隐藏数据嵌入到视频图像的低频段来实现。低频部分的图像变化不太明显，使得隐藏的数据不易被发现。

2. 加性白噪声（Additive White Gaussian Noise，AWGN）：该技术是将隐藏数据加入到原始视频的噪声中。需要注意的是，嵌入的信息需要与原始视频噪声的统计特征相似，保证隐藏数据不被攻击者发现。

3. 量化指数调制（Quantization Index Modulation，QIM）：该技术是将隐藏信息嵌入到视频的量化索引中。由于视频量化索引通常拥有一定的冗余性和可变性，因此可以利用这种特性来隐藏数据。

4. 时域编码（Time-Domain Coding，TDC）：该技术是利用音视频流中的同步信号周期性的嵌入隐藏信息。由于嵌入的信号是周期性的，因此与原始音视频信号的差异是非常小的，隐藏信号不容易被发现。

5. 特征点保护（Feature Point Protection，FPP）：该技术是将隐藏数据嵌入到视频中的特定的点上，例如图像的边缘、纹理、颜色等特征点。通过保护重要的特征点，可以有效提高隐藏数据的安全性和可靠性

这些流媒体信息隐藏技术各有优点和不足，应根据需要进行选择和应用。同时，随着信息隐藏技术的不断发展，新的嵌入和检测技术也在不断涌现。

3 ogg 文件格式剖析

3.1 ogg 文件格式简介

Ogg 是一个自由且开放标准的容器格式，由 Xiph.Org 基金会所维护。Ogg 格式并不受到软件专利的限制，并设计用于有效率地流媒体和处理高品质的数字多媒体。

“Ogg” 意指一种文件格式，可以纳入各式各样自由和开放源代码的编解码器，包含音效、视频、文字（像字幕）与元数据的处理。

在 Ogg 的多媒体框架下，Theora 提供有损的图像层面，而通常用音乐导向的 Vorbis 编解码器作为音效层面。针对语音设计的压缩编解码器 Speex 和无损的音效压缩编解码器 FLAC 与 OggPCM 也可能作为音效层面使用。

“Ogg” 这个词汇通常意指 Ogg Vorbis 此一音频文件格式，也就是将 Vorbis 编码的音效包含在 Ogg 的容器中所成的格式。在以往，.ogg 此一扩展名曾经被用在任何 Ogg 支持格式下的内容；但在 2007 年，Xiph.Org 基金会为了向后兼容的考量，提出请求，将.ogg 只留给 Vorbis 格式来使用。Xiph.Org 基金会决定创造一些新的扩展名和媒体格式来描述不同类型的内容，像是只包含音效所用的.oga、包含或不含声音的影片（涵盖 Theora）所用的.ogv 和程序所用的.ogx。

OGGVobis(oggVorbis) 是一种新的音频压缩格式，类似于 MP3 等的音乐格式。Ogg-Vobis 是完全免费、开放和没有专利限制的。OggVorbis 文件的扩展名是.OGG。Ogg 文件格式可以不断地进行大小和音质的改良，而不影响旧有的编码器或播放器。OGG Vorbis 有一个特点是支持多声道。

3.2 简单的文件剖析

Ogg 是一种开放的容器格式，用于存储音频、视频、元数据和字幕等数据类型。Ogg 格式文件通常使用.ogg 文件扩展名，并且允许使用多种编码方式对音频信息进行压缩。下面是 ogg 文件格式的详细介绍。

Ogg 文件格式由 Ogg 分装器和 Ogg 分离器组成，Ogg 分装器用于将不同类型的数据流打包成一个文件，如音频、视频等。Ogg 分离器用于从 Ogg 文件中提取单个数据流，并且可用于将不同类型的数据流复合在一起。这种方式与 MPEG 格式非常相似。Ogg 文件以文件头开头，并且在文件头中包含元数据和分段表。

头码头：Ogg 文件的头码头以字母”OggS” 开头，跟着一字节的版本号。版本号以 0x00 为开头，后面的三个数字分别是主版本、次版本和修订版本号。接下来两个字节是数据流类型的标识符，对于 audio 数据流应该填写为 0x01。然后是标志，标志位描述了流的具体性质，如数据的开头、中间或结尾。然后是一个 8 字节长的 64bit 全局唯一标识符，通常被称为 magic，用于标识 Ogg 文件。接下来的四个字节代表 Ogg 文件的第几个数据段。

元数据：在 Ogg 文件头之后，是一些可选的元数据信息，这些信息通常包括作品的名称、作者和版权信息。元数据信息是 Unicode 字符串，以一个 32 位的 big-endian 字节为前缀，表示这个元数据项包含的字节数。然后是元数据的类型（比如 “title” 或 “artist”），又是一个以 0 结尾的字节字符串。用于与此类型相关的值然后是元数据项的值，同样又是一个以 0 结尾的字节字符串。

数据流：文件头和元数据信息之后，就是 Ogg 文件的音频数据。数据流被分成一个个数据块，称为 Ogg 包，每个包都包含一个数据段的完整数据，以及一个包头。包头中包含了一个长度字段，指示这个数据块包含的数据量。

Ogg 采用公共的而非专利形式，使用 Ogg Vorbis 编解码器将数据压缩成常用音频格式。Ogg 文件格式的开放和非专有特性使得它成为广泛应用于音频和视频的容器格式之一。

3.3 使用 vscode 剖析 ogg 文件格式

OGG 是以页（page）为单位将逻辑流组织链接起来，每个页都有 pageheader 和 pagedata。

每个页之间相互独立，都包含了各自应有的信息，页的大小是可变的，通常为 4K - 8KB，最大值不能超过 65307bytes（27 + 255 + 255*255=65307）。

页标识（0x00-0x03）：ASCII 字符，0x4f 'O' 0x67 'g' 0x67 'g' 0x53 'S'，4 个字节大小，它标识着一个页的开始。其作用是分离 Ogg 封装格式还原媒体编码时识别新页的作用。

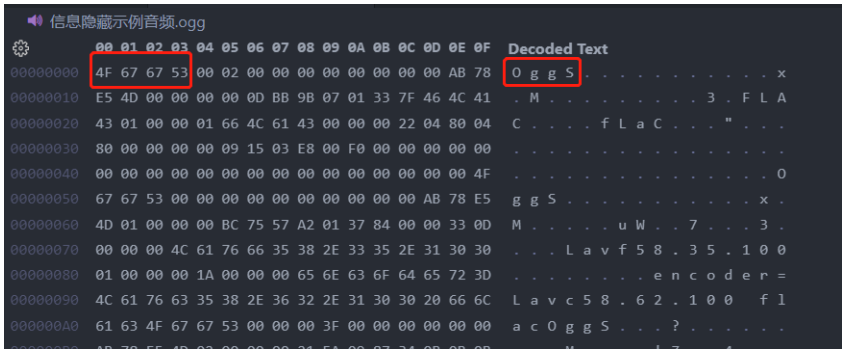

图 3.1: OGGS 页标识

版本 id（0x04）：一般当前版本默认为 0，1 个字节。

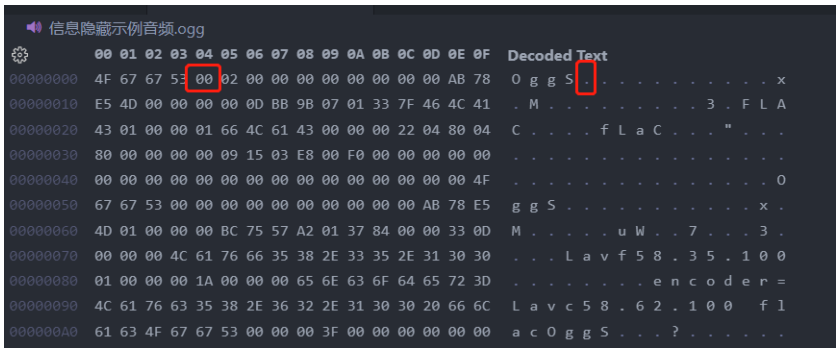

图 3.2: 版本标识

Header_type（0x05）：标识当前的页的类型，1 个字节

- 0x01：本页媒体编码数据与前一页属于同一个逻辑流的同一个 packet，若此位没有设，表示本页是以一个新的 packet 开始的；
- 0x02：表示该页为逻辑流的第一页，bos 标识，如果此位未设置，那表示不是第一页；
- 0x04：表示该页位逻辑流的最后一页，eos 标识，如果此位未设置，那表示本页不是最后一页。

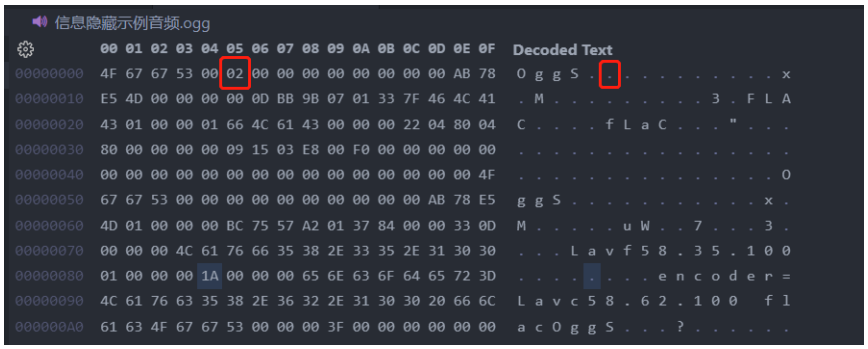

图 3.3: 当前页类型

Granule_position（0x06-0x0d）：媒体编码相关的参数信息，8 个字节，对于音频流来说，它存储着到本页为止逻辑流在 PCM 输出中采样码的数目，可以由它来算得时间戳。对于视频流来说，它存储着到本页为止视频帧编码的数目。若此值为-1，那表示截止到本页，逻辑流的 packet 未结束。在本例中为 0

Serial_number（0x0e-0x11）：当前页中的流的 id，4 个字节，它是区分本页所属逻辑流与其他逻辑流的序号，我们可以通过这个值来划分流。

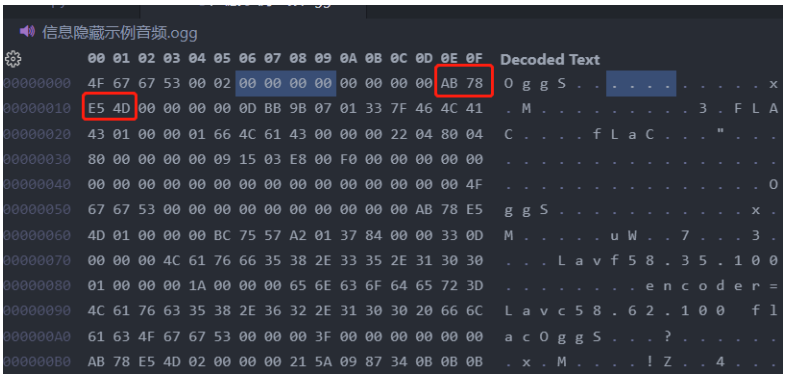

图 3.4: 流 id

Page_seguence(0x12-0x15)：本页在逻辑流的序号，4 个字节。OGG 解码器能据此识别有无页丢失。在本例中为 0

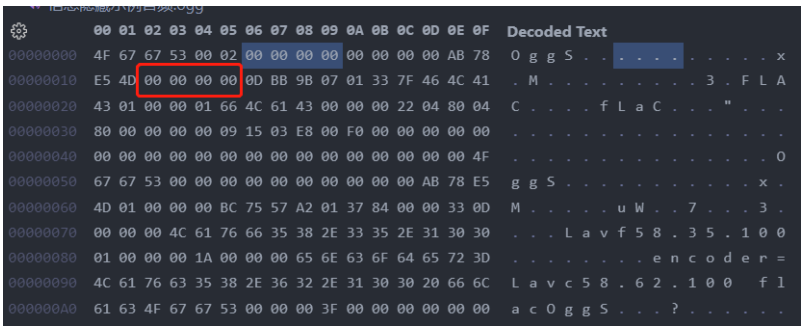

图 3.5: 本页逻辑流编号

CRC_cbecksum(0x16-0x19)：循环冗余校验码校验和，4 个字节，包含页的 32bit CRC

校验和（包括头部零 CRC 校验和页数据校验），它的产生多项式为：0x04c11db7

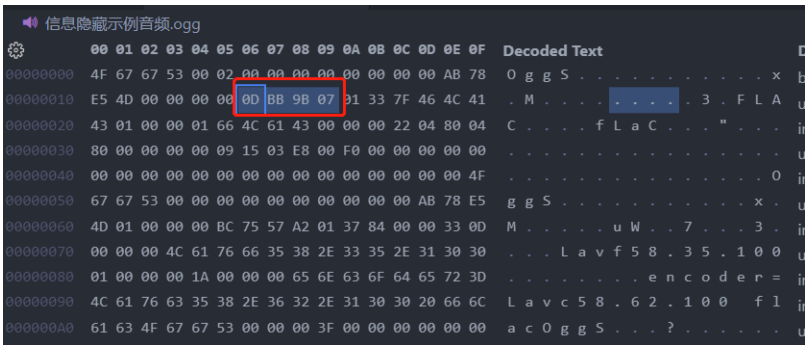

图 3.6: CRC 循环冗余校验和

Num_segments(0x1a)：给定本页在 segment_table 域中出现的 segement 个数，1 个字节。其最大值为 255. 页最大物理尺寸为 65307bytes, 小于 64KB。

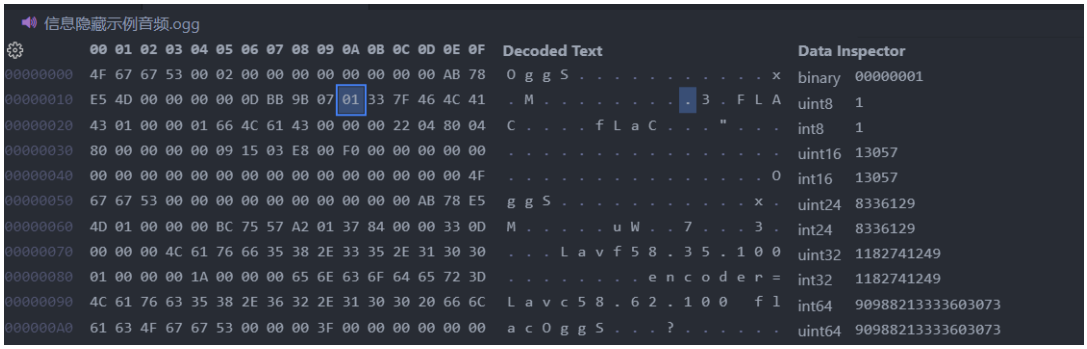

图 3.7: segment 数量

Segment_table(0x1b)：从字面看它就是一个表，表示着每个 segment 的长度，取值范围是 0 255。

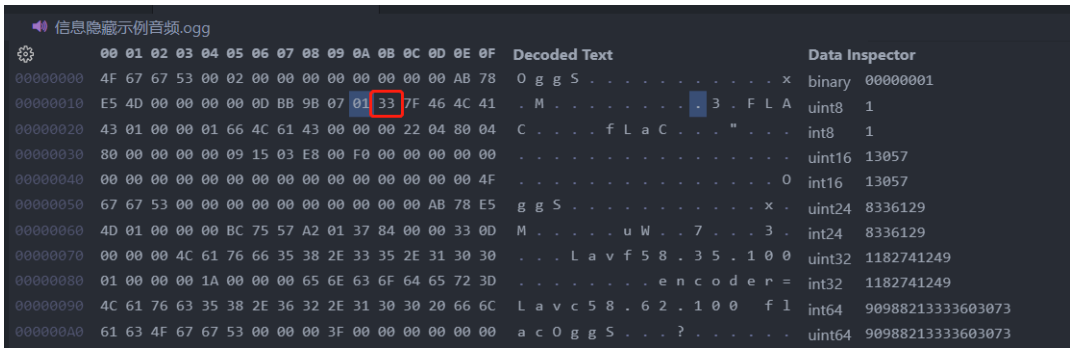

图 3.8: segment 数量

页头基本上就是由上述的参数组成，由此我们可以得到页头的长度和整个页的长度：

$$header\_size = 27 + Num\_segments(byte)$$

$$page\_size = header\_size + segment\_table \quad segment \quad ;$$

后面也有一些比特位是由很大用处的，这里挑一些进行讲解：

0x23-0x26 代表的是版本号，0x27 的位置代表通道数，后面紧跟着的四个比特是采样率

后面很大的一块是音频的注释包

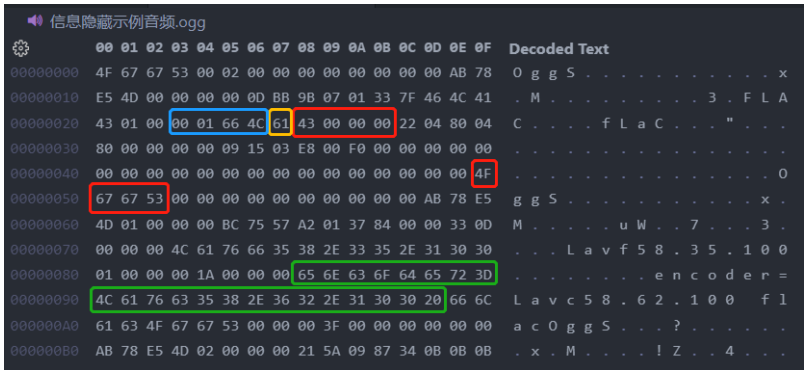

图 3.9: 其他重要比特作用详解

4 信息隐藏可行性分析

4.1 隐藏位置分析

很明显，ogg 文件的格式是以页为单位的，所以页首部我们肯定不能动，前面三个页通常都比较小，不便于我们隐藏数据，我们可以在第三个以后的页数中隐藏数据，那个时候每一个页的大小都会有接近 5000 个字节可以进行隐藏。于是我们根据前面给出的公式：

$$header\_size = 27 + Num\_segments\ (byte)$$

计算出页头的偏移，这里要注意，严格意义上我们需要去对应位置访问以确定究竟有多少个 Num_segments 存在，但是实际上我们只要偏移 27+255 就可以，这样一定是 segments 的位置，可以供我们自由的隐藏信息。

4.2 隐藏方式分析

基于流媒体的 LSB（最低有效位）隐藏法是一种数字水印技术，用于保护数字流的版权和完整性。该方法基于在音频、视频或图像流中嵌入水印信息，并用 LSB 算法将信息隐藏在最低有效位中。以下是 LSB 隐藏法的原理、合理性以及方法的简述：

原理：LSB 隐藏法的基本原理是利用最低有效位（LSB）嵌入二进制水印信息，并在不影响观看或听取媒体文件的质量下实现水印的提取与验证。在 LSB 隐藏法中，嵌入数值较小的水印信息可以通过修改原始数据的最低有效位来实现，并且不影响图像、音频或视频流的可视或听觉质量。因此，使用 LSB 隐藏法嵌入的水印可以对图像、音频和视频流实现隐蔽性保护。

合理性：基于流媒体的 LSB 隐藏法是一种通用和有效的音频、视频或图像保护技术。该技术可以在数字信号中嵌入唯一的标识符，以便对媒体文件的完整性和版权进行验证，从而防止数字文件的非法复制、篡改或盗版。此外，由于其技术简单性和高效性，LSB 隐藏法在音频、视频或图像保护领域中被广泛应用，并已被证明是一种强大且有效的保护机制。

方法：在 LSB 隐藏法中，嵌入的水印信息将压缩为二进制或数字字符串，并用 LSB 方法将信息位插入到音频、视频或图像流的最低有效位中。这些嵌入的信息可以用于对数字媒体进行验证和检查，以确定其是否由原始版权所有人发行。而回收隐藏的水印信息通常需要使用特殊的提取算法，根据 LSB 规律提取最低有效位中的隐藏数据，然后解密并重建水印信息。

综上所述，基于流媒体的 LSB 隐藏法是一种简单、通用且有效的音频、视频或图像保护技术，可以实现数字水印的嵌入和提取，并为数字媒体保护提供了一种隐蔽、高效和安全的解决方案。

5 信息隐藏实践

具体的可以执行的代码已经在下面贴出来了，具体的流程是，先通过读入二进制文件的形式读入.ogg 文件，然后将其转换为可以操作的 hex 格式的 str，然后我们通过寻找 OggS 的 ascii 码找到第三页的位置，然后进行替换。

紧接着，我们加载隐藏后的音频，然后按照加密的格式进行提取文件，随后将其转换为 asc 字符的形式并进行展示，整个 pipeline 的过程中均有 log 输出，可以帮助我们验证每一步的正确性

```
    import struct

'''
将 s 中以 index 为下标的值换成 value
在本代码中使用的十六进制字符串
'''
def replaceHex(s,index,value):
    l = list(s)
    l[index] = value
    res = "".join(l)
    return res


'''
```

```
15  通过输入一个字符串 s, 将其逐个字符转换为 ascii 码格式
16  然后再将 ascii 码转换为二进制码
17  '''
18  def GenInformation2Hide(s):
19      ret = ""
20      for alphabat in s:
21          i = ord(alphabat)
22          binum = bin(i)
23          binum = binum[2:]
24          while len(binum) < 8:
25              binum = '0' + binum
26          ret = ret + binum
27      return ret
28
29  '''
30  在本次实验中将隐藏的信息藏在第四个 OGG 页开始的位置
31  '''
32  def FindStrIndex(s):
33      index1 = s.find('4f6767')
34      index2 = s.find('4f6767',index1+3)
35      index3 = s.find('4f6767',index2+3)
36      index4 = s.find('4f6767',index3+3)
37      return index4
38
39  '''
40  计算应该用哪个字符代替原来的字符 c
41  如果 one 是 true 代表二进制结尾应该是 1
42  '''
43  def CalReplaceChar(c,one):
44      if one:
45          if (c=='1')|(c=='3')|(c=='5')|(c=='7')|(c=='9')|(c=='b')|(c=='d')|(c=='f'):
46              return c
47          if c=='0':
48              return '1'
49          if c=='2':
50              return '3'
51          if c=='4':
```

```
52              return '5'
53          if c=='6':
54              return '7'
55          if c=='8':
56              return '9'
57          if c=='a':
58              return 'b'
59          if c=='c':
60              return 'd'
61          if c=='e':
62              return 'f'
63      else:
64          if (c=='0')|(c=='2')|(c=='4')|(c=='6')|(c=='8')|(c=='a')|(c=='c')|(c=='e'):
65              return c
66          if c=='1':
67              return '0'
68          if c=='3':
69              return '2'
70          if c=='5':
71              return '4'
72          if c=='7':
73              return '6'
74          if c=='9':
75              return '8'
76          if c=='b':
77              return 'a'
78          if c=='d':
79              return 'c'
80          if c=='f':
81              return 'e'
82
83  '''
84  将 hide_information 隐藏到 origin_data 以 index2hide 开始的位置
85  '''
86  def HideInformation(origin_data,index2hide,hide_information):
87      count = 0
88      od = origin_data
```

```
89      for char in hide_information:
90          # print('end = ',char)
91          # print('origin_data = ',od[index2hide+count])
92          # if char == '1':
93          #     print('2Replace = ',CalReplaceChar(od[index2hide+count],True))
94          # if char == '0':
95          #     print('2Replace = ',CalReplaceChar(od[index2hide+count],False))
96          if char =='1':
97              od = replaceHex(od,index2hide+count,CalReplaceChar(od[index2hide+count],
98              pass
99          if char =='0':
100             od = replaceHex(od,index2hide+count,CalReplaceChar(od[index2hide+count],
101         count += 1
102     return od
103
104 '''
105 从 data 的 begin 位置按 lsb 方法提取 l 比特的信息
106 '''
107 def PatchInformation(data,begin,l):
108     infor = ""
109     zero = ['0','2','4','6','8','a','c','e']
110     one = ['1','3','5','7','9','b','d','f']
111     for i in range(0,l):
112         if data[begin+i] in zero:
113             infor = infor + '0'
114         if data[begin+i] in one:
115             infor = infor + '1'
116
117     char_length = l//8
118     ans = ""
119     for i in range(0,char_length):
120         temp = infor[i*8:(i+1)*8]
121         num = int(temp,2)
122         ch = chr(num)
123         ans = ans + ch
124
125     return ans
```

```
126
127
128     '''
129     整个隐藏以及提取信息的 pipeline
130     '''
131     def pipline():
132         binary_data = None
133         hex_data = None
134         with open('信息隐藏示例音频.ogg', 'rb') as f:
135             binary_data = f.read()
136         hex_data = binary_data.hex()
137         print('[PIPELINE LOG] SUCCESSFULLY READ OGG FILE')
138         print('[PIPELINE LOG] DISPLAY HEAD OF THE OGG FILE:',hex_data[:20])
139         index = FindStrIndex(hex_data)
140         index = index+27+255
141         print('[PIPELINE LOG] DISPLAY HIDING INDEX:',index)
142         # 在这里更改你想要隐藏的信息，因为没有 encode 和 decode 所以只能隐藏英文、数字和一
143         hide_infor = GenInformation2Hide('NKU')
144         print('[PIPELINE LOG] INFORMATION 2 HIDE IS: ',hide_infor)
145         hex_data2 = HideInformation(hex_data,index,hide_infor)
146         byte = bytes.fromhex(hex_data2)
147         # 打开文件，并以二进制写入模式写入二进制数据
148         with open('隐藏后音频.ogg', 'wb') as f:
149             f.write(byte)
150         print('[PIPELINE LOG] SUCCESSFULLY DONE INFORMATION HIDING!')
151         binary_data2 = None
152         with open('隐藏后音频.ogg', 'rb') as f:
153             binary_data2 = f.read()
154         hex_data2 = binary_data2.hex()
155         print('[PIPELINE LOG] START PATCHING INFORMATION FROM OGG FILE')
156         print('PIPELINE LOG] SUCCESSFULLY DONE INFORMATION PATCHING! INFORMATION IS :',P
157
158     if __name__ == '__main__':
159         pipline()
```

完整的代码和音频文件可以在我的 github 仓库中找到，仓库连接为：https://github.com/wbf1015/I
_hiding/tree/main/

最后的执行效果图如下所示

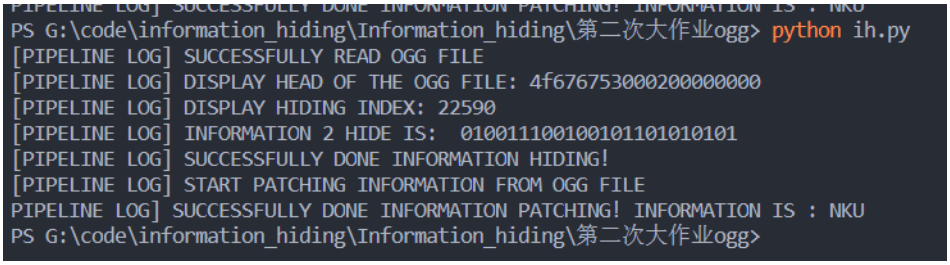

图 5.10: 程序执行效果图