《电子证据分析》课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 实验名称 |  | 姓名 |  |
| 班级 |  | 学号 |  |
| 实验日期 |  | 指导老师 |  |
| 实验环境  (实验软硬件要求) | Win10、python、office2019、bandzip、Stegsolve.jar | | |
| 实验目的 | （给出本次实验所涉及并要求掌握的知识点）  了解docx的相关知识，能够将其中隐藏的字符和图片恢复出来 | | |

|  |  |
| --- | --- |
| 实验内容 | 1、将一个把字符隐藏和一个图片高度宽度设置为0的docx文件里的字符和图片都恢复出来。  新建的文件内容如下：有两行文本和一张图片。    然后将第二段文本在字体中选择隐藏，并在图片中设置宽度和高度为0      这样的结果就是文本内容变成了如下：    可以看到只有一行文本了。  首先还原隐藏的 。全选，然后点击字体，将隐藏项取消即可。    就能看到原本的字回来了。    然后再修改回图片。  保存文件然后退出docx，随后复制一个副本改后缀为zip    解压后这个压缩包后进入：新建 Microsoft Word 文档\word目录中，并且点击document.xml    查看打开    根据课上知识可以知道这两个部分就是宽度和高度  随后使用编辑打开  将这两个地方原本是0的数值修改为较大的数值    然后保存并且将文件压缩回zip，改后缀为docx  然后就能看到图片了，但是这是没办法百分百恢复成为原本的图片的大小的，用这种办法是没办法恢复的，只能将这张照片恢复回来。    附加题目，假如将docx文件里面的字符改成了白色对应的标志位是什么？  首先新建一个文件，然后输入文本。如下：    一般来说都是这样的白底黑字，这里不直接改变，选择使用副本来修改，直接保存并且创建副本。如下：    然后打开副本，将文字改成白色    可以看到都是一片白色。  重新保存后退出，然后我们将这两个本来没有修改过的文件改成zip后缀。      看到的在白色字符的docx文档的对应文本里面多了这一部分，我认为这就是将字符修改为白色的标志位。  2、将所有的LSB中的题目都做出来  使用Stegsolve.jar来打开图片文件      先是steg.png，在Red通道中，低二位和最后一位的图片差异非常大。      同样的，在Green通道和Blue通道里面也都是这样，这就说明有很大概率有LSB隐写的可能。  随后进行如下操作，如图：      果然就是隐藏了一段话，直接导出文本即可。      这样就获取了文件的隐藏信息。  第二个是LSB.bmp    一样的用软件打开图片。  在三个RGB的最低位都发现最后又一行的黑色。这说明很可能是存在信息的  因为很难找到一个能够正好被隐藏进去的寄生数据填满的宿主，所以很多时候找的宿主照片都是会比寄生数据要多很多的。那么最低有效位的部分很多就会又多出的部分要么全部填1（1为白色），或者全部填0（0为黑色）  发现的如下：        然后重新进行分析    发现这是个标准的bmp文件  然后导出为二进制文件并且后缀为bmp  得到如下图片    最后还剩一个题目    这个图片是胜利之吻。很经典  这文件是bmp格式的，用软件查看都没有办法看出问题来，所以使用对应的文件，将其另存为PNG格式，这是因为bmp格式是有损压缩的格式，使用PNG的格式是无损压缩的，可能蕴藏的信息在被压缩掉的部分里面。  我本来是使用的PS的。但是Win10原本没有PS的话可以用下面的画图3D来进行转换，也是可以的。      这样子保存好就行，这时候重新使用软件查看。        在每一个最低位都可以看到这个文件里面又藏着一个二维码  使用软件扫码，得到结果如下：    这个就是答案了……很神奇的是目前我测试下来好像只有微信可以扫出来这个二维码……  3、掌握操作.db文件来进行缩略图的恢复  这个在老师上课讲的时候因为我近视眼……看不到东西，电脑显示器也是坏的……所以基本上没怎么明白，但是大致听明白了就是文件前面有一个记录了长度，后面就是文件的数据，所以按照这个规范，我找了一下LSB文件里面的db文件里面的缩略图都有哪些。  另外环境就是在自己虚拟机的XP中了……  在用winhex打开了db文件之后（没办法devC++没有安装……没有那个工具）往下翻几页就找到了一个缩略图    看得出来这个图片的长度就是0x1061个字节，然后开头起始是0x060C，那么两者加起来就是末尾了，即0x166C(这里算长度一定要注意，就和年份的问题是一样的，算好之后还要-1)将这些部分选中，然后编辑作为新文件输出即可    将其保存在桌面之后，双击查看就是下面这张图片。    然后就是同样的操作继续下去。    这里就简略过程了。  0x1A4C + 0x0E39 – 0x1 = 0x2884 但是这里的图片竟然是没有办法看出来的。    下一张，0x300C + 0x10B4 – 0x1 = 0x40BF    导出为3.jpg    最后这里还有一个……    计算就是0x4F4C + 0x08AE - 0x1= 0x57F9  得到4.jpg    附加题，写一个小程序，将db文件里的缩略图都导出来  import math  import time  def get\_time\_stamp():      # 网上找的，使用当前时间作为文件名      ct = time.time()      local\_time = time.localtime(ct)      data\_head = time.strftime("%Y%m%d%H%M%S", local\_time)      data\_secs = (ct - int(ct)) \* 1000      time\_stamp = "%s\_%03d" % (data\_head, data\_secs)      return time\_stamp  def outputJpg(file):      with open(file,'rb+') as f1:          byt = f1.read(4)          while (byt):              if b'\x0c\x00\x00\x00' == byt:                  byt = f1.read(4)                  if b'\x01\x00\x00\x00' == byt:                      # 判断照片缩略图位置                      list1 = []                      length = ''                      byt = f1.read(4)                      for i in byt:                          list1.append(i)                      list1.reverse()                      for i in list1:                          length = length + hex(i).replace("0x",'').zfill(2)                      length = math.ceil(int(length,16) / 4) \* 4                      content = f1.read(length)                      fileName = get\_time\_stamp()                      with open((fileName + '.jpg'),"wb+") as f2:                          f2.write(content)                          f2.close()              byt = f1.read(4)          f1.close()  def main(argv):      outputJpg(argv[1])  if \_\_name\_\_ == "\_\_main\_\_":      main(sys.argv)  需要使用命令行来操作，如下：    运行结果如下：    结束。 |
| 实验总结 | （对本次实验涉及到的知识点的重难点归纳、实验心得、思考与建议）  本次实验让我们进一步了解了图片隐写的知识，对于相关的LSB的知识有了一定的了解。我发现在做第三个实验的时候有些同学和我的db文件有一些差异，他们的0C00000001000000后面接了4个字节的长度信息，但是后面却不是马上就是文件数据。 |
| 评分 |  |