《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、微信 | | |
| 实验目的 | （给出本次实验所涉及并要求掌握的知识点）  *能够自己掌握并且编写小程序解出微信的图片* | | |

|  |  |
| --- | --- |
| 实验内容 | 编程，编写一个小程序能够将微信自己发送的图片文件按照原来的格式还原出来。  首先我们需要了解微信文件在哪里？  微信文件在C:\Users\<用户名>\Documents\WeChat Files\wxid\_<一堆字符> \FileStorage\Image 里面  并且里面是根据时间分成了多个文件夹的，具体如下：    打开文件夹可以看到都是一些.dat文件    其实，这些文件就是我们需要的图片了。只不过使用了一些特殊的办法进行了一些特殊的操作而已。因为我们可以看到，很多地方会出现连续相同的内容，很明显这个信息熵很低，没有加过密，如下图：    那么，微信到底用了什么操作呢？  其实就是用了一个特定的key，让它来和原本的数据异或，获得的新数据就是那个dat文件了……看起来很简单。  所以我的最初逻辑如下：  第一步，将dat文件的头两个字节先互相异或，然后再和其他的标准文件头异或出来的值进行比较，这样就能找出对应的文件类型。  第二步，将dat文件头第一个字节与已经确定的类型的文件的第一个字节进行异或，这样就相当于是密文（dat文件的第一个字节）与明文（已知的图片类型的第一个字节）异或来得到秘钥key  第三步，将秘钥key与dat文件内容进行不断的异或，并且将结果输出到一个新的文件中去。  这里首先就是异或判断文件类型的部分，这部分代码如下：    这里就是很简单地先将四种常见格式的图片的开头两个字节的异或计算出来并且分别放在对应的常量里面（python约定俗成的全大写表示常量），然后再将进行了异或后的dat文件的开头两个字节进行异或（保存在变量answer里面），随后将answer分别与前面的四个类型进行比较即可，这部分的代码因为我和后面的第二步的代码是黏连在一起的，就没有截图，在第二步里面有。  第二步，因为已经知道了类型，所以接下来就是需要计算秘钥key的值，直接选择用两个文件的开头第一个字节来进行异或，异或出来的结果就是对应key的值！下面就是详细代码：（还包含了上面的一些比较的部分）：    对第一、二步的简单总结：  可以看到，我这里使用了很简单的变量存储已经确定的四种常见图片格式的异或结果，然后使用文件操作获取前两个字节，再将其异或（这里就是我的前面的逻辑的第一步），然后再将其与之前的存放好的变量进行比较，比较完成之后再将dat文件的第一个字节和确定的类型的第一个字节进行异或，得到秘钥key（这里就是我前面说的第二步），然后返回对应的秘钥key和对应的类型的后缀字符串。（这里插一句，本来没有想着返回类型的字符串，结果发现后面再加上一个类型后缀的话就能很方便地完成后面的创建新的图片文件这一步，所以又添加了这个）  第三步，在获取了key之后就是直接进行全部的异或然后写新的图片文件了。  我因为有部分的代码选使用了之前实验写的代码（避免重复造轮子）所以这一个函数又被分成了多个小的功能函数组成。  主要的第三步的函数如下：    这里的逻辑应该是比较清楚的，打开文件（这里其实还可以做一个异常抛出，不过后面我还对这整个py程序进行了补全，所以它这里也不需要进行异常处理，一般来讲是不太要紧的），获取文件的大小长度，利用前面的函数获取了秘钥和类型之后（这里都是第一、第二步的内容），先读取掉dat文件里面所有的数据放到变量byte里面，然后调用函数BytToInt（这个函数的功能就是将一整串的byte流转换成对应的数值，实现起来很容易，后面会说）并且将获取的字符串全保存在变量ciphertext里面，之后调用函数Xor（看这个名字就知道，这个函数的作用和就是异或了，返回值就是明文了）返回值保存在answer里面，那么后面就是写入文件了，这里直接将原先的名字作为解出来的照片的名字，然后再加上正确的后缀名就行了  另外，这里其实写入的时候还用了一个struct的库，将所有前面异或好了的数字值将其再打包成为byte类  接下来解释一下之前没有明说的两个函数，一个是BytToStr还有一个就是Xor函数。  第一个BytToStr函数：    这个函数也其实是投机取巧的行为，因为python3.8（我使用的版本）在进行字节串的for循环遍历的时候会自己将原本是字节串的内容转换成对应的数值，所以我也就是直接利用了一个for循环而已。    第二个函数就是Xor：    这个函数就是很简单的异或操作，密文与秘钥异或得到原文，然后这个完整的原文作为一个返回值返回  主要的简单逻辑了  后面为了将其更加具有较强的功能性，我增加了能够支持递归解析子文件夹以及自动排除非dat文件（原理就是直接识别文件后缀，只要不是dat后缀的就直接跳过）。这样的同时也满足了避免出现了输入错误的路径会报错的问题。代码如下：    也其实是从上几次的作业里修改而来的，我只是改了改而已……  最后的调用：    下面就是完整的代码：  import binascii  import struct  import os  # 一个神奇的数学原理，所以，只要用原先压缩过的dat文件的值进行开头的异或，就能知道值了  # DA = A ^ K  # DB = B ^ K  # DA ^ DB = (A ^ K) ^ (B ^ K) = (A ^ B) ^ (K ^ K) = A ^ B  def IsJpgOrPng(dir):      PNG = 0x89 ^ 0x50      JPG = 0xff ^ 0xd8      TRFF = 0x49 ^ 0x49      BMP = 0x42 ^ 0x4d      with open(dir,"rb") as f1:          number = f1.read(2)          answer = int(number[0]) ^ int(number[1])          if (answer == PNG):              # print("%s is a PNG"%dir)              key = int(number[0]) ^ 0x89              kind = "png"          elif (answer == JPG):              # print("%s is a JPG"%dir)              key = int(number[0]) ^ 0xff              kind = "jpg"          elif (answer == TRFF):              key = int(number[0]) ^ 0x49              kind = "trff"          elif (answer == BMP):              key = int(number[0]) ^ 0x42              kind = "bmp"      return key,kind  def Xor(ciphertext,key):      answer = []      for i in ciphertext:          i = i ^ key          answer.append(i)      return answer  def BytToInt(string):      """此函数的作用就是将读取到的16进制字符流变成正常的字符列表，方便后面的异或"""      hexs = []      for s in string:          hexs.append(s)      return hexs  def change(file1):      f1 = open(r'%s'%file1,'rb')        length = os.path.getsize(r'%s'%file1)      # 动态获取要读取的文件的大小      key,kind = IsJpgOrPng(file1)      # 我的秘钥是28      byte = f1.read(length)      # 将读取到的内容先全部放入一个变量里面      ciphertext = BytToInt(byte)      answer = Xor(ciphertext,key)      filename = file1[:-4]        with open(r"%s.%s" % (filename, kind),"ab+") as fout:          for i in answer:              a = struct.pack('B',i)              fout.write(a)  def RecursiveFileSearch(path):      if (not os.path.exists(path)):  # 判断文件(文件夹)是否存在          print(path + "不存在！！")          return 0      if (os.path.isfile(path) and path[-3:] == 'dat') :          change(path)          print("文件%s已经修改完成"%path)      elif (os.path.isdir(path)):          dirlist = os.listdir(path)          for dir in dirlist:              dir = path + '\\' + dir              RecursiveFileSearch(dir)  if \_\_name\_\_ == "\_\_main\_\_":      path  = input("请输入文件夹或文件位置，支持自动递归解码，解码完成后的图片会保存在原来dat文件的位置\n")      RecursiveFileSearch(path) |
| 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） |
| 评分 |  |