Python 实现图片隐写术

原创

zerowwwww 于 2021-12-31 18:05:47 发布 2235 收藏
分类专栏： 蓝桥云课 文章标签： python 开发语言 后端

一、实验简介

wikipedia 关于隐写术的介绍：

> **隐写术**是一门关于信息隐藏的技巧与科学，所谓信息隐藏指的是不让除预期的接收者之外的任何人知晓信息的传递事件或者信息的内容。隐写术的英文叫做 Steganography，来源于特里特米乌斯的一本讲述密码学与隐写术的著作 _Steganographia_，该书书名源于希腊语，意为“隐秘书写”。

1.1 知识点

- Pillow 模块
- 最低有效位
- lambda 表达式递归
- UTF-8 编码

二、实验步骤

本节将通过实践操作，带领大家使用 Python3 实现图片隐写术。

2.1 实验原理

还是引用 wikipedia 的解释：

> 载体文件（cover file）相对隐秘文件的大小（指数据含量，以比特计）越大，隐藏后者就越加容易。
>
> 因为这个原因，数字图像（包含有大量的数据）在因特网和其它媒介上被广泛用于隐藏消息。这种方法使用的广泛程度无从查考。例如：一个 24 位的位图中的每个像素的三个颜色分量（红，绿和蓝）各使用 8 个比特来表示。如果我们只考虑蓝色的话，就是说有 2^8 种不同的数值来表示深浅不同的蓝色。而像 11111111 和 11111110 这两个值所表示的蓝色，人眼几乎无法区分。因此，这个最低有效位就可以用来存储颜色之外的信息，而且在某种程度上几乎是检测不到的。如果对红色和绿色进行同样的操作，就可以在差不多三个像素中存储一个字节的信息。
>
> 更正式一点地说，使隐写的信息难以探测的，也就是保证 “有效载荷”（需要被隐蔽的信号）对 “载体”（即原始的信号）的调制对载体的影响看起来（理想状况下甚至在统计上）可以忽略。这就是说，这种改变应该无法与载体中的噪声加以区别。
>
> （从信息论的观点来看，这就是说信道的容量必须大于传输“表面上”的信号的需求。这就叫做信道的冗余。对于一幅数字图像，这种冗余可能是成像单元的噪声；对于数字音频，可能是录音或者放大设备所产生的噪声。任何有着模拟放大级的系统都会有所谓的热噪声（或称 “1/f” 噪声），这可以用作掩饰。另外，有损压缩技术（如 JPEG）会在解压后的数据中引入一些误差，利用这些误差作隐写术用途也是可能的。）
>
> 隐写术也可以用作数字水印，这里一条消息（往往只是一个标识符）被隐藏到一幅图像中，使得其来源能够被跟踪或校验。

总而言之，本实验便是利用图片四个颜色分量（rgba）的最低有效位（英语：Least Significant Bit，lsb）来隐藏信息（本实验隐藏的是文字）。

2.2 安装包

本实验用到了 pillow 这个模块，实验环境中已经安装了 Python3 的包管理工具 pip3 ，升级 pip3 然后安装 pillow 模块。终端执行以下命令：

```
$ sudo pip3 install -U pip        # 升级包管理工具 pip3

$ sudo pip3 install pillow        # 安装所需工具包 pillow
```

2.3 编码

我们将以下代码写入 /home/shiyanlou/Code/steganography.py 脚本文件中。

先导入 Pillow 模块和 sys 模块：

```
import sys
from PIL import Image
```

根据前文所述实验原理创建一个函数，用于将图片的像素点数据值的二进制末位全部变成 0 ，以便后续事宜末位存储数据。该函数命名为 make_even_image ，它接收一个图片对象作为参数，返回一个处理后的图片对象：

```
def make_even_image(image):
    """取得一个 PIL 图像并且更改所有值为偶数，使最低有效位为 0
    """
    # image.getdata 方法返回的是一个可迭代对象，其中包含图片中所有像素点的数据
    # 每个像素点表示一个颜色，每种颜色有红绿蓝三种颜色按比例构成
    # R Red 红色；G Green 绿色；B Blue 蓝色；A Alpha 透明度
    # 更改所有像素点中四个数值为偶数（魔法般的移位）
    # 这里使用位运算符 >> 和 << 来实现数据处理
    # 奇数减一变偶数，偶数不变，这样处理后，所有数值的最低位变为零
    # pixels 为更改后的像素点数据列表
    pixels = [(r >> 1 << 1, g >> 1 << 1, b >> 1 << 1, a >> 1 << 1)
            for r, g, b, a in image.getdata()]
    # 调用 Image 的 new 方法创建一个相同大小的图片副本
    # 参数为模式（字符串）和规格（二元元组）
    # 这里使用 image 的属性值即可
    even_image = Image.new(image.mode, image.size)
    # 把处理之后的像素点数据写入副本图片
    even_image.putdata(pixels)
    return even_image
```

图片处理功能写好之后，就可以向图片对象中写入信息了，也就是把字符串编码到图片中。完成这一功能的函数为 encode_data_in_image ，该函数接收两个参数：图片对象和要隐藏到图片中的信息，函数的返回值是隐藏了字符串信息的新图片。函数代码如下：

```
def encode_data_in_image(image, data):
    """将字符串编码到图片中
    """
    # 获得最低有效位为 0 的图片副本
    even_image = make_even_image(image)
    # 匿名函数用于将十进制数值转换成 8 位二进制数值的字符串
    int_to_binary_str = lambda i: '0' * (8 - len(bin(i)[2:])) + bin(i)[2:]
    # 将需要隐藏的字符串转换成二进制字符串
    # 每个字符转换成二进制之后，对应一个或多个字节码
    # 每个字节码为一个十进制数值，将其转换为 8 位二进制字符串后相加
    binary = ''.join(map(int_to_binary_str, bytearray(data, 'utf-8')))
    # 每个像素点的 RGBA 数据的最低位都已经空出来，分别可以存储一个二进制数据
    # 所以图片可以存储的最大二进制数据的位数是像素点数量的 4 倍
    # 如果需要隐藏的字符串转换成二进制字符串之后的长度超过这个数，抛出异常
    if len(binary) > len(even_image.getdata()) * 4:
        raise Exception("Error: Can't encode more than " +
                len(even_image.getdata()) * 4 + " bits in this image. ")
    # 二进制字符串 binary 的长度一定是 8 的倍数
    # 将二进制字符串信息编码进像素点中
    # 当二进制字符串的长度大于像素点索引乘以 4 时
    # 这些像素点用于存储数据
    # 否则，像素点内 RGBT 数值不变
    encoded_pixels = [(r+int(binary[index*4+0]),
                       g+int(binary[index*4+1]),
                       b+int(binary[index*4+2]),
                       t+int(binary[index*4+3]))
            if index * 4 < len(binary) else (r,g,b,t)
            for index, (r, g, b, t) in enumerate(even_image.getdata())]
    # 创建新图片以存放编码后的像素
    encoded_image = Image.new(even_image.mode, even_image.size)
    # 把处理之后的像素点数据写入新图片
    encoded_image.putdata(encoded_pixels)
    # 返回图片对象
    return encoded_image
```

相关函数的文档链接：

> Image.getdata()
> PIL.Image.new()
> PIL.Image.Image.putdata()

`encode_data_in_image` 函数中，`bytearray` 方法将字符串转换为整数值序列（数字范围是 0 到 2^8-1），数值序列由字符串的字节数据转换而来，如下图：

utf-8 编码的一个中文字符就占了 3 个字节，那么四个字符共占 3×4=12 个字节，于是共有 12 个数值。然后 `map(int_to_binary_str, bytearray(data, 'utf-8'))` 对数值序列中的每一个值应用 `int_to_binary_str` 匿名函数，将十进制数值序列转换为二进制字符串序列。匿名函数里 `bin` 方法的作用是将一个 int 值转换为二进制字符串，详见： Built-in Functions — Python 3.10.1 documentation