忍法—图片隐写术

转载

mjTree 于 2019-05-26 13:50:30 发布 1165 收藏 2
分类专栏： 趣味代码 文章标签： 图片处理 隐写术 Pillow库


趣味代码 专栏收录该内容
12 篇文章 1 订阅
订阅专栏

隐写术：是一门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。

> 载体文件相对隐秘文件的大小（指数据含量）越大，隐藏后者就越加容易。
>
> 因为这个原因，数字图像在因特网和其他传媒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。(百度百科)

这是实验楼的一个项目，记得以前有一家公司B爬取了同行另一家公司A官网的照片挂在自己的网站主页上，最后B被A告了。证据就是A公司官网的图片中通过隐写术的黑技术将信息写入图片中，而这种方法，肉眼是看不出来的。


就像这两张图片，无法看出存在什么差别。但是右边那张是处理过的，里面加入了几个汉字。看完文章后可以自行解码查看信息。

这次要用到的是python3的pillow库，一个用于图片操作的库。源代码如下：

```
#coding:utf-8
# Python3 图片隐写术

from PIL import Image

"""
function: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)
"""
def makeImageEven(image):
    pixels = list(image.getdata())    # 得到一个[(r,g,b,t)]列表
    # 更改所有值为偶数(魔法般的移位)
    evenPixels = [(r>>1<<1,g>>1<<1,b>>1<<1,t>>1<<1) for [r,g,b,t] in pixels]
    evenImage = Image.new(image.mode, image.size)    # 创建一个相同大小的图片副本
    evenImage.putdata(evenPixels)    # 把上面的像素放入到图片副本
```

```
    return evenImage


"""
function:内置函数bin()的替代,返回固定长度的二进制字符串
"""
def constLenBin(int):
    # 去掉 bin()返回的二进制字符串中的'0b',并在左边补足'0'直到字符串长度为8
    binary = "0"*(8-(len(bin(int))-2))+bin(int).replace('0b','')
    return binary

"""
funtion:将字符串编码到图片中
"""
def encodeDataInImage(image, data):
    evenImage = makeImageEven(image)    # 获得最低有效位为0的图片副本
    binary = ''.join(map(constLenBin, bytearray(data, 'utf-8')))  # 将要隐藏字符串转成2进制
    # 如果不能编码全部数据则抛出异常
    if len(binary) > len(image.getdata()) * 4:
        raise Exception('在此图像中不能编码超过'+ len(evenImage.getdata())*4 +'位')
    # 将 binary 中的二进制字符串信息编码进像素里
    encodedPixels = [(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(list(evenImage.getdata()))]
    encodedImage = Image.new(evenImage.mode, evenImage.size)  # 创建新图片以存放编码后的像素
    encodedImage.putdata(encodedPixels)    # 添加编码后的数据
    return encodedImage

"""
function:从二进制字符串转为 UTF-8 字符串
"""
def binaryToString(binary):
    index = 0
    string = []
    rec = lambda x, i: x[2:8] + (rec(x[8:],i-1) if i>1 else '') if x else ''
    # rec = lambda x, i: x and (x[2:8] + (i > 1 and rec(x[8:], i-1) or '')) or ''
    fun = lambda x,i:x[i+1:8] + rec(x[8:],i-1)
    while index+1 < len(binary):
        chartype = binary[index:].index('0')
        length = chartype*8 if chartype else 8
        string.append(chr(int(fun(binary[index:index+length],chartype),2)))
        index += length
    return ''.join(string)

"""
function:解码隐藏数据
"""
def decodeImage(image):
    pixels = list(image.getdata())  # 获得像素列表
    # 提取图片中所有最低有效位中的数据
    binary = ''.join([str(int(r>>1<<1!=r))+str(int(g>>1<<1!=g))+str(int(b>>1<<1!=b))\
                      +str(int(t>>1<<1!=t)) for (r,g,b,t) in pixels])
    # 找到数据截止处的索引
    locationDoubleNull = binary.find('0000000000000000')
    endIndex = locationDoubleNull+(8-(locationDoubleNull % 8)) if locationDoubleNull%8 != 0 else locationDo
    data = binaryToString(binary[0:endIndex])
    return data
```

```
encodeDataInImage(Image.open("doraemon.png"), '我腿短呀').save('doraemon1.png')
print(decodeImage(Image.open("doraemon1.png")))
```

原理大致就是查找图片像素点中可以存储信息的位置，然后把我们所要存储的信息编码成二进制信息放在其中。