python图片隐写_python图片隐写术

weixin_39669701 于 2020-12-09 19:46:33 发布 135 收藏
文章标签： python图片隐写

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

1.2. 效果展示

可以看到“施法”前后的图片肉眼看不出区别，然而图片却真实的隐藏了一些数据在里面。

二、实验步骤

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，在安装它前更新源

2.3. 程序实现

先导入 Pillow 模块：

from PIL import Image

Image.open("my_handsome_photo.jpg"), '渣男特征长得帅，除了我！'

(,

'渣男特征长得帅，除了我！')

(image, data) = Image.open("my_handsome_photo.jpg"), '渣男特征长得帅，除了我！'

image, data

pixels = list(image.getdata())

pixels

[(185, 40, 37),

(185, 40, 37),

(185, 40, 37),

(184, 39, 36),

(185, 40, 37),

.....

(186, 41, 38),

(86, 52, 17),

(87, 53, 18),

(87, 53, 18),

(88, 54, 19),

...]

2.3.1. 编码

我们首先设计将隐藏信息编码到图片中的函数 encodeDataInImage()，其有两个参数，分别是用作载体的图片对象和需要被隐藏的字符串。也就是说我们可以这样调用它：

def makeImageEven(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)

"""

#得到一个这样的列表：[(r,g,b,t),(r,g,b,t)...]

```
pixels = list(image.getdata())

# 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]

# 创建一个相同大小的图片副本

evenImage = Image.new(image.mode, image.size)

# 把上面的像素放入到图片副本

evenImage.putdata(evenPixels)

return evenImage

ef encodeDataInImage(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

evenImage = makeImageEven(image)

# 将需要被隐藏的字符串转换成二进制字符串

binary = ''.join(map(constLenBin, bytearray(data, 'utf-8')))

if len(binary) > len(image.getdata()) * 4:

# 如果不可能编码全部数据，跑出异常

raise Exception("Error: Can't encode more than" + len(evenImage.getdata()) * 4 + " bits in this image. ")

# 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()))]

# 创建新图片以存放编码后的像素

encodedImage = Image.new(evenImage.mode, evenImage.size)

# 添加编码后的数据

encodedImage.putdata(encodedPixels)

return encodedImage

encodeDataInImage(Image.open("my_handsome_photo.jpg"), '渣男特征长得帅，除了我！')

---------------------------------------------------------------------------

ValueError Traceback (most recent call last)

in ()

----> 1 encodeDataInImage(Image.open("my_handsome_photo.jpg"), '渣男特征长得帅，除了我！')

in encodeDataInImage(image, data)
```

4 #数值序列由字符串的字节数据转换而来

5

----> 6 evenImage = makeImageEven(image) # 获得最低有效位为 0 的图片副本

7

8 binary = ''.join(map(constLenBin,bytearray(data, 'utf-8'))) # 将需要被隐藏的字符串转换成二进制字符串

in makeImageEven(image)

6 pixels = list(image.getdata())

7 # 更改所有值为偶数(魔法般的移位)

----> 8 evenPixels = [(r>>1<<1,g>>1<<1,b>>1<<1,t>>1<<1) for [r,g,b,t] in pixels]

9 # 创建一个相同大小的图片副本

10 evenImage = Image.new(image.mode, image.size)

in (.0)

6 pixels = list(image.getdata())

7 # 更改所有值为偶数(魔法般的移位)

----> 8 evenPixels = [(r>>1<<1,g>>1<<1,b>>1<<1,t>>1<<1) for [r,g,b,t] in pixels]

9 # 创建一个相同大小的图片副本

10 evenImage = Image.new(image.mode, image.size)

ValueError: not enough values to unpack (expected 4, got 3)

？？？问题所在点

(utf-8 编码的中文字符一个就占了3个字节)

然后 map(constLenBin,bytearray(data, ‘utf-8’)) 对数值序列中的每一个值应用 constLenBin() 函数，将十进制数值序列转换为二进制字符串序列。

def constLenBin(int):

"""

内置函数bin()的替代，返回固定长度的二进制字符串

"""

#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

2.3.1. 解码

decodeImage() 返回图片解码后的隐藏文字，其接受一个图片对象参数。

```
def decodeImage(image):

pixels = list(image.getdata()) # 获得像素列表

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]) # 提取图片中所有最低有效位中的数据

# 找到数据截止处的索引

locationDoubleNull = binary.find('0000000000000000')

endIndex = locationDoubleNull+(8-(locationDoubleNull % 8)) if locationDoubleNull%8 != 0 else locationDoubleNull

data = binaryToString(binary[0:endIndex])

return data

str.find()

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)
```

reference

实验楼图片隐写术

pillow包安装

相关函数的文档链接：

Image.getdata()

PIL.Image.new()

PIL.Image.Image.putdata()

bin()的作用:是将一个 int 值转换为二进制字符串

str.find()的作用