python处理图片隐写分析_Python3 图片隐写术

原创

大福 mkq0.~~ 于 2021-01-14 19:20:20 发布 284 收藏 3
文章标签： python处理图片隐写分析

Lsp最低有效位隐写术，本质上是变更了原图片颜色分量的位图数据进行了极低的修改，达到人眼不能识别图片区别。其根本是在一张图片上有规则添加了些东西，然后通过工具解析。

难点：python中可用lambda表达式写递归，unicode变长编码

完整代码

```
#! py -3

# coding: utf-8

from PIL import Image

# 待隐写图片初始化

def makeImageEven(image):

# 得到类表[(r,g,b,t),(r,g,b,t)...]

pixels = list(image.getdata())

# 将每个像素的最低有效位初始化为0

evenPixels = [(r >> 1 << 1, g >> 1 << 1, b >> 1 << 1, t >> 1 << 1)

for [r, g, b, t] in pixels]

# 创建原图副本

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

# 把擦除的像素放入副本

evenImage.putdata(evenPixels)

# 返回初始化的隐写图片

return evenImage

def constLenBin(int):

# 去掉bin()返回的二进制字符串的'0b'

# 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

# 隐写数据
```

```
def encodeDataInImage(image, data):
# 获得最低有效位为0的图片副本
evenImage = makeImageEven(image)
# 将被隐藏的字符串转换成二进制字符串
binary = ''.join(map(constLenBin, bytearray(data, 'utf-8')))
# 数据信息溢出图片空间，则抛异常
# 每个像素rgba有四个分量，即四个最低有效位
if len(binary) > len(image.getdata()) * 4:
raise Exception("Error: Can't encode more than " +
len(evenImage.getdata())*4+" bits in this image. ")
# 将字符二进制位数写入遍历像素的最低有效位
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()))]
# 创建新图存放编码后的像素
encodedImage = Image.new(evenImage.mode, evenImage.size)
# 添加数据
encodedImage.putdata(encodedPixels)
return encodedImage
# 解码接受图片对象参数
def decodeImage(image):
# 获取像素列表
pixels = list(image.getdata())
# 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
# 中文字符utf-8两个字节，16个二进制位
locationDoubleNull = binary.find('0000000000000000')
# 不足位补零
endIndex = locationDoubleNull + \
```

```
(8-(locationDoubleNull % 8)
) if locationDoubleNull % 8 != 0 else locationDoubleNull
data = binaryToString(binary[0:endIndex])
return data
# 位字节->字符
# 将提取出来的二进制字符转换为隐藏的文本
def binaryToString(binary):
index = 0
string = []
# 提取码点真正的字符数据
# 区分字节标志与字符数据
def rec(x, i): return x[2:8] + \
(rec(x[8:], i-1) if i > 1 else '') if x else ''
# 获取字符数据
def fun(x, i): return x[i+1:8] + rec(x[8:], i-1)
while index + 1 < len(binary):
# 分析字节序列
# 字符的字节数据中，第一个字节开头 1 的数目便是字符所占的字节数
chartype = binary[index:].index('0')
length = chartype*8 if chartype else 8
# chr()将unicode码点转为对应字符
string.append(chr(int(fun(binary[index:index+length], chartype), 2)))
index += length
return ''.join(string)
if __name__ == '__main__':
encodeDataInImage(Image.open('coffee.png'),
'你好世界, hello World!').save('encodeImage.png')
print(decodeImage(Image.open("encodeImage.png")))
```

效果

```
D:\code-base\python\syl\pillow-hidden> py -3 "d:\code-base\python\syl\pillow-hidden\stegan.py"
你好世界, hello World!
```