python图片隐写_Python3 图片隐写术的实现

weixin_39737111 于 2020-12-09 19:46:38 发布 308 收藏
文章标签： python图片隐写

UPDATE：原本的实现程序只能处理 png 格式的图片，笔者考虑添加对 jpg 格式的支持；经过一段时间的尝试，最后发现因为此方案是基于 Pillow 模块实现对图片文件的处理，而 Pillow 模块在保存 jpg 格式图片时默认会将色彩信息的最后一位去除，为有损压缩，故加密信息会在最后的保存步骤中被抹去，所以暂时不支持 jpg 格式的图片文件。

背景简介

维基百科中关于隐写术的介绍：

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”。

知识点

Pillow 模块

最低有效位

lambda 表达式递归

UTF-8 编码

效果展示

加入隐藏信息后，通过对比，加密前后的图片并不能通过人眼看出区别。

具体实现

隐藏原理

引用维基百科的解释：

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28 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。

总之，此次的实现程序是利用PNG格式图片每个像素的四个颜色分量「rgba」的最低有效位来隐藏信息，把文本信息转为比特信息再填入色彩数值的最后一个比特位。

依赖模块

该 Python 程序需要 Pillow 模块的图片处理功能和 docopt 模块的命令行参数解析功能。

在笔者的 Mac 环境下，用 homebrew 可轻松实现 python3 的安装：

Shell

$ brew install python3

1

$brewinstallpython3

然后用 pip 软件包管理系统可以轻松安装 Pillow 和 docopt 模块：

Shell

$ sudo pip3 install Pillow docopt

1

$sudopip3installPillowdocopt

程序实现

先导入必要的模块：

Python

from PIL import Image

from docopt import docopt

1

2

fromPILimportImage

fromdocoptimportdocopt

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：

Python

encodeDataInImage(Image.open("coffee.png"), '你好世界，Hello world!')

1

encodeDataInImage(Image.open("coffee.png"),'你好世界，Hello world!')

encodeDataInImage() 函数如下：

Python

```
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'))) # 将需要被隐藏的字符串转换成二进制字符串

if len(binary) > len(image.getdata()) * 4: # 如果不可能编码全部数据， 抛出异常

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

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()))] # 将 binary 中的二进制字
符串信息编码进像素里

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
```

1

2

3

4

5

6

7

8

9

```
defencodeDataInImage(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')))# 将需要被隐藏的字符串转换成二进制字符串

iflen(binary)>len(image.getdata())*4:# 如果不可能编码全部数据， 抛出异常

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

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]))ifindex*4

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

returnencodedImage
```

makeImageEven() 函数的实现如下：

Python

```
def makeImageEven(image):
pixels = list(image.getdata()) # 得到一个这样的列表： [(r,g,b,t),(r,g,b,t)...]
evenPixels = [(r>>1<<1,g>>1<<1,b>>1<<1,t>>1<<1) for [r,g,b,t] in pixels] # 更改所有值为偶数(魔法般的移位)
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
```

1

2

3

4

5

6

```
defmakeImageEven(image):
pixels=list(image.getdata())# 得到一个这样的列表： [(r,g,b,t),(r,g,b,t)...]
evenPixels=[(r>>1<<1,g>>1<<1,b>>1<<1,t>>1<<1)for[r,g,b,t]inpixels]# 更改所有值为偶数(魔法般的移位)
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
returnevenImage
```

相关函数的文档链接：

encodeDataInImage() 中，bytearray() 将字符串转换为整数值序列(数字范围是 0 到 28 -1)，数值序列由字符串的字节数据转换而来，如下图：

utf-8 编码的中文字符一个就占了3个字节，那么四个字符共占 3×4=12 个字节，于是共有12个数字。

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。

Python

```
def constLenBin(int):
binary = "0"*(8-(len(bin(int))-2))+bin(int).replace('0b','') # 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
return binary
```

1

2

3

defconstLenBin(int):

binary="0"*(8-(len(bin(int))-2))+bin(int).replace('0b','')# 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

returnbinary

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。

Python

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

1

2

3

4

5

6

7

8

defdecodeImage(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)inp
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))iflocationDoubleNull%8!=0elselocationDoubleNull

data=binaryToString(binary[0:endIndex])

returndata

找到数据截止处所用的字符串 '0000000000000000' 很有意思，他的长度为16，而不是直觉上的 8，因为两个包含数据的字节的接触部分可能有 8 个 0。

binaryToString() 函数将提取出来的二进制字符串转换为隐藏的文本：

Python

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)

1

2

3

4

5

6

7

8

9

10

11

12

defbinaryToString(binary):

index=0

string=[]

rec=lambdax,i:x[2:8]+(rec(x[8:],i-1)ifi>1else'')ifxelse''

rec = lambda x, i: x and (x[2:8] + (i > 1 and rec(x[8:], i-1) or '')) or ''

fun=lambdax,i:x[i+1:8]+rec(x[8:],i-1)

whileindex+1

chartype=binary[index:].index('0')

length=chartype*8ifchartypeelse8

string.append(chr(int(fun(binary[index:index+length],chartype),2)))

index+=length

return''.join(string)

要看明白这个，必须要先搞懂 UTF-8 编码的方式，可以在 wikipedia 上了解 UTF-8 编码：https://zh.wikipedia.org/wiki/UTF-8

UTF-8 是 UNICODE的一种变长度的编码表达方式，也就是说一个字符串中，不同的字符所占的字节数不一定相同，这就给我们的中文支持工作带来了一点复杂度。

码点的位数

码点起值

码点终值

字节序列

Byte 1

Byte 2

Byte 3

Byte 4

Byte 5

Byte 6

7

U+0000

U+007F

1

0xxxxxxx

11

U+0080

U+07FF

2

110xxxxx

10xxxxxx

16

U+0800

U+FFFF

3

1110xxxx

10xxxxxx

10xxxxxx

21

U+10000

U+1FFFFF

4

11110xxx

10xxxxxx

10xxxxxx

10xxxxxx

26

U+200000

U+3FFFFFF

5

111110xx

10xxxxxx

10xxxxxx

10xxxxxx

10xxxxxx

31

U+4000000

U+7FFFFFFF

6

1111110x

10xxxxxx

10xxxxxx

10xxxxxx

10xxxxxx

10xxxxxx

在上图中，只有 x 所在的位置「也即是字节中第一个 0 之后的数据」存储的是真正的字符数据，因此我们使用下面两个匿名函数来提取出这些数据：

Python

rec = lambda x, i: x[2:8] + (rec(x[8:], i-1) if i > 1 else '') if x else ''

fun = lambda x, i: x[i+1:8] + rec(x[8:], i-1)

1

2

rec=lambdax,i:x[2:8]+(rec(x[8:],i-1)ifi>1else'')ifxelse''

fun=lambdax,i:x[i+1:8]+rec(x[8:],i-1)

fun() 接受 2 个参数，第一个参数为表示一个字符的二进制字符串，这个二进制字符串可能有不同的长度(8、16、24…48)；第二个参数为这个字符占多少个字节。

lambda x, i: x[i+1:8] + rec(x[8:], i-1) 中 x[i+1:8] 获得第一个字节的数据，然后调用 rec()，以递归的方提取后面字节中的数据。

这里要提一句，rec = lambda x, i: x[2:8] + (rec(x[8:], i-1) if i > 1 else '') if x else ''，你可能对在表达式里面引用了 rec 感到不可理解，的确，严格意义上这样是不能实现递归的，但在 python 里这样是可以的，这就是 python 的语法糖了。

使用 lambda 表达式写递归从来不是一件简单的事，因为匿名函数引用自身并不简单，大家可以参考一下大牛刘未鹏的博文：http://blog.csdn.net/pongba/article/details/1336028

我们注意到，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：

Python

chartype = binary[index:].index('0') # 存放字符所占字节数，一个字节的字符会存为 0

1

chartype=binary[index:].index('0')# 存放字符所占字节数，一个字节的字符会存为 0

string.append(chr(int(fun(binary[index:index+length],chartype),2))) 这一行中用到的函数 int() 以及 chr 的作用如下：

chr()：接受一个参数，参数为 int 值，返回 Unicode 码点为这个 int 值的字符。见下图：

while 循环的最后我们将当前字符的索引增加当前字符的长度，得到下一个字符的索引。

这样，我们可以识别出二进制字符串中哪些部分代表哪些字符了，然后就能调用 fun() 取得各个字符的数据了。

参数解析

最后，我们通过 docopt 模块对命令行输入参数进行解析，通过对各个参数的输入情况进行判断分析功能的调用：

Python

```
if __name__ == '__main__':
"""command-line interface"""
arguments = docopt(__doc__)
# print(arguments)
if arguments['-e'] or arguments['encode']:
if arguments[''] is None:
arguments[''] = "待加密的文本"
if arguments[''] is None:
arguments[''] = "encodedImage.png"
if isTextFile(arguments['']):
with open(arguments[''], 'rt') as f:
arguments[''] = f.read()
print("载体图片:")
print(arguments['']+"\n")
print("待加密密文:")
print(arguments['']+"\n")
print("加密后图片:")
print(arguments['']+"\n")
print("加密中……\n")
encodeDataInImage(Image.open(arguments['']), arguments['']).save(arguments[''])
print("加密完成,密文为：\n"+arguments['']+"\n")
elif arguments['-d'] or arguments['decode']:
```

```
print("解密中……\n")
print("揭秘完成，密文为：\n"+decodeImage(Image.open(arguments['']))+"\n")
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
if__name__=='__main__':
"""command-line interface"""
```

```
arguments=docopt(__doc__)
#    print(arguments)
ifarguments['-e']orarguments['encode']:
ifarguments['']isNone:
arguments['']="待加密的文本"
ifarguments['']isNone:
arguments['']="encodedImage.png"
ifisTextFile(arguments['']):
withopen(arguments[''],'rt')asf:
arguments['']=f.read()
print("载体图片:")
print(arguments['']+"\n")
print("待加密密文:")
print(arguments['']+"\n")
print("加密后图片:")
print(arguments['']+"\n")
print("加密中……\n")
encodeDataInImage(Image.open(arguments['']),arguments['']).save(arguments[''])
print("加密完成,密文为：\n"+arguments['']+"\n")
elifarguments['-d']orarguments['decode']:
print("解密中……\n")
print("揭秘完成，密文为：\n"+decodeImage(Image.open(arguments['']))+"\n")
```

效果测试

输入以下命令分别对图片加密和解密：

Shell

```
# 将文本信息隐写入 coffee.png 图片
$ ./imageEncoding -e coffee.png 新玛特真是太帅了
# 从图片中解析出文本信息
$ ./imageEncoding -d encodedImage.png
```

1

2

3

4

5

将文本信息隐写入 coffee.png 图片

$./imageEncoding-ecoffee.png新玛特真是太帅了

从图片中解析出文本信息

$./imageEncoding-dencodedImage.png

如下图，演示程序成功从隐写图片中提取出了加密文本

完整代码

下面贴出完整的实现代码，包括以 png 文件为载体的加密解密：

Python

#!/usr/bin/env python3

#coding=utf-8

"""Encode png image via command-line.

Usage:

imageEncoding (-e|encode) [] []

imageEncoding (-d|decode)

Options:

-h,--help 显示帮助菜单

-e 加密

-d 解密

Example:

imageEncoding -e coffee.png hello textOrFileToEncode encodedImage.png

imageEncoding -d encodedImage.png

"""

from PIL import Image

from docopt import docopt

import crash_on_ipy

"""

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 RGBAmakeImageEven(image):

pixels = list(image.getdata()) # 得到一个这样的列表： [(r,g,b,t),(r,g,b,t)...]

evenPixels = [(r>>1<<1,g>>1<<1,b>>1<<1,t>>1<<1) for [r,g,b,t] in pixels] # 更改所有值为偶数(魔法般的移位)

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

def RGBmakeImageEven(image):

pixels = list(image.getdata()) # 得到一个这样的列表： [(r,g,b,t),(r,g,b,t)...]

evenPixels = [(r>>1<<1,g>>1<<1,b>>1<<1) for [r,g,b] in pixels] # 更改所有值为偶数(魔法般的移位)

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

"""

内置函数 bin() 的替代，返回固定长度的二进制字符串

"""

def constLenBin(int):

binary = "0"*(8-(len(bin(int))-2))+bin(int).replace('0b','') # 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

return binary

"""

将字符串编码到图片中

"""

def RGBAencodeDataInImage(image, data):

evenImage = RGBAmakeImageEven(image) # 获得最低有效位为 0 的图片副本

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

if len(binary) > len(image.getdata()) * 4: # 如果不可能编码全部数据， 抛出异常

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

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()))] # 将 binary 中的二进制字符串信息编码进像素里

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

def RGBencodeDataInImage(image, data):

evenImage = RGBmakeImageEven(image) # 获得最低有效位为 0 的图片副本

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

if len(binary)%3 != 0: # 将转换的比特流数据末位补零，使其长度为3的倍数，防止其在下面重新编码的过程中发生越界

rema = len(binary)%3

binary = binary+('0'*(3-rema))

# print(len(binary))

if len(binary) > len(image.getdata()) * 3: # 如果不可能编码全部数据， 抛出异常

raise Exception("Error: Can't encode more than " + len(evenImage.getdata()) * 3 + " bits in this image. ")

# evenImageList = list(evenImage.getdata())

# for index in range(len(evenImageList)):

# if index*3 < len(binary):

# tup = evenImageList[index]

# r = tup[0]

# g = tup[1]

# b = tup[2]

# r += int(binary[index*3+0])

# evenImageList[index] = (r,g,b)

# else:

# break

# if index*3+1 < len(binary):

# tup = evenImageList[index]

# r = tup[0]

# g = tup[1]

# b = tup[2]

# g += int(binary[index*3+1])

# evenImageList[index] = (r,g,b)

# else:
```

```
# break

# if index*3+2 < len(binary):

# tup = evenImageList[index]

# r = tup[0]

# g = tup[1]

# b = tup[2]

# b += int(binary[index*3+2])

# evenImageList[index] = (r,g,b)

# else:

# break

# index += 1

# encodedPixels = evenImageList

encodedPixels = [(r+int(binary[index*3+0]),g+int(binary[index*3+1]),b+int(binary[index*3+2])) if index*3 < len(binary) else (r,g,b) for index, (r,g,b) in enumerate(list(evenImage.getdata()))] # 将 binary 中的二进制字符串信息编码进像素里

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

"""

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') # 存放字符所占字节数，一个字节的字符会存为 0

length = chartype*8 if chartype else 8

string.append(chr(int(fun(binary[index:index+length],chartype),2)))

index += length
```

```
return ''.join(string)

"""

解码隐藏数据

"""

def RGBAdecodeImage(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

def RGBdecodeImage(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)) for (r,g,b) 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

def isTextFile(path):

if path.endswith(".txt"):

return True

elif path.endswith(".m"):

return True

elif path.endswith(".h"):

return True

elif path.endswith(".c"):

return True
```

```
elif path.endswith(".py"):
return True
else:
return False
if __name__ == '__main__':
"""command-line interface"""
arguments = docopt(__doc__)
# print(arguments)
if arguments['-e'] or arguments['encode']:
if arguments[''] is None:
arguments[''] = "待加密的文本"
if arguments[''] is None:
arguments[''] = "encodedImage.png"
if isTextFile(arguments['']):
with open(arguments[''], 'rt') as f:
arguments[''] = f.read()
print("载体图片:")
print(arguments['']+"\n")
print("待加密密文:")
print(arguments['']+"\n")
print("加密后图片:")
print(arguments['']+"\n")
print("加密中……\n")
im = Image.open(arguments[''])
if im.mode == 'RGBA':
RGBAencodeDataInImage(im, arguments['']).save(arguments[''])
# elif im.mode == 'RGB':
# RGBencodeDataInImage(im, arguments['']).save(arguments[''])
else:
print("暂不支持此图片格式……")
print("加密完成,密文为：\n"+arguments['']+"\n")
```

```
elif arguments['-d'] or arguments['decode']:
print("解密中……\n")
im = Image.open(arguments[''])
if im.mode == 'RGBA':
print("解秘完成，密文为：\n"+RGBAdecodeImage(im)+"\n")
# elif im.mode == 'RGB':
# print("解秘完成，密文为：\n"+RGBdecodeImage(im)+"\n")
else:
print("非法的图片格式……")
```

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

```
#!/usr/bin/env python3
#coding=utf-8
"""Encode png image via command-line.
Usage:
imageEncoding (-e|encode) [] []
imageEncoding (-d|decode)
Options:
-h,--help   显示帮助菜单
```

-e 加密

-d 解密

Example:

imageEncoding -e coffee.png hello textOrFileToEncode encodedImage.png

imageEncoding -d encodedImage.png

"""

fromPILimportImage

fromdocoptimportdocopt

importcrash_on_ipy

"""

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)

"""

defRGBAmakeImageEven(image):

pixels=list(image.getdata())# 得到一个这样的列表： [(r,g,b,t),(r,g,b,t)...]

evenPixels=[(r>>1<<1,g>>1<<1,b>>1<<1,t>>1<<1)for[r,g,b,t]inpixels]# 更改所有值为偶数(魔法般的移位)

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

returnevenImage

defRGBmakeImageEven(image):

pixels=list(image.getdata())# 得到一个这样的列表： [(r,g,b,t),(r,g,b,t)...]

evenPixels=[(r>>1<<1,g>>1<<1,b>>1<<1)for[r,g,b]inpixels]# 更改所有值为偶数(魔法般的移位)

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

returnevenImage

"""

内置函数 bin() 的替代，返回固定长度的二进制字符串

"""

defconstLenBin(int):

binary="0"*(8-(len(bin(int))-2))+bin(int).replace('0b','')# 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

returnbinary

"""

将字符串编码到图片中

"""

defRGBAencodeDataInImage(image,data):

evenImage=RGBAmakeImageEven(image)# 获得最低有效位为 0 的图片副本

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

iflen(binary)>len(image.getdata())*4:# 如果不可能编码全部数据， 抛出异常

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

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]))ifindex*4

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

returnencodedImage

defRGBencodeDataInImage(image,data):

evenImage=RGBmakeImageEven(image)# 获得最低有效位为 0 的图片副本

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

iflen(binary)%3!=0:# 将转换的比特流数据末位补零，使其长度为3的倍数，防止其在下面重新编码的过程中发生越界

rema=len(binary)%3

binary=binary+('0'*(3-rema))

print(len(binary))

iflen(binary)>len(image.getdata())*3:# 如果不可能编码全部数据， 抛出异常

raiseException("Error: Can't encode more than "+len(evenImage.getdata())*3+" bits in this image. ")

evenImageList = list(evenImage.getdata())

for index in range(len(evenImageList)):

if index*3 < len(binary):

tup = evenImageList[index]

r = tup[0]

g = tup[1]

b = tup[2]

r += int(binary[index*3+0])

evenImageList[index] = (r,g,b)

```
#     else:
#         break
#     if index*3+1 < len(binary):
#         tup = evenImageList[index]
#         r = tup[0]
#         g = tup[1]
#         b = tup[2]
#         g += int(binary[index*3+1])
#         evenImageList[index] = (r,g,b)
#     else:
#         break
#     if index*3+2 < len(binary):
#         tup = evenImageList[index]
#         r = tup[0]
#         g = tup[1]
#         b = tup[2]
#         b += int(binary[index*3+2])
#         evenImageList[index] = (r,g,b)
#     else:
#         break
#     index += 1
# encodedPixels = evenImageList
encodedPixels=[(r+int(binary[index*3+0]),g+int(binary[index*3+1]),b+int(binary[index*3+2]))ifindex*3
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
returnencodedImage
"""
从二进制字符串转为 UTF-8 字符串
"""
defbinaryToString(binary):
index=0
```

```
string=[]

rec=lambdax,i:x[2:8]+(rec(x[8:],i-1)ifi>1else'')ifxelse''

# rec = lambda x, i: x and (x[2:8] + (i > 1 and rec(x[8:], i-1) or '')) or ''

fun=lambdax,i:x[i+1:8]+rec(x[8:],i-1)

whileindex+1

chartype=binary[index:].index('0')# 存放字符所占字节数，一个字节的字符会存为 0

length=chartype*8ifchartypeelse8

string.append(chr(int(fun(binary[index:index+length],chartype),2)))

index+=length

return''.join(string)

'''''

解码隐藏数据

'''''

defRGBAdecodeImage(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)inp
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))iflocationDoubleNull%8!=0elselocationDoubleNull

data=binaryToString(binary[0:endIndex])

returndata

defRGBdecodeImage(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))for(r,g,b)inpixels])# 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))iflocationDoubleNull%8!=0elselocationDoubleNull

data=binaryToString(binary[0:endIndex])

returndata
```

```
defisTextFile(path):
ifpath.endswith(".txt"):
returnTrue
elifpath.endswith(".m"):
returnTrue
elifpath.endswith(".h"):
returnTrue
elifpath.endswith(".c"):
returnTrue
elifpath.endswith(".py"):
returnTrue
else:
returnFalse
if__name__=='__main__':
"""command-line interface"""
arguments=docopt(__doc__)
#    print(arguments)
ifarguments['-e']orarguments['encode']:
ifarguments['']isNone:
arguments['']="待加密的文本"
ifarguments['']isNone:
arguments['']="encodedImage.png"
ifisTextFile(arguments['']):
withopen(arguments[''],'rt')asf:
arguments['']=f.read()
print("载体图片:")
print(arguments['']+"\n")
print("待加密密文:")
print(arguments['']+"\n")
print("加密后图片:")
print(arguments['']+"\n")
```

```
print("加密中……\n")
im=Image.open(arguments[''])
ifim.mode=='RGBA':
RGBAencodeDataInImage(im,arguments['']).save(arguments[''])
# elif im.mode == 'RGB':
#     RGBencodeDataInImage(im, arguments['']).save(arguments[''])
else:
print("暂不支持此图片格式……")
print("加密完成,密文为：\n"+arguments['']+"\n")
elifarguments['-d']orarguments['decode']:
print("解密中……\n")
im=Image.open(arguments[''])
ifim.mode=='RGBA':
print("解秘完成，密文为：\n"+RGBAdecodeImage(im)+"\n")
# elif im.mode == 'RGB':
#     print("解秘完成，密文为：\n"+RGBdecodeImage(im)+"\n")
else:
print("非法的图片格式……")
```