在图像中隐藏数据：用 Python 来实现图像隐写术

原创

Python新世界 于 2020-07-17 13:27:35 发布 1857 收藏 16
文章标签： python 编程语言
GokuCode
本文链接：https://blog.csdn.net/weixin_46089319/article/details/107406183
版权

什么是“隐写术”?

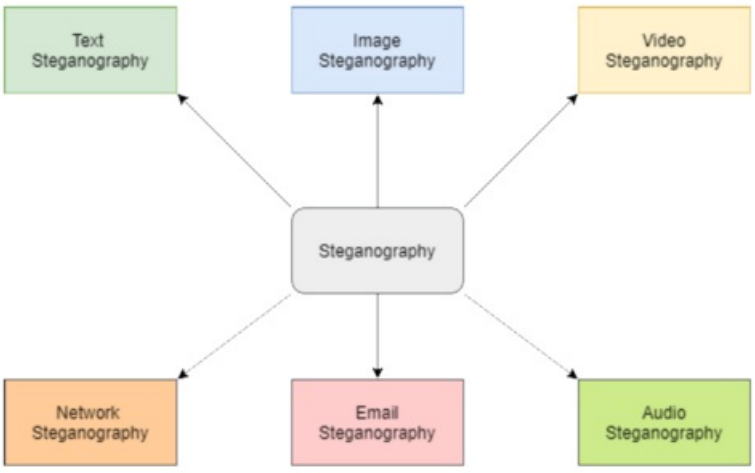
隐写术是将机密信息隐藏在更大的信息中，使别人无法知道隐藏信息的存在以及隐藏信息内容的过程。隐写术的目的是保证双方之间的机密交流。与隐藏机密信息内容的密码学不同，隐写术隐瞒了传达消息的事实。尽管隐写术与密码学有所不同，但是两者之间有许多类似，并且一些作者会将隐写术归类为一种密码学形式，因为隐秘通信也是一种机密消息。

使用隐写术比使用密码学加密有什么优势？

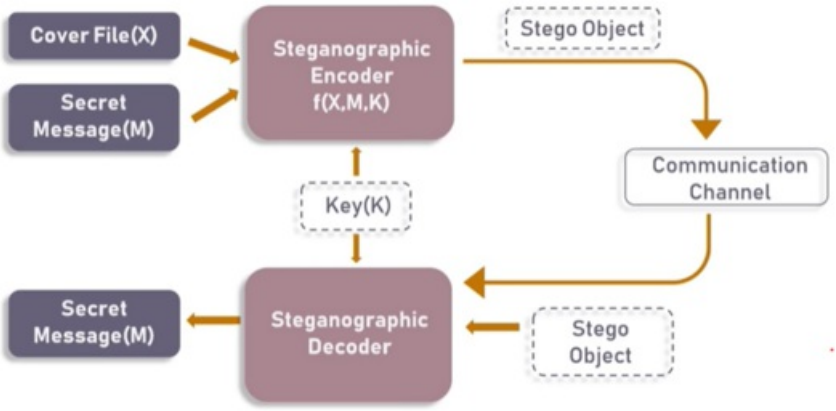
迄今为止，密码学一直是作用于保护发送者与接收者之间的保密性。然而，现在除了密码学之外，隐写术也越来越多地用于为需要被隐藏的数据添加更多保护层。使用隐写术比单独使用 密码学 的优势在于，有意加密的消息不会作为被监视的对象而引起注意。明显可见的加密消息，无论其多么难以解破，都会引起人们的注意。并且在 加密 是非法行为的国家中，这本身可能就是在犯罪。

隐写术的分类

隐写术目前已经可以在图像、视频、文本或音频等多种传输媒介上进行。



基本的隐写术的分类模型



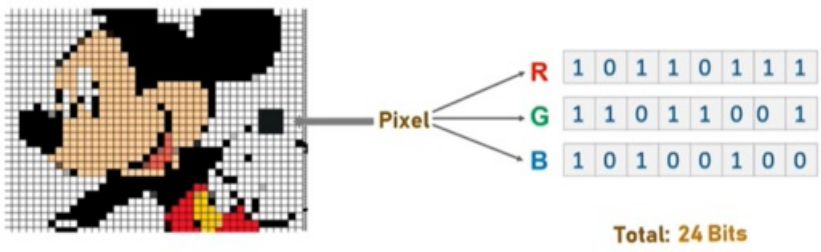
如上图所示，原始图像文件（X）和机密消息（M）都作为入参传入到隐写术编码器中。隐写术编码器函数 f(X,M,K) 通过使用最低有效位编码等技术将机密消息写入到封面图像文件中。最后生成的隐写术图像看起来与封面图像文件非常相似，肉眼难辨。这样就完成了编码。若要取出机密消息，将之前生成的隐写术图像输入隐写术解码器即可。

本文将使用 Python 来实现图像隐写术。手把手教您使用 Python 语言，通过一种叫“ 最低有效位（Least Significant Bit，LSB） ”的技术来隐藏文本消息。

最低有效位隐写术

我们可以将 **数字图像** 描述为一组有限的数字值，称为像素。像素是图像中最小的不可分割单位，其值表示在任何特定点上给定颜色的亮度。因此，我们可以将图像想象为像素的矩阵（或二维数组），其中包含固定数量的行和列。

最低有效位（LSB）是一种将每个像素的最后一位修改并用机密消息的数据位代替的技术。



从上图可以清楚地看出，如果我们修改最高有效位（MSB），它将对最终值产生更大的影响，但是如果我们修改最低有效位（LSB），则对最终值的影响将是最小的，因此，我们使用最低有效位隐写术。

最低有效位是如何工作的？

每个像素包含三个值，红、绿、蓝，这些值的范围从 0 到 255，换句话说，它们是一个 8 位二进制数。让我们举一个例子来说明它是如何工作的，假设您想要将消息 “ **hi** ” 隐藏到一个 **4x4** 的图像中，该图像具有以下像素值：

[(225, 12, 99), (155, 2, 50), (99, 51, 15), (15, 55, 22),(155, 61, 87), (63, 30, 17), (1, 55, 19), (99, 81, 66),(219, 77, 91), (69, 39, 50), (18, 200, 33), (25, 54, 190)]

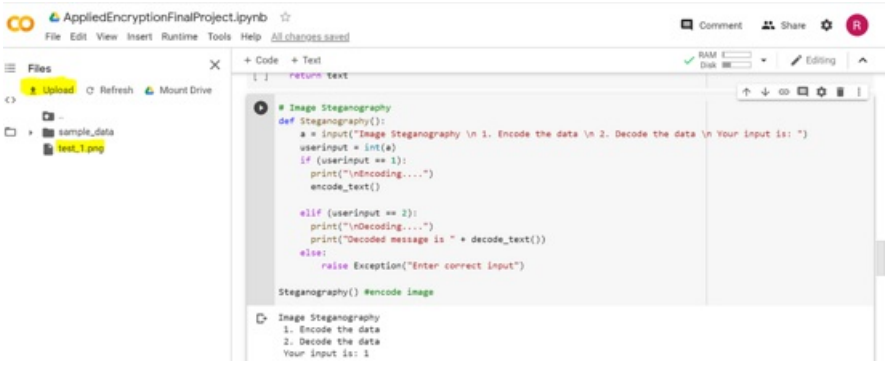
使用 ASCII 表 ，我们可以先将机密消息转换为十进制值，然后再转换为二进制： **0110100 0110101** 。现在，我们对像素值逐一进行迭代，在将它们转换为二进制后，我们将每个最小有效位依次替换为该信息位。（例如 225 是 11100001，我们替换最后一位，最右边的（1）和机密消息的第一位（0），依次类推)。这样的操作只会对像素值进行 +1 或 -1 的修改，因此肉眼根本看不出来。执行最低有效位隐写术后得到的像素值如下所示：

[(224, 13, 99),(154, 3, 50),(98, 50, 15),(15, 54, 23),(154, 61, 87),(63, 30, 17),(1, 55, 19),(99, 81, 66),(219, 77, 91),(69, 39, 50),(18, 200, 33),(25, 54, 190)]

使用 Python 在图像中隐藏文本

在本节中，我们将使用 Python 代码逐步了解隐藏文本和显示文本的过程。首先，打开 google collab notebook ，按照下面的步骤操作：

在开始编写代码之前，可以使用左侧菜单栏中的 upload 选项上传要用于隐写的图像（png 文件）。



第一步：导入所有必需的 Python 库。

```
#import all the required libraries

import cv2
import numpy as np
import types
from google.colab.patches import cv2_imshow #Google colab crashes if you try to display
#image using cv2.imshow() thus use this import
```

第二步：定义一个可以将任何类型的数据转换为二进制数据的函数，我们将在编码和解码阶段使用这个函数来将机密消息数据和像素值转换为二进制。

```
def messageToBinary(message):
  if type(message) == str:
    return ''.join([ format(ord(i), "08b") for i in message ])
  elif type(message) == bytes or type(message) == np.ndarray:
    return [ format(i, "08b") for i in message ]
  elif type(message) == int or type(message) == np.uint8:
    return format(message, "08b")
  else:
    raise TypeError("Input type not supported")
```

第三步：编写一个函数，通过改变最低有效位将机密消息隐藏到图像中。

```
[ ]  # Function to hide the secret message into the image

     def hideData(image, secret_message):

       # calculate the maximum bytes to encode
       n_bytes = image.shape[0] * image.shape[1] * 3 // 8
       print("Maximum bytes to encode:", n_bytes)

       #Check if the number of bytes to encode is less than the maximum bytes in the image
       if len(secret_message) > n_bytes:
           raise ValueError("Error encountered insufficient bytes, need bigger image or less data !!")

       secret_message += "#####" # you can use any string as the delimeter

       data_index = 0
       # convert input data to binary format using messageToBinary() fucntion
       binary_secret_msg = messageToBinary(secret_message)

       data_len = len(binary_secret_msg) #Find the length of data that needs to be hidden
       for values in image:
```

```
for values in image:
    for pixel in values:
        # convert RGB values to binary format
        r, g, b = messageToBinary(pixel)
        # modify the least significant bit only if there is still data to store
        if data_index < data_len:
            # hide the data into least significant bit of red pixel
            pixel[0] = int(r[:-1] + binary_secret_msg[data_index], 2)
            data_index += 1
        if data_index < data_len:
            # hide the data into least significant bit of green pixel
            pixel[1] = int(g[:-1] + binary_secret_msg[data_index], 2)
            data_index += 1
        if data_index < data_len:
            # hide the data into least significant bit of  blue pixel
            pixel[2] = int(b[:-1] + binary_secret_msg[data_index], 2)
            data_index += 1
        # if data is encoded, just break out of the loop
        if data_index >= data_len:
            break

return image
```

第四步：定义一个函数，用于从隐藏后的图像中解码隐藏信息。

```
[ ]  def showData(image):

       binary_data = ""
       for values in image:
           for pixel in values:
               r, g, b = messageToBinary(pixel) #convert the red,green and blue values into binary format
               binary_data += r[-1] #extracting data from the least significant bit of red pixel
               binary_data += g[-1] #extracting data from the least significant bit of red pixel
               binary_data += b[-1] #extracting data from the least significant bit of red pixel
       # split by 8-bits
       all_bytes = [ binary_data[i: i+8] for i in range(0, len(binary_data), 8) ]
       # convert from bits to characters
       decoded_data = ""
       for byte in all_bytes:
           decoded_data += chr(int(byte, 2))
           if decoded_data[-5:] == "#####": #check if we have reached the delimeter which is "#####"
               break
       #print(decoded_data)
       return decoded_data[:-5] #remove the delimeter to show the original hidden message
```

第五步：定义将输入的图像名称和机密消息作为用户的输入的函数。

```
[ ]  # Encode data into image
     def encode_text():
       image_name = input("Enter image name(with extension): ")
       image = cv2.imread(image_name) # Read the input image using OpenCV-Python.
       #It is a library of Python bindings designed to solve computer vision problems.

       #details of the image
       print("The shape of the image is: ",image.shape) #check the shape of image to calculate the number of bytes in it
       print("The original image is as shown below: ")
       resized_image = cv2.resize(image, (500, 500)) #resize the image as per your requirement
       cv2_imshow(resized_image) #display the image


       data = input("Enter data to be encoded : ")
       if (len(data) == 0):
         raise ValueError('Data is empty')

       filename = input("Enter the name of new encoded image(with extension): ")
       encoded_image = hideData(image, data) # call the hideData function to hide the secret message into the selected image
       cv2.imwrite(filename, encoded_image)
```

第六步：创建一个函数，要求用户输入需要解码的图像的名称，然后调用 showData() 函数以返回解码后的消息。

```
# Decode the data in the image
def decode_text():
  # read the image that contains the hidden image
  image_name = input("Enter the name of the steganographed image that you want to decode (with extension) :")
  image = cv2.imread(image_name) #read the image using cv2.imread()

  print("The Steganographed image is as shown below: ")
  resized_image = cv2.resize(image, (500, 500))  #resize the original image as per your requirement
  cv2_imshow(resized_image) #display the Steganographed image

  text = showData(image)
  return text
```

第七步： 主函数

```
# Image Steganography
def Steganography():
    a = input("Image Steganography \n 1. Encode the data \n 2. Decode the data \n Your input is: ")
    userinput = int(a)
    if (userinput == 1):
      print("\nEncoding....")
      encode_text()

    elif (userinput == 2):
      print("\nDecoding....")
      print("Decoded message is " + decode_text())
    else:
        raise Exception("Enter correct input")

Steganography() #encode image
```

输出/结果：

加密消息：

```
Image Steganography
 1. Encode the data
 2. Decode the data
 Your input is: 1

Encoding....
Enter image name(with extension): test_1.png
The shape of the image is:  (1254, 1254, 3)
The original image is as shown below:
```

```
Enter data to be encoded : hakunamatata
Enter the name of new encoded image(with extension): test_1_encoded.png
Maximum bytes to encode: 589693
```

解码消息：

```
Image Steganography
  1. Encode the data
  2. Decode the data
  Your input is: 2

Decoding....
Enter the name of the steganographed image that you want to decode (with extension) :test_1_encoded.png
The Steganographed image is as shown below:
```

```
Decoded message is hakunamatata
```

如果您对代码感兴趣，可以在 Github 上查看我的 jupyter notebook 代码。