Python-LSB隐写算法

原创

小狐狸FM 于 2021-09-13 20:36:26 发布 1645 收藏 27

分类专栏： Python 安全 # CTF夺旗 文章标签： python 算法

本文链接：https://blog.csdn.net/smallfox233/article/details/120214050

Python 同时被 3 个专栏收录

88 篇文章 5 订阅

订阅专栏

安全

91 篇文章 9 订阅

订阅专栏

CTF夺旗

38 篇文章 0 订阅

订阅专栏

文章目录

前言

- 使用 `PIL` 库的时候，使用命令 `pip install pillow` 进行下载即可
- 对于文中的 `LSB` 随机隐写来说，是没有办法通过 `Stegsolve` 解出隐藏的内容，必须得知道随机数种子才能解密
- 在读写的时候发现数据的最后一个字符会出错，可能是写入的问题
  也可能是读取出了问题
- 本来是想随机 `RGB` 通道的顺序，后来发现写起来循环和 `if` 语句太多了
  所以就改成了每个像素点随机取其中的一个通道存数据
- 还有一个比较有用的思路我没去实现，你可以在存入数据的时候多添加几个字符作为结束的标志，
  在读取的时候遇到结束标志就结束读取节省运行时间。
- 声明一下第四个步骤的图片对比不是我自己写的，忘记是哪篇文章爬来的，原作者可评论来认领一下
  本来是想自己写 `LstBit` 和 `StringtoBin` 函数的，之后发现大佬的函数简洁明了就直接拿过来用了
- 读取像素点的RGB通道： `im.getpixel((w,h))`
  写入像素点的RGB通道： `im.putpixel((w,h),(R_bit,G_bit,B_bit))`
  保存图片： `im.save(new_path)`

LSB顺序+随机隐写和提取(matlab)

关于图片LSB隐写

pil库是python的内置标准库吗_python安装pil库方法及代码

lec2-LSB隐写术

Lsb图片隐写

一、LSB隐写原理

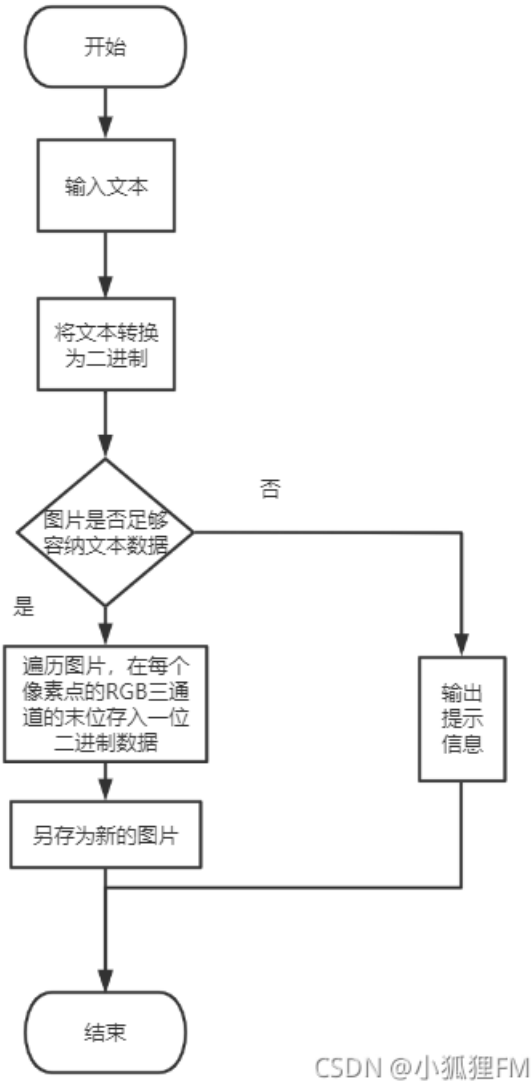
`LSB` 的全称为 `Least Significant Bit` ，在二进制数中意为最低有效位， `LSB` 位于二进制数的最右侧。 `MSB` 的全称为 `Most Significant Bit` ，在二进制数中属于最高有效位， `MSB` 是最高加权位，若 `MSB=1` 则表示数据为负值，若 `MSB=0` 则表示数据为正。

由于人类的眼睛对于颜色的辨别能力有限，稍微变更图像的某一位数据时候是没法直接看出来前后修改差别的，所以我们可以通过修改最低有效位 `LSB` 来达到隐藏信息的目的， `LSB` 也称作空域图像水印技术。

二、LSB顺序隐写

[1]. 加密流程

> 每张图片存在多个像素点，每个像素点有三个通道 `RGB`，
> 取 `RGB` 三个通道的二进制末尾，将文本数据转化为二进制按顺序存入对应的通道末位。



[2]. 解密流程

> 解密的时候和加密一样，遍历所有的像素点 `RGB` 通道，
> 取出对应通道的末位二进制，最后以每 `8` 位二进制转换成字符输出结果。

[3]. 代码

```
# 小狐狸FM
from PIL import Image
import re


def LstBit(source, target):
    '''替换最后一位的数据'''
    replace_reg = re.compile(r'[1|0]$')  # 匹配二进制最后一位
    return replace_reg.sub(target, source)


def StringtoBin(string):
    '''将字符串转换成二进制,不够8位补齐8位'''
    return ''.join(bin(ord(c)).replace('0b', '').rjust(8, '0') for c in string)


def encode(path, new_path, data):
    '''
    LSB加密
    path: 原图路径
    new_path: 为加密后图片的路径
    data: 为需要加密的数据
    '''
    # 初始化、定义
    data = StringtoBin(data)
    im = Image.open(path)
    width = im.size[0]  # 宽度
    height = im.size[1]  # 高度
```

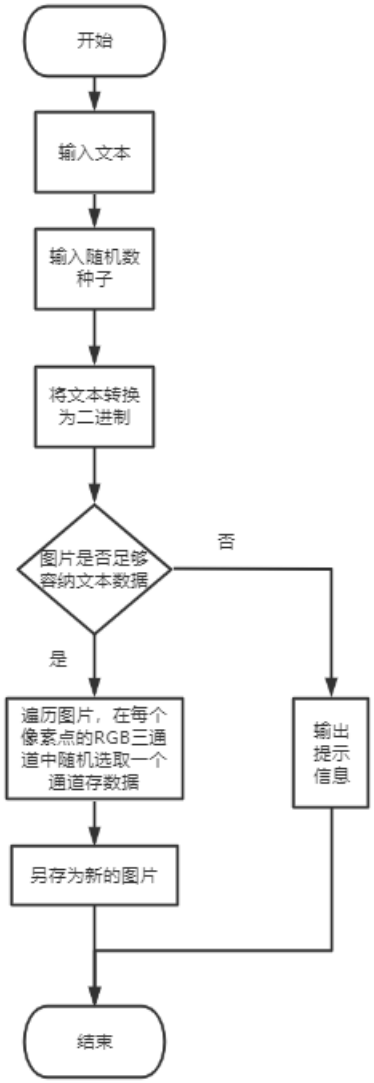
```
    height = im.size[1]  # 高度
    count = 0
    if width * height * 3 < len(data):
        print("数据过大")
        return
    # 遍历图片所有像素点
    for w in range(width):
        for h in range(height):
            # 获取像素点
            pixel = im.getpixel((w, h))
            # 取三通道值（十进制）
            R = pixel[0]
            G = pixel[1]
            B = pixel[2]
            # 十进制转二进制
            R_bit = bin(R).replace("0b", "")
            G_bit = bin(G).replace("0b", "")
            B_bit = bin(B).replace("0b", "")
            # 在R通道的末位存数据
            R_bit = LstBit(R_bit, data[count])
            count += 1
            if count == len(data):  # 存完数据时
                break
            # 在G通道的末位存数据
            G_bit = LstBit(G_bit, data[count])
            count += 1
            if count == len(data):  # 存完数据时
                break
            # 在B通道的末位存数据
            B_bit = LstBit(R_bit, data[count])
            count += 1
            if count == len(data):  # 存完数据时
                break
            # 二进制转十进制
            R_bit = int(R_bit, 2)
            G_bit = int(G_bit, 2)
            B_bit = int(B_bit, 2)
            # 写回像素点
            im.putpixel((w, h), (R_bit, G_bit, B_bit))
        if count == len(data):  # 存完数据时
            break
    im.save(new_path)


def decode(path):
    '''
    LSB解密
    path:需要解密的图片路径
    '''
    data_bit = ""
    im = Image.open(path)
    width = im.size[0]  # 宽度
    height = im.size[1]  # 高度
    count = 0
    # 遍历图片所有像素点
    for w in range(width):
        for h in range(height):
            # 获取像素点
            pixel = im.getpixel((w, h))
            # 取三通道值（十进制）
```

```
            R = pixel[0]
            G = pixel[1]
            B = pixel[2]
            # 十进制转二进制
            R_bit = bin(R).replace("0b", "")
            G_bit = bin(G).replace("0b", "")
            B_bit = bin(B).replace("0b", "")
            # 在RGB通道的末位取数据
            data_bit += R_bit[-1:]
            data_bit += G_bit[-1:]
            data_bit += B_bit[-1:]
    # 二进制转字符
    data = ""
    i = 0
    try: #有时候会出现报错，一报错就输出结果
        while i <= len(data_bit):  # 步长为8
            data += chr(int(data_bit[i:i + 8],2))
            i += 8
    except:
        return data
    return data
if __name__ == "__main__":
    encode("data/new.png","data/new1.png","hello world")
    print(decode("data/new1.png"))
```

三、LSB随机隐写

[1]. 加密流程

> 先确定一个随机数种子，根据该随机数种子生成对应的伪随机数，
> 对于每个像素点，通过产生的伪随机数在 `RGB` 三个通道中选取一个通道，在该通道的末位存储数据



[2]. 解密流程

> 使用加密过程一样的随机数种子，同样地遍历图片的每个像素点，
> 根据随机数选择对应的通道取出该通道的末尾二进制，最后以每 8 位二进制转换成字符输出结果。

[3]. 代码

```
# 小狐狸FM
from PIL import Image
import re
import random

def LstBit(source,target):
    '''替换最后一位的数据'''
    replace_reg = re.compile(r'[1|0]$') #匹配二进制最后一位
    return replace_reg.sub(target,source)

def StringtoBin(string):
    '''将字符串转换成二进制,不够8位补齐8位'''
    return ''.join(bin(ord(c)).replace('0b','').rjust(8,'0') for c in string)

def encode(path,new_path,data):
    '''
    LSB加密
    path: 原图路径
    new_path: 为加密后图片的路径
    data: 为需要加密的数据
    '''
    #初始化、定义
    data = StringtoBin(data)
    im = Image.open(path)
    width = im.size[0]  #宽度
    height = im.size[1] #高度
```

```
    height = im.size[1] #高度
    count = 0
    if width*height<len(data):
        print("数据过大")
        return
    #遍历图片所有像素点
    for w in range(width):
        for h in range(height):
            #获取像素点
            pixel = im.getpixel((w,h))
            #取三通道值（十进制）
            R = pixel[0]
            G = pixel[1]
            B = pixel[2]
            #十进制转二进制
            R_bit = bin(R).replace("0b","")
            G_bit = bin(G).replace("0b","")
            B_bit = bin(B).replace("0b","")
            #随机选取一个通道存末位二进制
            choose = random.randint(0,2)
            if choose==0:   #在R通道的末位存数据
                R_bit = LstBit(R_bit,data[count])
                count += 1
                if count==len(data): #存完数据时
                    break
            elif choose==1: #在G通道的末位存数据
                G_bit = LstBit(G_bit,data[count])
                count += 1
                if count==len(data): #存完数据时
                    break
            elif choose==2: #在B通道的末位存数据
                B_bit = LstBit(R_bit,data[count])
                count += 1
                if count==len(data): #存完数据时
                    break
            #二进制转十进制
            R_bit = int(R_bit,2)
            G_bit = int(G_bit,2)
            B_bit = int(B_bit,2)
            #写回像素点
            im.putpixel((w,h),(R_bit,G_bit,B_bit))
        if count==len(data): #存完数据时
            break
    im.save(new_path)

def decode(path):
    '''
    LSB解密，随机数
    path:需要解密的图片路径
    '''
    data_bit = ""
    im = Image.open(path)
    width = im.size[0]  #宽度
    height = im.size[1] #高度
    count = 0
    #遍历图片所有像素点
    for w in range(width):
        for h in range(height):
            #获取像素点
            pixel = im.getpixel((w,h))
```

```
            #取三通道值（十进制）
            R = pixel[0]
            G = pixel[1]
            B = pixel[2]
            #十进制转二进制
            R_bit = bin(R).replace("0b","")
            G_bit = bin(G).replace("0b","")
            B_bit = bin(B).replace("0b","")
            #随机选取一个通道读取末位二进制
            choose = random.randint(0,2)
            if choose==0:   #在R通道的末尾读取数据
                data_bit += R_bit[-1:]
            elif choose==1: #在G通道的末尾读取数据
                data_bit += G_bit[-1:]
            elif choose==2: #在B通道的末尾读取数据
                data_bit += B_bit[-1:]
    #二进制转字符
    try: #有时候会出现报错，一报错就输出结果
        while i <= len(data_bit):  # 步长为8
            data += chr(int(data_bit[i:i + 8],2))
            i += 8
    except:
        return data
    return data

if __name__=='__main__':
    random.seed(17)
##    encode("pic.png","new.png","smallfox2345")
    print(decode("new.png"))
```

四、图片对比

```
from PIL import Image
import math
import operator
from functools import reduce


def image_contrast(img1, img2):

    image1 = Image.open(img1)
    image2 = Image.open(img2)

    h1 = image1.histogram()
    h2 = image2.histogram()

    result = math.sqrt(reduce(operator.add,  list(map(lambda a,b: (a-b)**2, h1, h2)))/len(h1) )
    return result

if __name__ == '__main__':
    img1 = "pic.png"  # 指定图片路径
    img2 = "new.png"
    result = image_contrast(img1,img2)
    print(result)
```