学习LSB隐写总结

原创

YYGP 于 2020-12-09 21:56:10 发布 940 收藏 10

分类专栏： LSB隐写

本文链接：https://blog.csdn.net/weixin_46727990/article/details/110941347

LSB最低有效位隐写

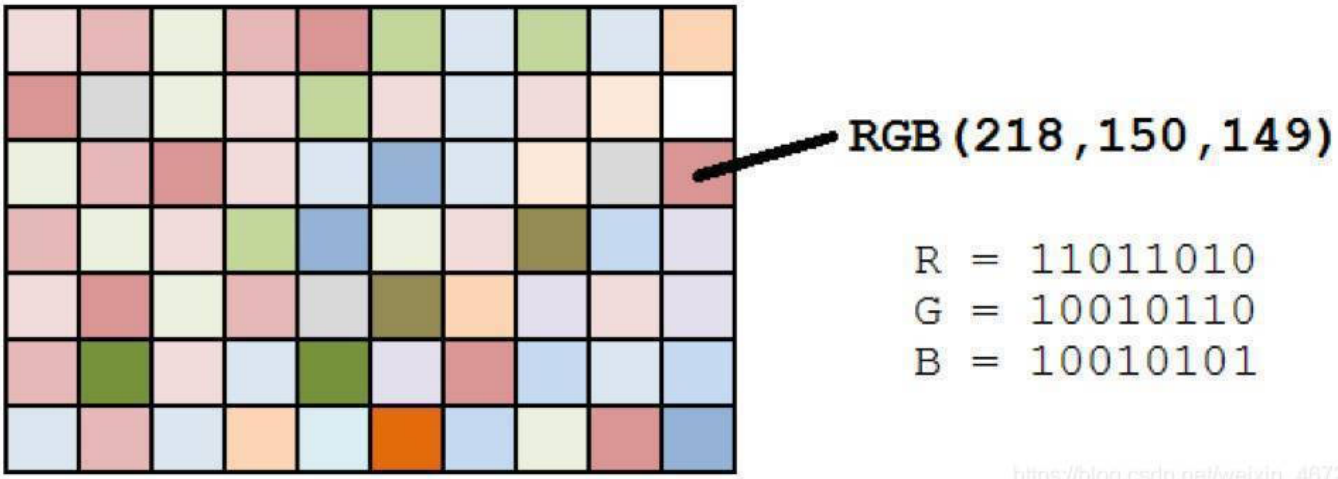
LSB全称leastsignificant bit，是一种基于图片最低有效位修改储存信息的隐写方法，，我们都知道三原色，即：红绿蓝。就是大多数的颜色都可以通过三原色的不同比例调剂出来。LSB图片隐写是基于LSB算法的一种图片隐写术。隐写是一种信息隐藏技术，这项技术目标是使对方对信息“视而不见”的效果，像诗歌中的藏头诗。

从图片格式上说

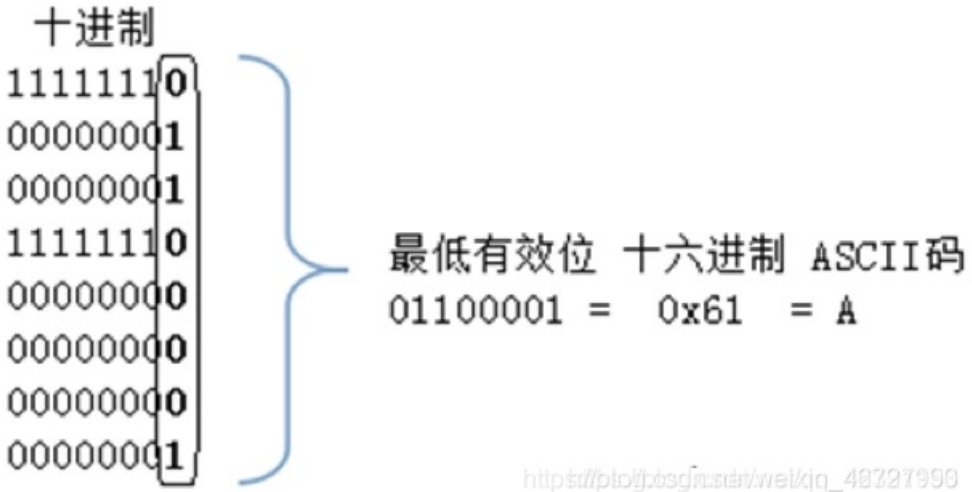
一般来说图片有BMP、JPG、BNG等格式，BMP、PNG是无损压缩的图片，而JPG是有损压缩的图片，所以一般使用PNG或BMP进行信息隐藏。一般BMP是没有压缩过的图片，所以会比较大。

从图片像素上说

人类能说识别大概1000万中颜色，而BMP等的三原色有从0x00~0xFF，也就是有256的3次方，大概是1600多万，也就是说人类大概有600万颜色无法识别，但是机器并不是通过像素识别，而是通过该像素点的数值大小识别，所以在人类眼中一样的，在机器那不一样。一个像素点占8位，每位有3个颜色，每个像素点能隐藏3个信息。



所以低有效位的数据改变在人眼看来是一样的，所以在最后一位隐藏数据是一个有效的方法。



Python脚本实现lsb隐写(python2)

```
# -*- coding: UTF-8 -*-

from PIL import Image

def plus(str):

    return str.zfill(8)

def get_key(strr):

      #获取要隐藏的文件内容

    tmp = strr
```

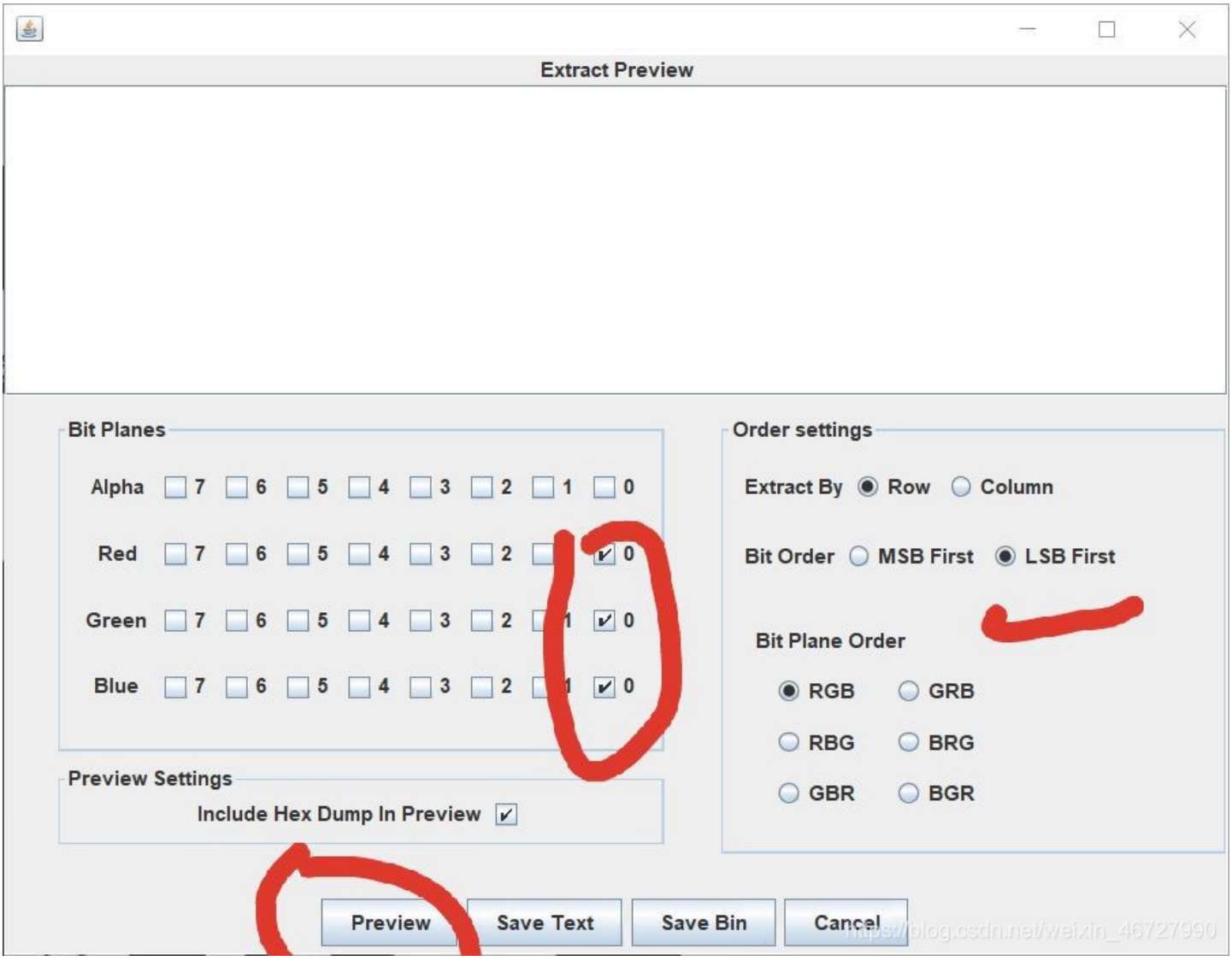
```
    f = file(tmp,"rb")
    str = ""
    s = f.read()
    for i in range(len(s)):
        #逐个字节将要隐藏的文件内容转换为二进制，并拼接起来
        #1.先用ord()函数将s的内容逐个转换为ascii码
        #2.使用bin()函数将十进制的ascii码转换为二进制
        #3.由于bin()函数转换二进制后，二进制字符串的前面会有"0b"来表示这个字符串是二进制形式，所以用replace()替换为空
        #4.又由于ascii码转换二进制后是七位，而正常情况下每个字符由8位二进制组成，所以使用自定义函数plus将其填充为8位
        str = str+plus(bin(ord(s[i])).replace('0b',''))
        #print str
    f.closed
    return str
def mod(x,y):
    return x%y;
#str1为载体图片路径，str2为隐写文件，str3为加密图片保存的路径
def func(str1,str2,str3):
    im = Image.open(str1)
    #获取图片的宽和高
    width = im.size[0]
    print "width:"+str(width)+"\n"
    height = im.size[1]
    print "height:"+str(height)+"\n"
    count = 0
    #获取需要隐藏的信息
    key = get_key(str2)
    keylen = len(key)
    for h in range(0,height):
        for w in range(0,width):
```

```
            pixel = im.getpixel((w,h))
            a=pixel[0]
            b=pixel[1]
            c=pixel[2]
            if count == keylen:
                break
            #下面的操作是将信息隐藏进去
            #分别将每个像素点的RGB值余2，这样可以去掉最低位的值
            #再从需要隐藏的信息中取出一位，转换为整型
            #两值相加，就把信息隐藏起来了
            a= a-mod(a,2)+int(key[count])
            count+=1
            if count == keylen:
                im.putpixel((w,h),(a,b,c))
                break
            b =b-mod(b,2)+int(key[count])
            count+=1
            if count == keylen:
                im.putpixel((w,h),(a,b,c))
                break
            c= c-mod(c,2)+int(key[count])
            count+=1
            if count == keylen:
                im.putpixel((w,h),(a,b,c))
                break
            if count % 3 == 0:
                im.putpixel((w,h),(a,b,c))
    im.save(str3)
#原图
old = "tset1.png"
```

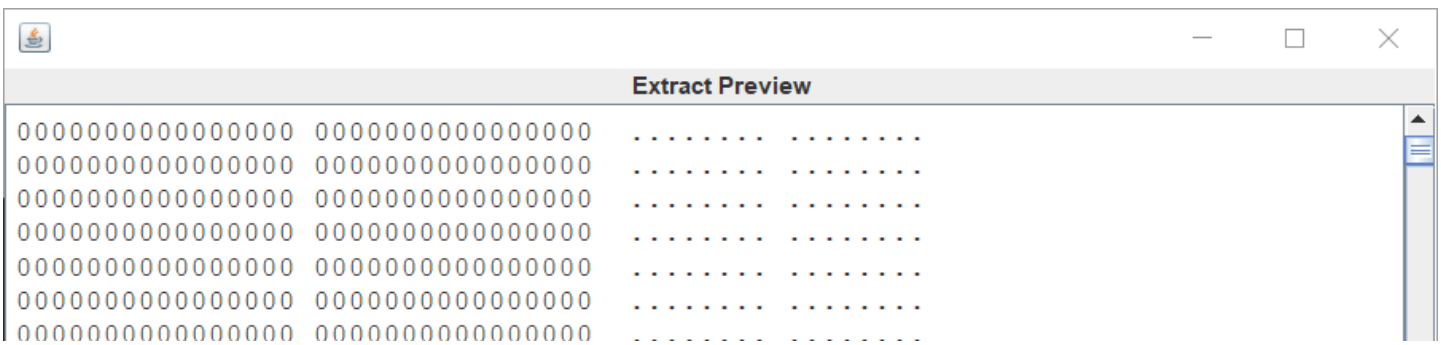
```
#处理后输出的图片路径

new = "LSBencode.png"

#需要隐藏的信息

enc = "1.txt"

func(old,enc,new)
```

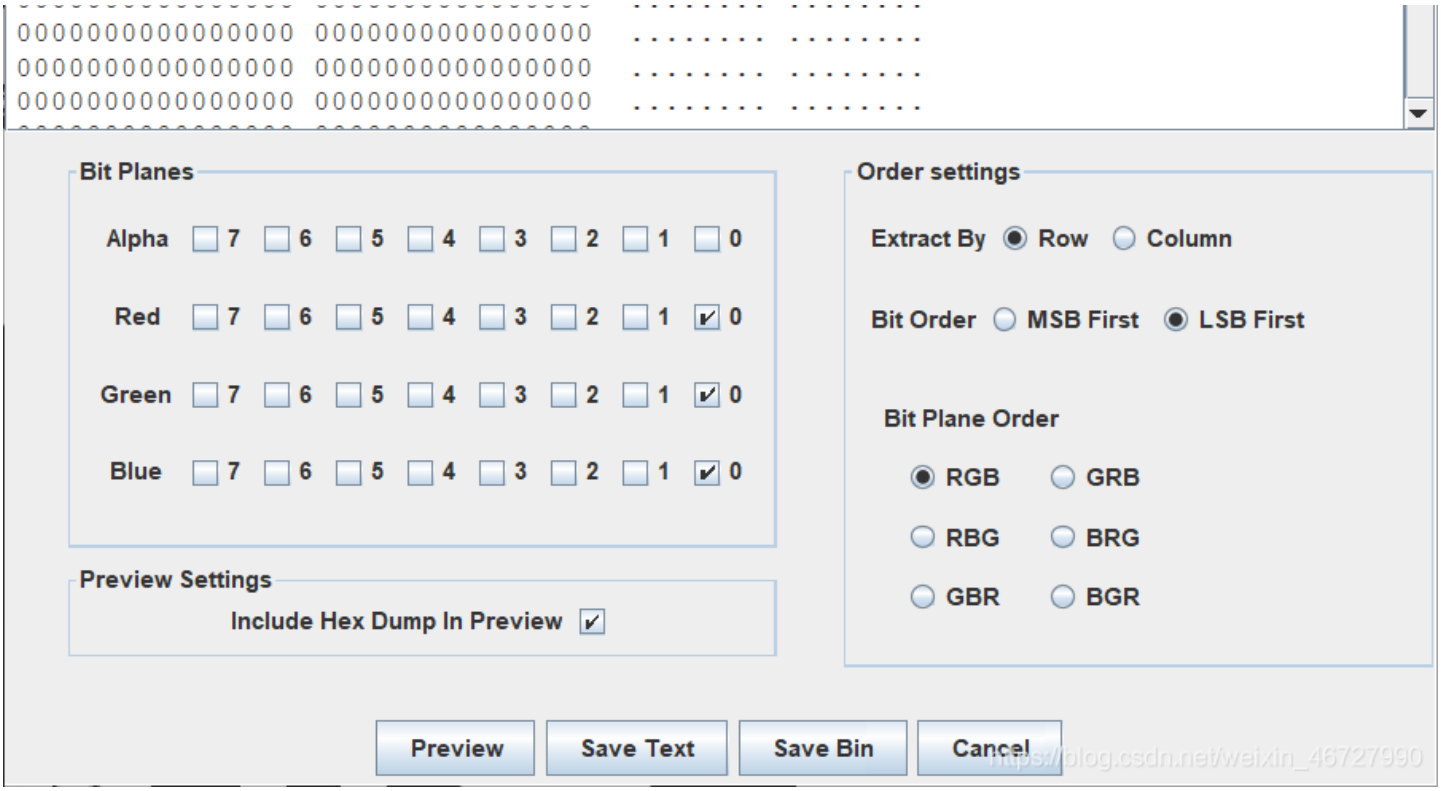
下面对一张图片进行加密实验验证：

这里我先用Stegsolve这个软件打开，使用Stegsolve——Analyse——Frame Browser这个可以浏览三个颜色通道中的每一位。（Stegsolve这个软件的下载地址是http://www.caesum.com/handbook/Stegsolve.jar）使用这个功能后需要将RGB三个位的0勾选上，并且把Bit Order选择为LSB First,因为是LSB最低位隐写，然后Preview，即可看到图片的最低位信息。如图



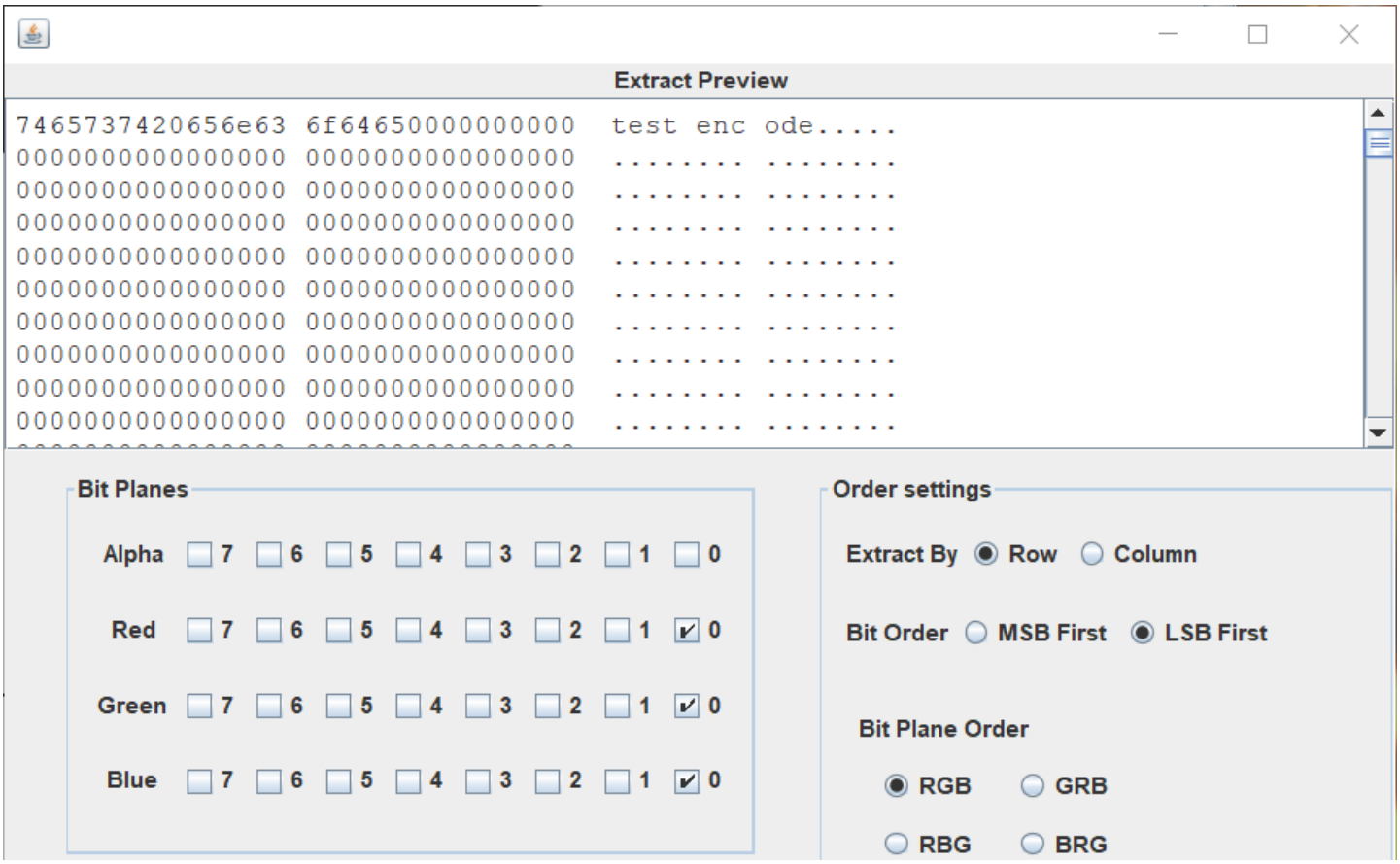
加密前图片里的信息：

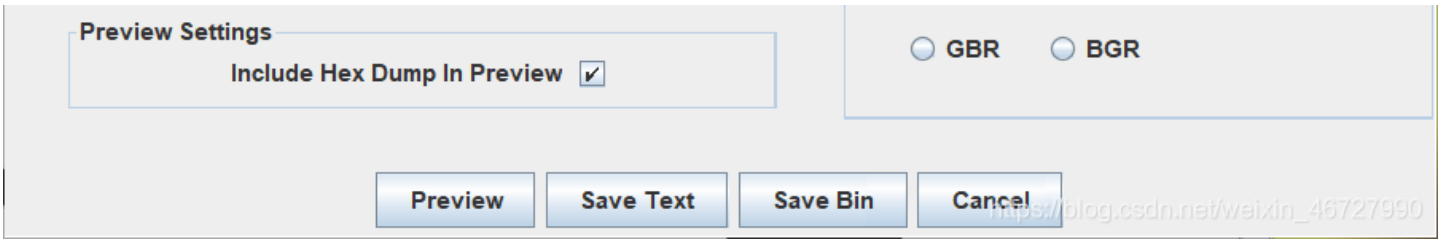




加密内容：

使用上面的脚本加密得到LSBencode.png，使用Stegsolve打开LSBencode.png，并观察最低位为：





可以看到我们需要加密的内容已经插入到图片里了。

个人总结：LSB隐写就是将需要隐藏的信息转二进制后，再将载体图片里的像素点的RGB值转二进制，然后将我们需要隐藏的信息（二进制后）逐一替换转换后的RGB值的最后一位，最后再将转换后的RGB值再转为图片输出即可，因为只改变了RGB值的最后一位，在我们人眼看来没有变化从而达到隐藏信息的功能。这里可以去ppt或者其他可以调RGB值的软件里试试将颜色的RGB值其中一位从255改为256试试看能不能看出区别。

学习大佬的文章：(https://www.jianshu.com/p/5d2b3f0edff1)