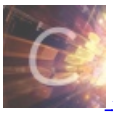
用java实现LSB图像隐写算法

原创

太菜了怎么办？ 于 2021-05-23 10:38:56 发布 1197 收藏 22

分类专栏： java 密码学 文章标签： java 图像处理 算法 经验分享

本文链接：https://blog.csdn.net/qq_43632414/article/details/117188876



没错，这次又是因为被老师部置的作业难到，经历一番学习之后的总结bolg，T-T……5555

1.准备知识

1.1像素

百度百科中像素的定义是这样的

也就是说一张图片是由很多个像素单元组成的

像素又是由什么组成的呢？ 答：RGB三原色！

学过ps的都应该知道，一个像素点是由RGB三原色

java中也有这样的接口

如果说我只是将r由236，改变到237，颜色变化是并不大的！所以这就我们就LSB算法是完全可以实现的

2.2RGB

在java中，你导入一张图片到程序中(ImageIO)，R、G、B代表的是三个字节！一个像素点是由三个字节组成

一般电脑像素(分辨率)是1920x1080

也就是所是由说你的屏幕上的图片是是由宽1920，高1080的1920x1080个像素点组成

也就是1920x1080x3个字节组成

1.3LSB的实现

就是将要加密数据转成2进制数据，然后按照规则放入每个像素点三个字节中的最后一位

1.4开始着手

我只改变了每个像素点中的R所对应的字节的最后一位 B，G都无变化
图片最好选用png格式的，像jpg格式的图片都会经过压缩，最低有效位会产生变化

2代码实现

```
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Scanner;

public class LSB {
    public static void main(String[] args) throws IOException {
        Scanner scan = new Scanner(System.in);
        System.out.println("***************LSB图像隐写编码程序****************");
        System.out.println("*********请选择想要使用的功能(输入数字)*************");
        System.out.println();
        System.out.println("******************1.LSB编码********************");
        System.out.println("******************2.LSB解码********************");
        System.out.println();
        System.out.print("请输入你想选择的功能:");
        String choice = scan.next();
        switch(choice){
            case "1":
                System.out.print("请输入需要加密的文件的路径:");
                String textPath = scan.next();
                System.out.print("请输入png图像辅助文件的路径:");
                String imagePath = scan.next();
                System.out.print("最后再输入一下生成的png图片的保存路径:");
                String  imageOutputPath= scan.next();
                LSBEncoder(textPath,imagePath,imageOutputPath);
                scan.close();
                break;
            case "2":
                System.out.print("请输入待解码的png图片的路径:");
                String imageInputPath = scan.next();
                System.out.print("请输入解码后，存放数据的文件名称");
                String textFilePath = scan.next();
                LSBDecoder(imageInputPath,textFilePath);
                scan.close();
                break;
            default:
                System.out.print("谁叫你乱按键盘的啊！！！活该不能执行，哼！");
                scan.close();
                break;
        }
    }
    public static void LSBEncoder(String textPath, String imagePath,String imageOutputPath ) throws IOException
{

        //读取png图像
        BufferedImage image = ImageIO.read(new File(imagePath));
        int width = image.getWidth();
        int height = image.getHeight();
        int[][][] rgb = new int[width][height][3];
        //将图像每个点的像素(R,G,B)存储在数组中
        for (int w = 0; w < width; w++) {
            for (int h = 0; h < height; h++) {
                int pixel = image.getRGB(w, h);//读取的是一个24位的数据
                //数据三个字节分别代表R、B、G
```

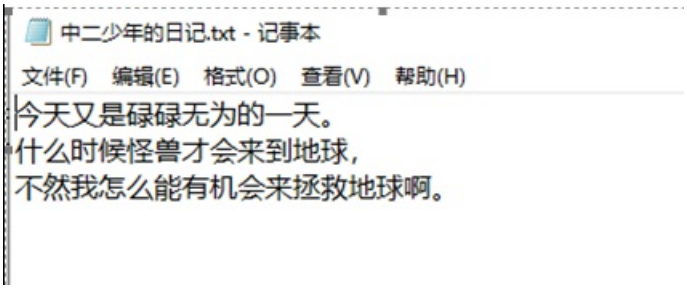
```
                rgb[w][h][0] = (pixel & 0xff0000) >> 16;//R
                rgb[w][h][1] = (pixel & 0xff00) >> 8;//B
                rgb[w][h][2] = (pixel & 0xff);//G
            }
        }

        //导入待加密机密文件
        FileInputStream fis = new FileInputStream(textPath);
        int byteLen = fis.available();
        byte[] buf = new byte[byteLen];
        fis.read(buf);


        //我用两个字节(16位)表示数据部分的长度，也就是说，图像转像素点后，前16个像素点的R对应的字节的最低有效位都是默认表示数据字节数组的长度
        //规定，数据的长度最大为2^16-1
        int[] bufLen = new int[2];
        bufLen[0] = (byteLen & 0xff00 ) >> 8;
        bufLen[1] = (byteLen & 0xff);
        for (int i = 0; i < 2; i++) {
            for (int j = 7; j >= 0; j--) {
                int h =(i * 8 +(7 - j)) / width;
                int w = (i * 8 + (7-j)) % width;
                //只取每个像素点的R，的字节的最低位
                if((bufLen[i] >>j & 1) == 1 ){
                    rgb[w][h][0] = rgb[w][h][0] | 1;
                } else {
                    rgb[w][h][0] = rgb[w][h][0] & 0xe;
                }
            }
        }

        //按照规则将数据的二进制序列全部放到每一个像素点的第一个字节的最后一位上
        for (int i = 2; i < byteLen + 2; i++) {
            for (int j = 7; j >= 0; j--) {
                //高
                int h = (i * 8 + (7 - j)) / width ;
                //宽
                int w = (i * 8 + (7 - j)) % width;
                if ((buf[i-2] >> j & 1) == 1) {
                    rgb[w][h][0] = rgb[w][h][0] | 1;//变1
                } else {
                    rgb[w][h][0] =rgb[w][h][0] & 0xe;
                }
            }
        }
        //         构建通过编码生成的png图片的类型
        BufferedImage imageOutput = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        for (int w = 0; w < width; w++) {
            for (int h = 0; h < height; h++) {
                int[] color = new int[3];
                color[0] = rgb[w][h][0] << 16;
                color[1] = rgb[w][h][1] << 8;
                color[2] = rgb[w][h][2];
                int pixel = 0xff000000 | color[0] | color[1] | color[2];
                imageOutput.setRGB(w, h, pixel);
            }
        }
        ImageIO.write(imageOutput, "png", new File(imageOutputPath));
    }
```

```
    public static void LSBDecoder(String imageInputPath,String textFilePath) throws IOException {
        BufferedImage imageInput = ImageIO.read(new File(imageInputPath));
        int width = imageInput.getWidth();
        int height = imageInput.getHeight();
        int[] bufLen=  new int[2];
        //将图像每个点的像素(R,G,B)存储在数组中
        for (int i = 0; i < 2; i++) {
            int[] bits = new int[8];
            for (int j = 7; j >= 0; j--) {
                int h =(i * 8 +(7 - j)) / width;
                int w = (i * 8 + (7-j)) % width;
                int pixel = imageInput.getRGB(w,h);
                int r = (pixel & 0xff0000) >> 16;
                bits[j] = (r & 1) << j;
            }
            bufLen[i] =  bits[7] | bits[6] | bits[5] | bits[4] | bits[3] | bits[2] | bits[1] | bits[0];
        }
        int byteLen =  ( (bufLen[0] << 7) |bufLen[1]);
//        System.out.println(byteLen);
        byte[] buf = new byte[byteLen];
        for (int i = 2; i < byteLen + 2; i++) {
            int[] bits = new int[8];
            for (int j = 7; j >= 0; j--) {
                int h = (i * 8 + (7 - j)) / width;
                int w = (i * 8 + (7 - j)) % width;
                int pixel = imageInput.getRGB(w, h);
                int r = (pixel & 0xff0000) >> 16;
                bits[j] = (r & 0x1) << j;
            }
            buf[i-2] = (byte)(bits[7] | bits[6] | bits[5] | bits[4] | bits[3] | bits[2] | bits[1] | bits[0]);
        }
//        System.out.println(new String(buf));
        FileOutputStream fos = new FileOutputStream(textFilePath);
        fos.write(buf);
        fos.close();
    }
}
```

3.实现截图

1) 准备

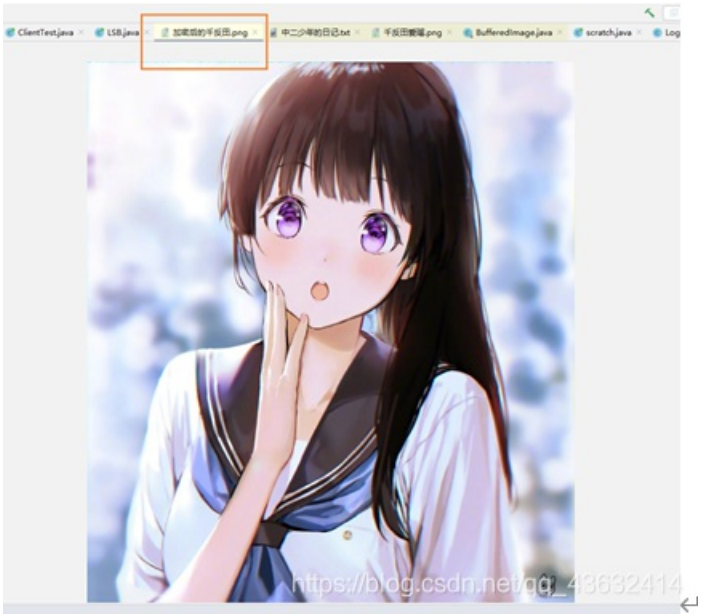
待加密文件的内容(txt)文件



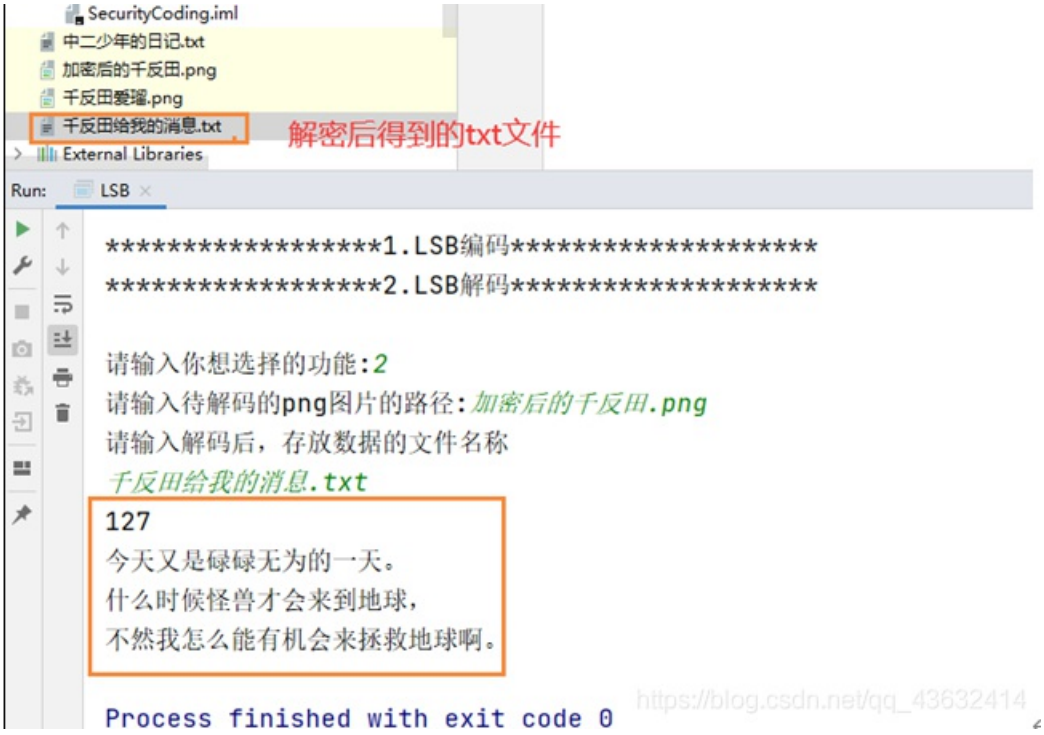
辅助用的png图片(我老婆) 最好是使用png图片

2）运行代码，实现通过png图片对txt文件经行加密处理

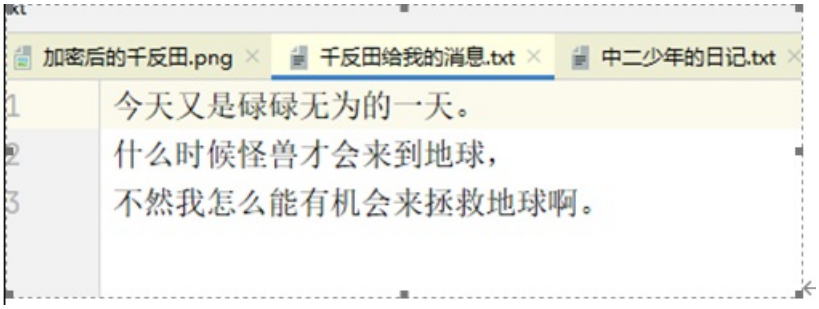
生成的png，和原图基本上毫无区别



3）运行代码，通过加密后png文件读出



生成的txt文件



（==结束语，文章中若存在什么问题，欢迎指教！^-^==）