# 一．[LSB](http://www.eepw.com.cn/news/listbylabel/label/LSB)[算法](http://www.eepw.com.cn/news/listbylabel/label/%E7%AE%97%E6%B3%95)的基本原理是：

对空域的[LSB](http://www.eepw.com.cn/news/listbylabel/label/LSB)做替换，用来替换[LSB](http://www.eepw.com.cn/news/listbylabel/label/LSB)的序列就是需要加入的水印信息、水印的数字摘要或者由水印生成的伪随机序列。由于水印信息嵌入的位置是LSB，为了满足水印的不可见性，允许嵌入的水印强度不可能太高。然而针对空域的各种处理，如游程编码前的预处理，会对不显著分量进行一定的压缩，所以LSB[算法](http://www.eepw.com.cn/news/listbylabel/label/%E7%AE%97%E6%B3%95)对这些操作很敏感。因此LSB[算法](http://www.eepw.com.cn/news/listbylabel/label/%E7%AE%97%E6%B3%95)最初是用于脆弱性水印的。

1.算法

import javax.imageio.ImageIO;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import java.util.ArrayList;

public class LSB {

    //图片宽高

    private static int imgHight = 1;

    private static int imgWidth = 1;

    private static int imgHight2= 1;

    private static int imgWidth2 = 1;

    //像素矩阵集合

    private static ArrayList<String[][]> resList = new ArrayList<>();

    //原像素矩阵

    public static void main(String[] args) throws IOException {

//加密

        encryption("img/zui10/载体图像.png","img/zui10/秘密图像预处理6.png");

        //解密

        Decrypt("img/zui10/已伪装图像.png");

    }

    /\*\*

     \* 初始化并加密

     \* @param src 载体图像路径

     \* @param src2 秘密图像路径

     \* @throws IOException

     \*/

    public static void encryption(String src, String src2) throws IOException {

        System.out.println("载体图像路径："+src+" 秘密图像路径："+src2);

        System.out.println("开始加密....");

        //图像操纵对象

        BufferedImage imgSrc = ImageIO.read(new File(src));

        BufferedImage imgSrc2 = ImageIO.read(new File(src2));

        //图片类型

        int imgType = 5;

        imgHight = imgSrc.getHeight();

        imgWidth = imgSrc.getWidth();

        imgHight2 = imgSrc2.getHeight();

        imgWidth2 = imgSrc2.getWidth();

        //创建存储像素信息的三个矩阵

        resList.clear();

        resList.add(ru(imgWidth,imgHight));

        resList.add(ru(imgWidth2,imgHight2));

        resList.add(ru(imgWidth,imgHight));

        //记录载体图像像素信息到第一个矩阵里

        for (int i = 0; i < imgWidth; i++) {

            for (int j = 0; j < imgHight; j++) {

                resList.get(0)[i][j] = Integer.toBinaryString(imgSrc.getRGB(i, j));

            }

        }

        //记录秘密图像像素信息到第二个矩阵里

        for (int i = 0; i < imgWidth2; i++) {

            for (int j = 0; j < imgHight2; j++) {

                resList.get(1)[i][j] = Integer.toBinaryString(imgSrc2.getRGB(i, j));

            }

        }

        //默认从左上角开始隐藏

        for (int i = 0; i < imgWidth; i++) {

            //如果小于隐藏图像的宽度，列号

            if (i<imgWidth2){

                for (int j = 0; j < imgHight; j++) {

                    //且小于隐藏图像的高度，行号

                    //则为秘密图像的宽高范围，应发生置换

                    if (j<imgHight2){

                        char[] chars1 = resList.get(0)[i][j].toCharArray();

                        char[] chars3 = chars1;

                        char[] chars2 = resList.get(1)[i][j].toCharArray();

                        //就将对应的位置信息放置在对应像素的透明度、红、绿、蓝通道的低4bit位

                        //采用U型替换规则，加密顺时针替换，解密逆时针替换

                        //蓝通道

                        chars3[31] = chars2[24];

                        chars3[30] = chars2[25];

                        chars3[29] = chars2[26];

                        chars3[28] = chars2[27];

                        //绿通道

                        chars3[23] = chars2[16];

                        chars3[22] = chars2[17];

                        chars3[21] = chars2[18];

                        chars3[20] = chars2[19];

                        //红通道

                        chars3[15] = chars2[8];

                        chars3[14] = chars2[9];

                        chars3[13] = chars2[10];

                        chars3[12] = chars2[11];

                        //透明度通道

                        chars3[7] = chars2[0];

                        chars3[6] = chars2[1];

                        chars3[5] = chars2[2];

                        chars3[4] = chars2[3];

                        //保存置换后的像素信息至第三个矩阵中

                        resList.get(2)[i][j] = String.valueOf(chars3);

                    }else {

                        //保存没有置换的像素信息至第三个矩阵中

                        resList.get(2)[i][j] = String.valueOf(resList.get(0)[i][j].toCharArray());

                    }

                }

            }else {

                for (int j = 0; j < imgHight; j++) {

                    resList.get(2)[i][j] = String.valueOf(resList.get(0)[i][j].toCharArray());

                }

            }

        }

        //输出加密结果

        oimg(resList.get(2),"已伪装图像",imgWidth,imgHight);

        System.out.println("加密完成....");

    }

    public static void Decrypt(String src) throws IOException {

        //图像操纵对象

        BufferedImage imgSrc = ImageIO.read(new File(src));

        //图片类型

        int imgType = 5;

        imgWidth = imgSrc.getWidth();

        imgHight = imgSrc.getHeight();

        System.out.println(imgWidth2);

        System.out.println(imgHight2);

        resList.clear();

        resList.add(ru(imgWidth,imgHight));

        resList.add(ru(imgWidth2,imgHight2));

        for (int i = 0; i < imgWidth; i++) {

            if (i<imgWidth2){

                for (int j = 0; j < imgHight; j++) {

                    if (j<imgHight2){

                        char[] chars1 = Integer.toBinaryString(imgSrc.getRGB(i, j)).toCharArray();

                        char[] chars2 = new char[32];

                        char[] chars3 = new char[32];

                        for (int h = 0; h < 32; h++){

                            chars2[h] = '0';

                            chars3[h] = '0';

                        }

                        chars2[24] = chars1[24];

                        chars2[25] = chars1[25];

                        chars2[26] = chars1[26];

                        chars2[27] = chars1[27];

                        chars3[24] = chars1[31];

                        chars3[25] = chars1[30];

                        chars3[26] = chars1[29];

                        chars3[27] = chars1[28];

                        chars2[16] = chars1[16];

                        chars2[17] = chars1[17];

                        chars2[18] = chars1[18];

                        chars2[19] = chars1[19];

                        chars3[16] = chars1[23];

                        chars3[17] = chars1[22];

                        chars3[18] = chars1[21];

                        chars3[19] = chars1[20];

                        chars2[8] = chars1[8];

                        chars2[9] = chars1[9];

                        chars2[10] = chars1[10];

                        chars2[11] = chars1[11];

                        chars3[8] = chars1[15];

                        chars3[9] = chars1[14];

                        chars3[10] = chars1[13];

                        chars3[11] = chars1[12];

                        chars2[0] = chars1[0];

                        chars2[1] = chars1[1];

                        chars2[2] = chars1[2];

                        chars2[3] = chars1[3];

                        chars3[0] = chars1[7];

                        chars3[1] = chars1[6];

                        chars3[2] = chars1[5];

                        chars3[3] = chars1[4];

                        resList.get(0)[i][j] = String.valueOf(chars2);

                        resList.get(1)[i][j] = String.valueOf(chars3);

                    }else {

                        resList.get(0)[i][j] = Integer.toBinaryString(imgSrc.getRGB(i, j));

                    }

                }

            }else {

                for (int j = 0; j < imgHight; j++) {

                    resList.get(0)[i][j] = Integer.toBinaryString(imgSrc.getRGB(i, j));

                }

            }

        }

        oimg(resList.get(0),"解密后载体图像",imgWidth,imgHight);

        oimg(resList.get(1),"解密后秘密图像",imgWidth2,imgHight2);

        System.out.println("解密完成....");

    }

    /\*\*

     \* 生产像素矩阵

     \* @param width

     \* @param hight

     \* @return

     \*/

    public static String[][] ru(int width,int hight){

        return new String[width][hight];

    }

    /\*\*

     \* 图片输出方法

     \* @param matrix

     \* @param name

     \* @param imgWidth

     \* @param imgHight

     \*/

    public static void oimg(String[][] matrix, String name,int imgWidth, int imgHight){

        BufferedImage imgRes = new BufferedImage(imgWidth, imgHight, 5);

        for (int i = 0; i < imgWidth; i++) {

            for (int j = 0; j < imgHight; j++) {

//                System.out.println("第"+name+"个  i:"+i+"  j:"+j+"  值："+Integer.parseUnsignedInt(r[i][j],2));

                imgRes.setRGB(i,j,Integer.parseUnsignedInt(matrix[i][j],2));

            }

        }

        File imgOut = new File("img/zui10/"+name+".png");

        //输出文件

        try {

            ImageIO.write(imgRes, "png", imgOut);

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

}

# 二．信息安全工程师知识点：Patchwork算法

该算法是随机选择N对像素点(ai，bi)，然后将每个ai点的亮度值加1，每个bi点的亮度值减1，这样整个图像的平均亮度保持不变。适当地调整参数，Patchwork方法对JPEG压缩、FIR滤波以及商像裁剪有一定的抵抗力，但该方法嵌入的信息量有限。为了嵌入更多的水印信息，可以将图像分块，然后对每一个图像块进行嵌入操作。

1.算法

clc;

clear ;

figure(1);      %打开窗口

%% 载入数据

I=imread("I.png");

%I=imresize(I,[512,512],'nearest');

subplot( 2,2,1) ,imshow(I),title('载体图像');

W=imread("W.png");

%W=imresize(W,[64,64],'nearest');

subplot(2,2,2),imshow(W),title('水印图像');

[row,col,t] = size(W);

%% 嵌入水印部分

%分离R，G，B通道

IR=I(:,:,1);

IG=I(:,:,2);

IB=I(:,:,3);

WR=W(:,:,1);

WG=W(:,:,2);

WB=W(:,:,3);

%设置k,不同图片k值不同

k=0.162;

%% 使用Arnold变换置乱水印

WRA=arnold(WR,1,1,1);

WRA=double(WRA);

WGA=arnold(WG,1,1,1);

WGA=double(WGA);

WBA=arnold(WB,1,1,1);

WBA=double(WBA);

%%

% 对载体图像进行8\*8分块处理，然后对每块分别DCT变化

IRD=blkproc(IR,[8,8],'dct2');

IGD=blkproc(IG,[8,8],'dct2');

IBD=blkproc(IB,[8,8],'dct2');

IRDE=IRD;

IGDE=IGD;

IBDE=IBD;

%% 提取直流分量,并向矩阵中嵌入水印

for i=0:(row-1)

    for j=0:(col-1)

        x=i\*8;

        y=j\*8;

        IRDE(x+1,y+1)=IRD(x+1,y+1)+k\*WRA(i+1,j+1);

        IBDE(x+1,y+1)=IBD(x+1,y+1)+k\*WBA(i+1,j+1);

        IGDE(x+1,y+1)=IGD(x+1,y+1)-k\*WGA(i+1,j+1);

    end

end

%% 对载体图像进行分块反DCT变换

IR2=blkproc(IRDE,[8,8],'idct2');

IG2=blkproc(IGDE,[8,8],'idct2');

IB2=blkproc(IBDE,[8,8],'idct2');

IR2=uint8(IR2);

IG2=uint8(IG2);

IB2=uint8(IB2);

%% 合成

I\_embed = I;

I\_embed(:,:,1) = IR2;

I\_embed(:,:,2) = IG2;

I\_embed(:,:,3) = IB2;

subplot( 223) ,imshow(I\_embed),title('嵌入水印后的载体图像');

%% 提取水印

%% 分离通道

P=I\_embed;

PR=P(:,:,1);

PG=P(:,:,2);

PB=P(:,:,3);

%% 将带水印图像进行DCT变换

PRD=blkproc(PR,[8,8],'dct2');

PGD=blkproc(PG,[8,8],'dct2');

PBD=blkproc(PB,[8,8],'dct2');

WR2=WR;

WB2=WB;

WG2=WG;

%% 提取水印

for i=0:(row-1)

    for j=0:(col-1)

        x=i\*8;

        y=j\*8;

        WR2(i+1,j+1)=(PRD(x+1,y+1)-IRD(x+1,y+1))/k;

        WB2(i+1,j+1)=(PBD(x+1,y+1)-IBD(x+1,y+1))/k;

        WG2(i+1,j+1)=(IGD(x+1,y+1)-PGD(x+1,y+1))/k;

    end

end

%% 逆arnold

WR2=uint8(WR2);

WG2=uint8(WG2);

WB2=uint8(WB2);

WR2=rearnold(WR2,1,1,1);

WG2=rearnold(WG2,1,1,1);

WB2=rearnold(WB2,1,1,1);

%% 合成水印

W2=W;

W2(:,:,1)=WR2;

W2(:,:,2)=WG2;

W2(:,:,3)=WB2;

subplot( 224) ,imshow(W2),title('提取出的水印');

imwrite(I\_embed,'I\_embed.png');

imwrite(W2,'W2.png');

%% arnold变换

function arnoldImg = arnold(img,a,b,n)

[h,w] = size(img);

N=h;

arnoldImg = zeros(h,w);

for i=1:n

    for y=1:h

        for x=1:w

            %防止取余过程中出现错误，先把坐标系变换成从0 到 N-1

            xx=mod((x-1)+b\*(y-1),N)+1;

            yy=mod(a\*(x-1)+(a\*b+1)\*(y-1),N)+1;

            arnoldImg(yy,xx)=img(y,x);

        end

    end

    img=arnoldImg;

end

arnoldImg = uint8(arnoldImg);

end

function img = rearnold(arnoldImg,a,b,n)

[h,w] = size(arnoldImg);

img = zeros(h,w);

N = h;

for i=1:n

    for y=1:h

        for x=1:w

            xx=mod((a\*b+1)\*(x-1)-b\*(y-1),N)+1;

            yy=mod(-a\*(x-1)+(y-1),N)+1  ;

            img(yy,xx)=arnoldImg(y,x);

        end

    end

    arnoldImg=img;

end

img = uint8(img);

end