

# 数字内容安全实验第3组实验报告

## 一. 实验目的

熟悉并仿真测试wong脆弱水印算法，分析不足以及提出改进方案。

## 二. 实验环境

python 3.10.7 vscode

## 三. wong脆弱水印算法的介绍

wong脆弱水印算法的原理：

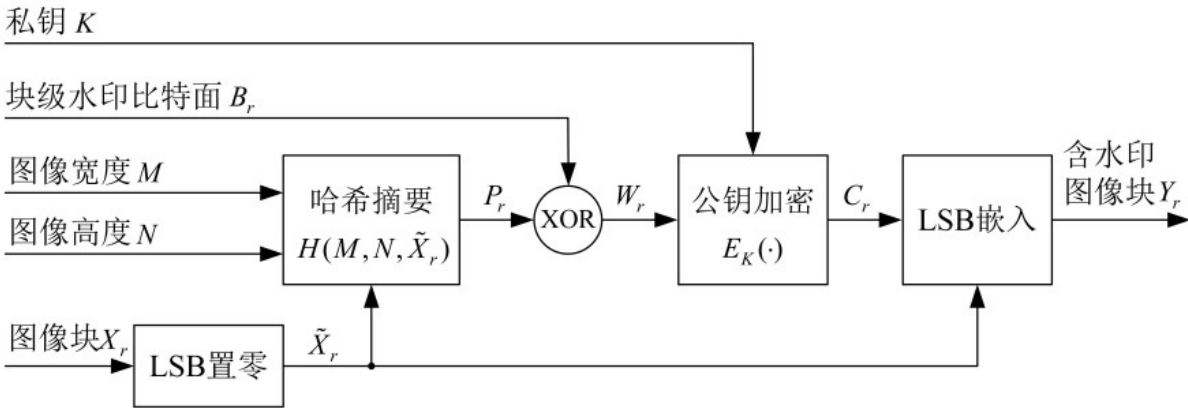
利用hash函数的单向性和能在输入发生微小变化时得到完全不同输出值的特点，结合图像分块，嵌入的水印来完成对图像像素值微小改变的定位。借助公钥系统的便利性，使得任何人，只要他能获得作者的公钥，便可以完成对图像的完整性检测和身份认证。

wong脆弱水印算法的目的：在加入水印的图像中，即使像素值微小改变，提取出的水印上都能直观的显示出被篡改的区域。

所以对于此算法，它的鲁棒性理想值就是  $0$ ，它的特点就是对任何改动都十分敏感。它并没有隐藏信息的需要，所以没必要讨论其容量。而它的透明性是关键，因为水印的嵌入是基于LSB的，它无法抵御卡方分析的检测。

接下来重点介绍水印嵌入与提取。

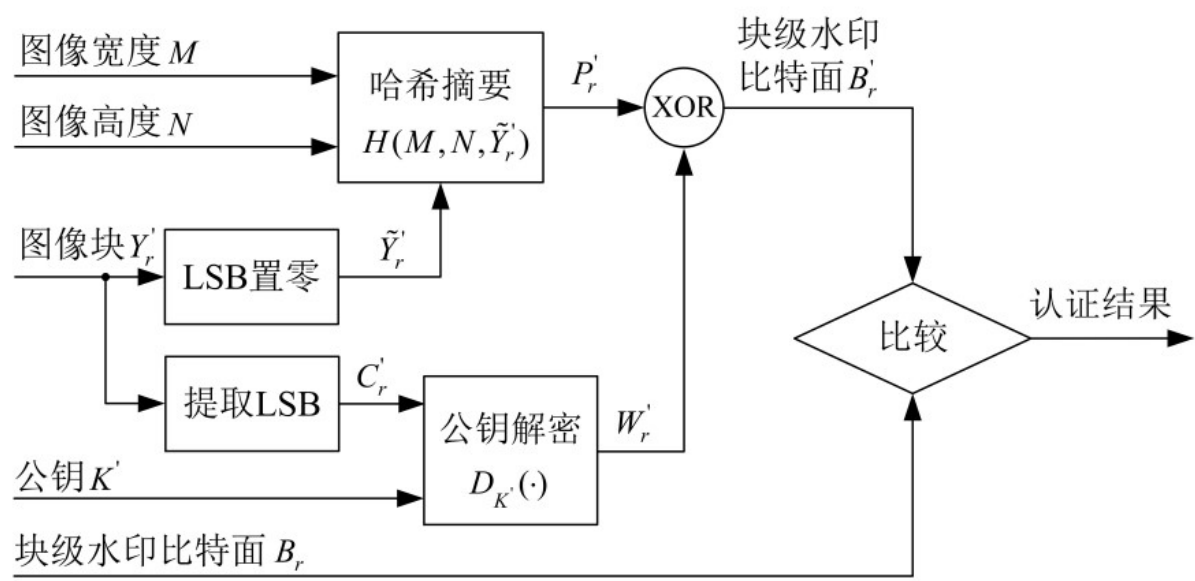
### 水印嵌入流程图



[1] A Public Key Watermark for Image Verification and Authentication\_ICIP1998 (cite500+)

[2] Secret and Public Key Image Watermarking Schemes for Image Authentication and Ownership Verification\_TIP2001 (cite500+)

水印提取/验证流程图



在嵌入和提取水印的过程中, 利用了异或操作的特点,即:  $Pr \oplus Br = Wr$ , 而  $Wr' \oplus Pr' = Br'$

原图像的信息生成的哈希摘要 $Pr$ 与块级水印比特面 $Br$ 进行异或, 得到二进制流 $Wr$ ,  $Wr$ 进行加密后嵌入图像的LSB位。

嵌入的水印,块级水印比特面 $Br$ 为二值图像。

对于传输过来的图像信息生成的哈希摘要 $Pr'$ 与提取出来解密后的 $Wr'$ 进行异或, 即可得到嵌入的水印 $Br$ 。

如果图像没有被篡改, 那么 $Wr'$ 与原图像的信息生成的哈希摘要 $Pr'$ 异或后得到的嵌入水印 $Br'$ 与原 $Br$ 在视觉效果上相同。

如果图像被篡改了, 图像生成的哈希摘要 $Pr'$ , 提取出来的二进制流 $Wr$ 就会与 $Pr$ ,  $Wr$ 产生很大的差异, 从而使得二值图像 $Br$ 的视觉效果产生很大的差异, 从而使得我们大致定位到图像被篡改的位置, 从而实现我们的目的。

四. 实验内容及步骤

- 1.我们读取图像1,2,3,4。图像1的大小为256x256, 图像2的大小为512x512, 图像3的大小为1024x1024, 图像4的大小为2048x2048。将图片进行水印的添加和提取。
- 2.对彩色图像进行水印嵌入和提取, 展示彩色图像嵌入水印前和嵌入水印后的图像, 然后展示R,G,B三个通道嵌入的水印以及和起来嵌入的水印。
- 3.计算图像1, 2, 3, 4的PSNR值和SSIM值并绘制成为表格, 进行不透明性分析, 观察并分析差异。
- 4.计算图像1, 2, 3, 4图像在嵌入水印前的灰度直方图和嵌入水印后的灰度直方图。

- 5.篡改(通过对左上角小块图像区域内像素值加一进行图像的篡改，下面报告中提到的篡改均是这  
样)图像1左上角的不同小块，观察图像以及提取的水印的前后变化。
- 6.对图像1嵌入水印后的图像添加椒盐噪点，椒噪声密度为0.001，盐噪声密度为0.001，提取水  
印，观察水印的变化。
- 7.对图像1嵌入水印后的图片添加高斯噪声，提取水印，观察图像以及提取的水印的变化。
- 8.对图像进行特定的区域复制粘贴攻击，看看wong水印算法能否抵挡此攻击。

五. 对wong水印算法进行最基本的实现




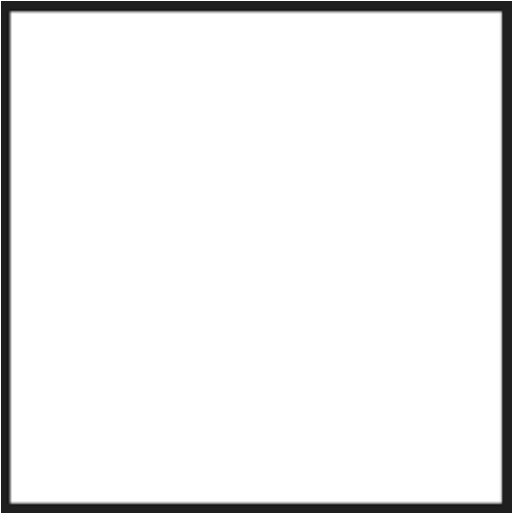

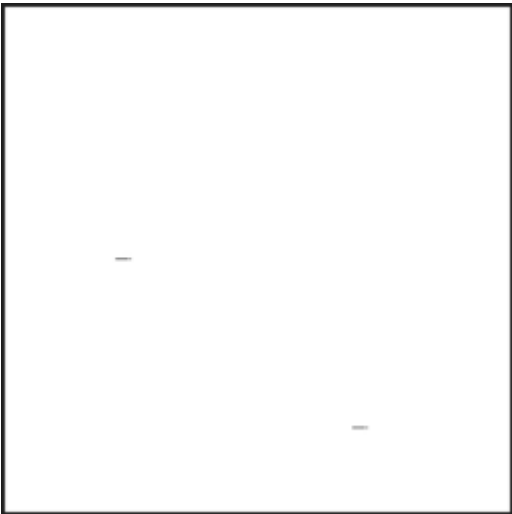
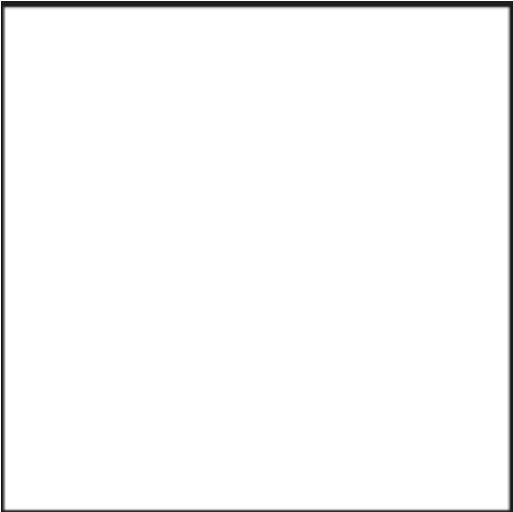
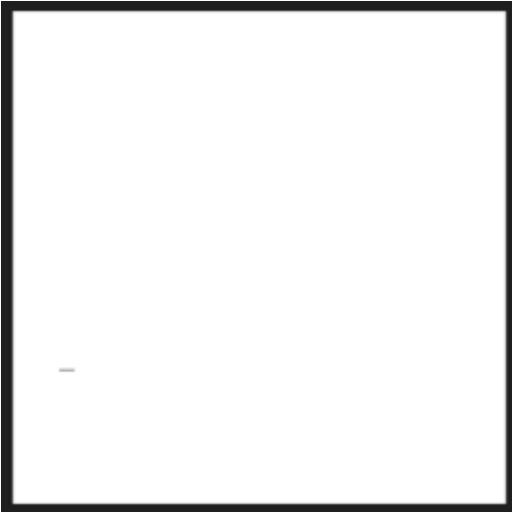
我们选择嵌入的水印是白色图像,如上图

注：因为我们选择的水印是白色的原因，所以如果不加以修改，呈现出来的实验报告会感觉看起来图片不存在一样，因为为了看起来方便将图像加了黑边，下面的报告中还有很多图片也是如此



六. 对彩色图像进行水印嵌入和提取，展示彩色图像嵌入水印前和嵌入水印后的图像，然后展示R,G,B三个通道嵌入的水印以及和起来嵌入的水印。

下面是展示

原图像	准备嵌入的水印
	
嵌入水印后的图像	从B通道中提取出的水印
	
从G通道中提取出的水印	从R通道中提取出的水印
	

## 七. 图像分析

在完成了wong水印算法最基本的代码后，我们开始对它的各种指标进行一系列的分析

使用四幅图像，分别为上面提到的1， 2， 3， 4.根据这四幅图像PSNR值和SSIM值绘制了这幅图像 对wong脆弱水印算法进行不可见性分析。

### 1.PSNR分析：

- PSNR评价标准：
- PSNR值越大，表示图像的质量越好，一般来说：
- (1) 高于40dB：说明图像质量极好(即非常接近原始图像)

(2) 30–40dB：通常表示图像质量是好的(即失真可以察觉但可以接受)

(3) 20–30dB：说明图像质量差

(4) 低于20dB：图像质量不可接受

PSNR值统一大于五十，说明图像质量极好。

### 2.SSIM分析：

结构相似性指数 (Structural Similarity Index, SSIM) 是一种用于测量两幅图像结构相似程度的指标。SSIM综合考虑了亮度、对比度和结构三个方面的信息，以更全面地评估图像的相似性。

SSIM的计算公式如下：

$$SSIM(x,y)=\frac{(2\mu_x\mu_y+c_1)(2\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)}$$

- 其中，x和y分别表示待比较的两幅图像， $\mu_x$ 和 $\mu_y$ 表示图像x和图像y的像素均值， $\sigma_x$ 和 $\sigma_y$ 表示图像x和图像y的像素标准差， $\sigma_{xy}$ 表示图像x和图像y的像素协方差，c1和c2是两个常数，用于稳定计算。
- SSIM的取值范围在[-1, 1]之间，数值越接近1表示两幅图像结构相似度越高，数值越接近-1表示结构相似度越低，通常认为，当 SSIM 值大于 0.8 时，两个图像具有较高的相似度。
- SSIM的计算考虑了亮度、对比度和结构的差异，因此在图像质量评价、图像复原和图像压缩等领域都具有广泛应用。

图片1,2,3,4的SSIM值都极限接近于1，说明wong脆弱水印算法嵌入水印后的图像与原图像及其相似。

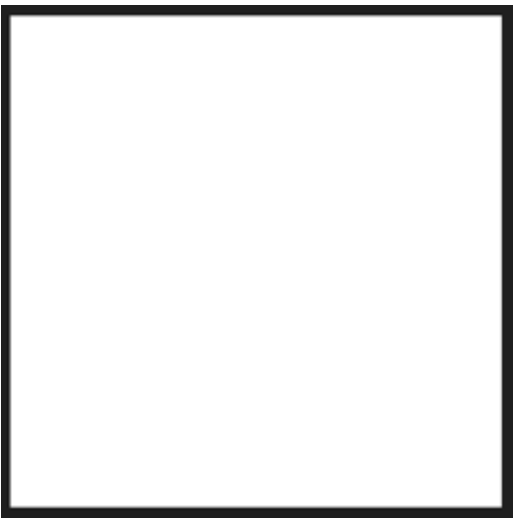
### 3.篡改图片1左上角小块，观察图片以及提取的水印的前后变化。

接下来我们对图片1左上角的小块进行了篡改，对wong脆弱水印算法进行敏感性测试。

原图片



嵌入的水印



下面是我们分别对图像进行不同的N\*N分块来篡改

64x64



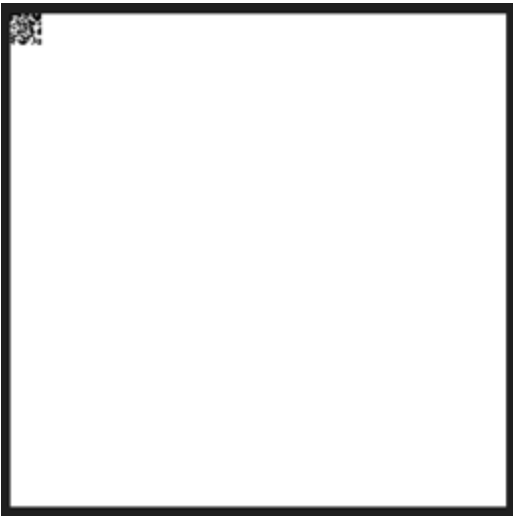
提取篡改后的水印



16x16



提取篡改后的水印



被加入了水印的图像中, 篡改某个区域的像素值, 提取出的水印上都能直观的显示出被篡改的区域, 由此可见得wong脆弱水印算法的敏感性。

4.对图片1嵌入水印后的图片添加椒盐噪声点，椒噪声密度为0.001，盐噪声密度为0.001，提取水印，观察水印的变化。

下面分别是添加椒盐噪点前的图片和添加椒盐噪点后的图片，以及添加椒盐噪点前提取的水印和添加椒盐噪点后的水印

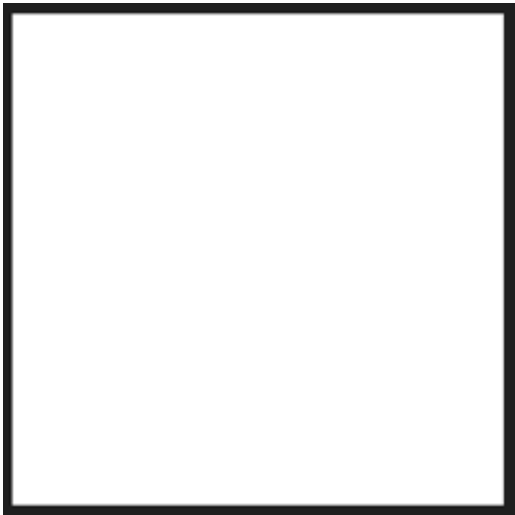
添加椒盐噪点前的图片



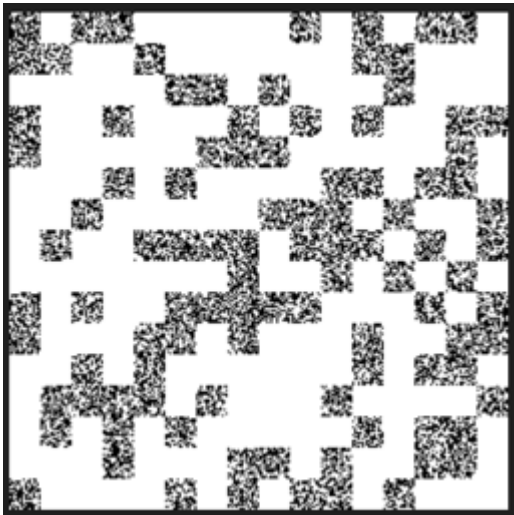
椒噪声密度为0.001时



添加椒盐噪点前提取的水印



提取椒盐密度为0.001的水印



添加椒盐噪点前的图片



椒噪声密度为0.0001时





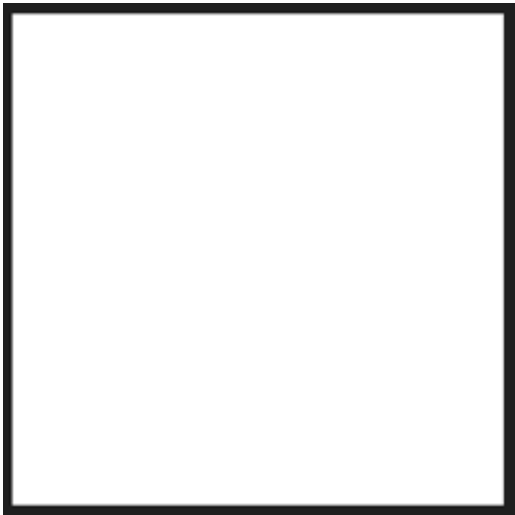
添加椒盐噪点前的图片



椒噪声密度为0.0001时



添加椒盐噪点前提取的水印



提取椒盐密度为0.0001的水印



添加椒盐噪点前的图片



椒噪声密度为0.00001时



添加椒盐噪点前提取的水印

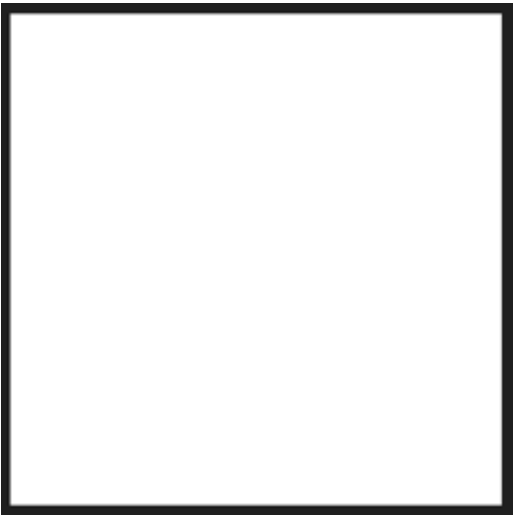


提取椒盐密度为0.00001的水印

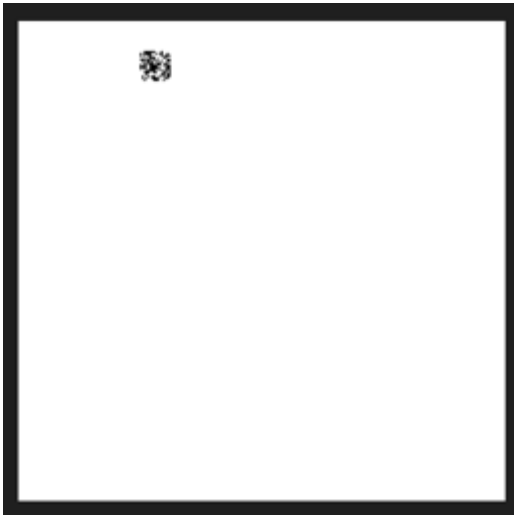




添加椒盐噪点前提取的水印



提取椒盐密度为0.00001的水印



我们组只进行了椒噪声密度为0.001，0.0001甚至为0.00001的密度（盐噪声密度与椒噪声一样）的攻击，提取出的水印上都能直观的显示出被篡改的区域，如此完全可以反映出我们的wong脆弱水印算法在篡改攻击面前的敏感性.

5.对图片1嵌入水印后的图片添加高斯噪声，提取水印，观察图片以及提取的水印的变化。

添加高斯噪点前的图像



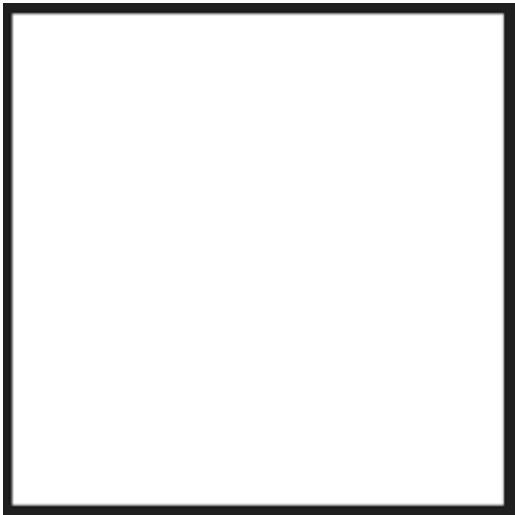
添加高斯噪点后的图像



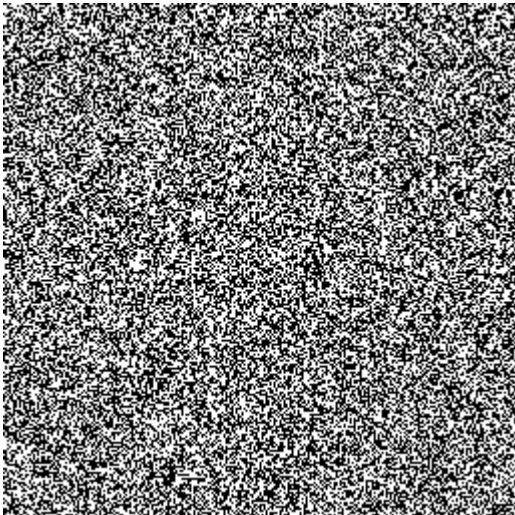
添加高斯噪点前提取的图像水印

添加高斯噪点后提取的图像水印

添加高斯噪点前提取的图像水印



添加高斯噪点后提取的图像水印



很显然，wong脆弱水印算法敏感性极强，这也正是wong脆弱水印算法的优点。通过与前面添加椒盐噪声进行对比，我们可以发现椒盐噪声只是影响了图像的局部的一些区域，而高斯噪声则是影响了整个图像。

6.对图片进行特定的区域复制粘贴攻击，看看wong水印算法能否抵挡此攻击。

接下来是对图片1进行区域复制黏贴攻击

上面的区域复制黏贴攻击是一种假设，假设对方知道了我使用的是wong脆弱水印算法，且掌握了我如何进行划分小块，例如在我们组的代码中就使用了16x16的分块方法。这个改进的灵感来源于曹刚老师在上数字取证的时候对图像区域复制粘贴型篡改取证算法的讲解。

原理是：假设我把原图像某一个块复制到原图像另一个块，由于块跟块之间是独立的，所以如果我们这样做，会导致提取出的水印后，还是无法判别是否被篡改，针对不能抵抗特定的区域复制黏贴攻击，我们选择在wong脆弱水印算法的哈希那一步对于每一个块加入每个块独特的下标*i*，传入哈希函数则为变量index，来解决该问题。

原哈希函数参数为： $H(N,M,Xr)$

现在的哈希函数参数为： $H(N,M,Xr,index)$  index为每个图像块的序号

这样使得即使被篡改这进行区域复制粘贴攻击后，由于下标的不同，会导致哈希后pr的不同，进而导致后面一系列结果都会不同，所以我们能够检测出来图片是被攻击过的

原图像

左上角是由其他块复制粘贴的



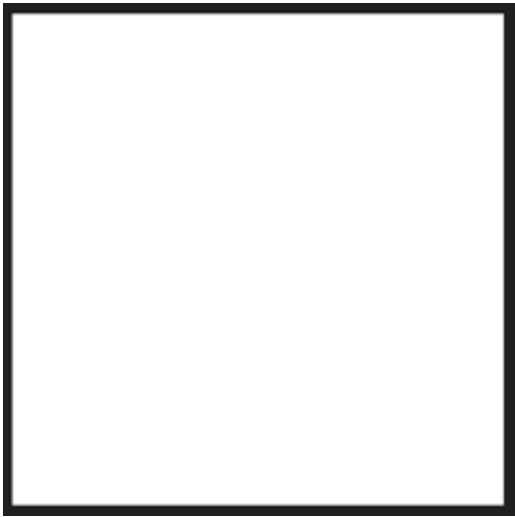
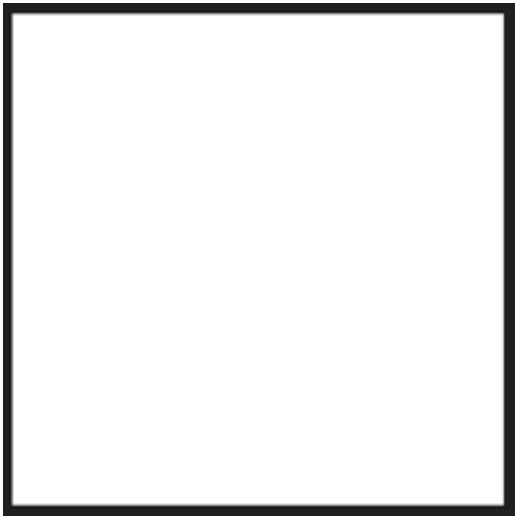
原图像

左上角是由其他块复制粘贴的



原图像水印

攻击过后的图像水印



经过分析与原来的图像水印完全一致。 结合我们的推断以及实验结论，可以得出wong脆弱水印算法无法抵挡特定的区域复制粘贴攻击。

## 八. 我们遇到的问题

我们在对wong脆弱水印算法实现的过程中遇到了点问题：

我们将 $W_r$ 通过公钥加密为 $C_r$ ，即将二进制流加密为等长的二进制流。我们听取了老师的建议，将其进行分组，每16个二进制0/1转换为  $0 \sim 65535$  之间的十进制整数，再对整数进行RSA加密，然后将加密后的整数再转换为0/1，解密也同理。

但问题是 RSA算法需要选择的素数  $p$ 和 $q$ ，其乘积 $n$ 必须很接近  $65535$ ，如果 $n$ 大了，则解密出的数字可能会有大于65535 的，这样转换为二进制就会出问题；如果 $n$ 小了，则有部分位于  $n$ 和  $65535$ 之间的数字无法被正确加密，这是由于RSA算法的数学性质的导致的，无法避免

我们选择的素数是  $19 * 3449$ ，其乘积很接近  $65535$ ，但还是会有  $5 / 65535 * 100\% = 0.0076\%$  的像素无法被正常处理。

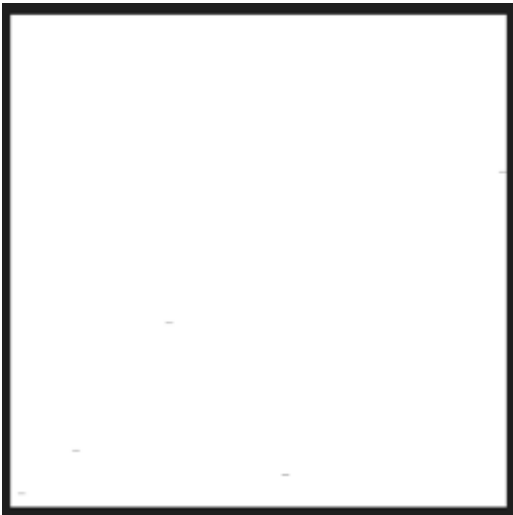
以下是我们选择不同的素数p,q进行水印的提取

$2\ 29 * 263 = 60227$

$251 * 257 = 64507$



$19 * 3449=65531$



九. 分析测试该算法的不足，试提出改进方案

改进的思路:我们要求对数字水印有一定的透明性，由于嵌入过程是基于LSB的，可行的改进方案也是从LSB方面考虑，可以用LSB匹配，JSteg等，来实现更好的透明性。

十. 报告总结

通过此次实验，我们组首先对wong脆弱水印算法进行了实现和改进，在这个过程中加深了对wong脆弱水印算法的理解，同时对对于RSA加密二进制流的问题产生了新的认识与理解，接下来我们组通过对wong脆弱水印算法进行多种实验，验证了wong脆弱水印算法的敏感性极高，定位能力取决于分块大小，原有的方案无法抵抗特定的区域复制黏贴攻击，在卡方检测上会表现出明显的LSB嵌入的特征。我们改进后的算法，能够抵抗特定的区域复制黏贴攻击，收获颇丰。