



南昌大学实验报告

学生姓名： 丁俊 学 号： 8003119100 专业班级： 信息安全 193 班
实验类型： ☐ 验证 ☐ 综合 ☐ 设计 ☐ 创新 实验日期： 2022.4.29 实验成绩：

一、实验项目名称

最不重要位替换和基于 DCT 变换的水印攻击实验

二、实验目的

通过对 LSB 和基于 DCT 变换两种不同算法的嵌入数字水印进行攻击，然后提取出水印和初始水印进行比较，对比 LSB 算法和基于 DCT 变换的水印算法的性能差异。本实验采用 StirMark 软件进行水印攻击评测。

三、实验基本原理

1、最不重要位替换（LSB）算法原理：

LSB 算法利用了数字图像处理中位平面的原理，即改变图像的最低位的信息，对图像信息产生的影响非常小。人眼的视觉感知系统往往不能察觉。以一幅 256 灰度的图像为例，256 灰度共需要 8 个位来表示，但其中每一个位的作用是不一样的，越高位对影像的影响越大，反之越低的位影响越小，甚至不能感知。

LSB 数字水印算法按照上文介绍的四种数字水印分类方法分别属于：鲁棒性数字水印、图像数字水印、空域数字水印、不可见数字水印。

缺点： 改变最低有效位数据，数字图像进行数学变换等攻击方式使得嵌有数字水印的图片很容易受到攻击，缺乏鲁棒性

优点： 实现简单，隐藏量大。

LSB 水印算法实现：

LSB 算法实现较为简单，首先，需要考虑嵌入的数字水印的数据量，如果嵌入最低的 1 位，则可以嵌入的信息量是原始图像信息量的 $1/8$ ，如果适用最低两位则可以嵌入的信息量是 $1/4$ 。但是嵌入的数字水印的信息量越大，同时对图像的视觉效果影响也越大。在这里要嵌入一个二值的图像。然后，适当调整数字水印图像的大小和比特位数，以适应数字水印图像数据量的要求。最后，对原始图像中要使用的最低位置 0，再将数字水印数据放入原始图像的最低位即可。下面通过 MATLAB 2010b2 实现这一算法

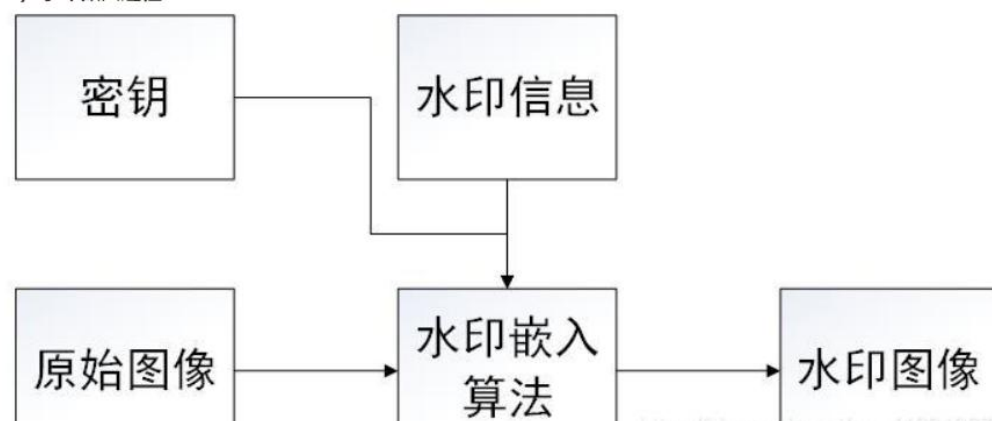
这里选用一幅 256×256 像素，数字水印用“全屏水印”的字样的二值图像。

置 0 的方法是调用模 2 函数 $\text{mod}(a, 2)$ ，将得到的数值与原水印相减，从而得到最低位为 0 的图片。（使用两位最低有效位的话则用模 4 函数 $\text{mod}(a, 4)$ ）

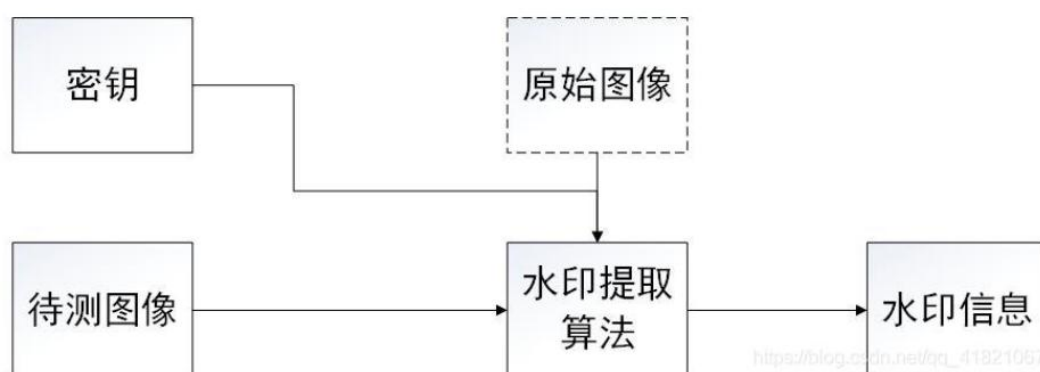
然后相减)。

因为这里加入了噪声干扰,所以对水印和原始图片进行了重编码,首先将原图片扩大两倍,并且使用两位最低有效位然后图片相加得到加入水印的图像。然后对图片加入噪声得到输出图像。

水印嵌入过程流程图:



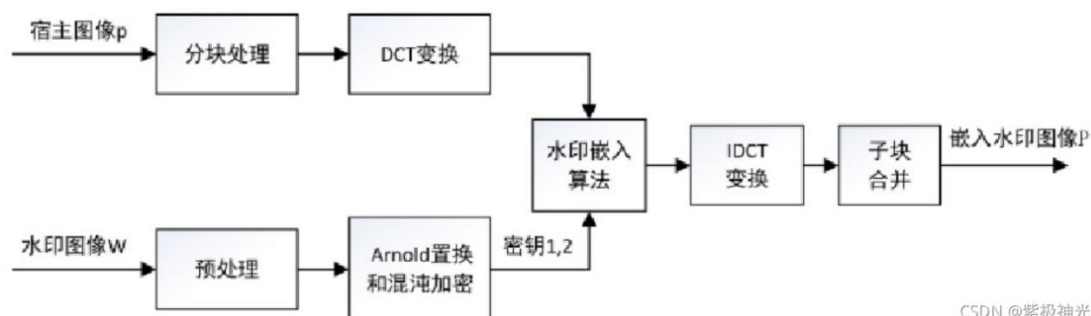
水印提取流程图



2、基于 DCT 变换

目前,基于 DCT 域的水印方法已经成为数字水印算法研究的热点,它的核心思想就是通过离散傅立叶变换对图像块进行处理后,再选择变换域中的一些系数值依据一定规则来嵌入水印。

由于图像块中 DCT 系数频带分布由左上角的直流分量 DC 往下对应的系数频率由低频升至高频,因此在不影响原图质量的前提下,可将水印信息根据能量大小嵌入相应系数频带中。通过图像块量化与水印嵌入结合的处理方法将水印信息均匀分布在图像的整个空间域,在图像裁剪和滤波方面,变换域的水印比在空间域的更能表现出一定的鲁棒性。



二维 DCT 变换

$$F(u, v) = c(u)c(v) \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i, j) \cos \left[\frac{(i+0.5)\pi}{N} u \right] \cos \left[\frac{(j+0.5)\pi}{N} v \right]$$
$$c(u) = \begin{cases} \sqrt{\frac{1}{N}}, & u = 0 \\ \sqrt{\frac{2}{N}}, & u \neq 0 \end{cases}$$

其中， $f(i)$ 为原始的信号， $F(u)$ 是 DCT 变换后的系数， N 为原始信号的点数， $c(u)$ 可以认为是一个补偿系数，可以使 DCT 变换矩阵为正交矩阵。

二维 DCT 逆变换

$$F(u, v) = c(u)c(v) \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i, j) \cos \left[\frac{(i+0.5)\pi}{N} u \right] \cos \left[\frac{(j+0.5)\pi}{N} v \right]$$
$$c(u) = \begin{cases} \sqrt{\frac{1}{N}}, & u = 0 \\ \sqrt{\frac{2}{N}}, & u \neq 0 \end{cases}$$

对二维图像进行离散余弦变换

由以上对二维离散余弦变换的定义以及公式可知，求二维图像的离散余弦变换要进行如下步骤：

1. 获得图像的二维数据矩阵 $f(x, y)$;
2. 求离散余弦变换的系数矩阵 $[A]$;
3. 求系数矩阵对应的转置矩阵 $[A]^T$;
4. 根据公式 (7) $[F(u, v)] = [A][f(x, y)][A]^T$ 计算离散余弦变换;

3、数字水印攻击

对含有水印图像的攻击方法分为有意义攻击和无意攻击两大类，水印必须对一些无意的攻击具有鲁棒性，也就是对那些能保持感官相似性的数字处理操作具有鲁棒性，常见的操作有：剪切、亮度和对比度的修改、增强、模糊和其他滤波算法、放大、缩小和旋转、有损压缩、加噪声等；通常假定在检测水印时不能获得原始产品。下面是有意攻击的一些分类：

- (1) 伪造水印的抽取
- (2) 伪造的肯定检测
- (3) 统计学上的水印抽取
- (4) 多重水印

本次实验采用的攻击是 AFFINE_1(仿射变换)、JPEG_50(压缩攻击)、LATESTNRDDIST_1、MEDIAN_5(中值滤波)、PSNR_20(噪声攻击)五种攻击测试，利用的软件是 StirMark。

四、主要仪器设备及耗材

Window10、pycharm、StriMark 软件、BMP 图像

五、实验步骤

通过 LSB 算法和基于 DCT 变换的数字水印算法嵌入数字水印,然后利用 StriMark 软件实现对数字水印的攻击,最后提取出数字水印,与最初的水印进行比较。见第六步实验数据以处理结果。

LSB 嵌入和提取水印代码

```
1. clear all;
2. start_time=cputime;
3. cover_object=imresize(imread('cam.png'),[256,256]);
4. [row,col]=size(cover_object);
5. M=row;
6. N=col;
7. MN=col*row;
8. %%%读入原始水印%%%%
9. m=imresize(imread('nd.png'),[256,256]);%!!!
10. for i=1:256
11.     for j=1:256
12.         if double(m(i,j))==0
13.             w(i,j)=0;
14.         else
15.             w(i,j)=1;
16.         end
17.     end
18. end
19.
20. %%%水印信息的嵌入%%%%
21. s=cover_object;
22. for i=1:256
23.     for j=1:256
```

```

24.         s(i,j)=bitset(cover_object(i,j),1,w(i,j)); %%weizhi=3,数越大,
    水印越明显%%
25.         end                                     %%数越小, 水印越模糊
    (最低有效位的来历)
26.     end
27.     imwrite(s,'lsb_watermarkedx.bmp');
28.
29.
30.     figure(1)
31.     subplot(1,2,1);
32.     imshow(s,[]);
33.     title('嵌入水印图像')
34.     subplot(1,2,2);
35.     imshow(cover_object,[])
36.     title('原图像')
37.
38.     %计算 PSNR
39.     A=s;X=cover_object;
40.     [height width]=size(X);
41.     X=double(X);
42.     A=double(A);
43.     sigma1=0;
44.     for i=1:height
45.         for j=1:width
46.             sigma1=sigma1+(X(i,j)-A(i,j))^2;
47.         end
48.     end
49.     mse=(sigma1/(height*width));    %均方误差
50.     psnr=10*log10((255^2)/mse)
51.
52.     %tiqv
53.     file_name='hb.png';
54.     orig_watermark=imread(file_name);
55.     for i=1:256
56.         for j=1:256
57.             mm(i,j)=bitget(uint8(s(i,j)),1);
58.             if double(mm(i,j))==1
59.                 mm(i,j)=255;
60.             else
61.                 mm(i,j)=0;
62.             end
63.         end
64.     end
65.     %err=sum(sum(xor(m,mm)))/(256*256),

```

```

66.
67. %显示运行时间
68. elapsed_time=cputime-start_time,
69.
70. figure(3)
71. subplot(1,2,1);
72. imshow(uint8(mm));
73. title('提取水印图像');
74. subplot(1,2,2);
75. imshow(orig_watermark,[])
76. title('原始水印')

```

DCT 嵌入和提取水印代码

嵌入

```

1. %Project: Threshold-Based Correlation in DCT mid-band
2. % Uses two PN sequences; one for a "0" and another for
   a "1"
3. %水印嵌入
4.
5. clear all;
6.
7. % 保存开始时间
8. start_time=cputime;
9.
10. k=30; % 设置嵌入强度
11. blocksize=8; % 设置块的大小
12.
13. midband=[ 0,0,0,1,1,1,1,0; % defines the mid-band frequencies of an 8x8 dct
14.           0,0,1,1,1,1,0,0;
15.           0,1,1,1,1,0,0,0;
16.           1,1,1,1,0,0,0,0;
17.           1,1,1,0,0,0,0,0;
18.           1,1,0,0,0,0,0,0;
19.           1,0,0,0,0,0,0,0;
20.           0,0,0,0,0,0,0,0 ];
21.
22. % 读入原始图像
23. file_name='cam.png';
24. cover_object=double(imread(file_name));
25.
26. % 原始图像的行数与列数
27. Mc=size(cover_object,1); %原图的行数
28. Nc=size(cover_object,2); %原图的列数

```

```

29.
30. % 确定可嵌入的最大信息量
31. max_message=Mc*Nc/(blocksize^2);
32.
33. % 读入水印图像
34. file_name='nd.png';
35. message=double(imread(file_name));
36. %水印图像的行数与列数
37. Mm=size(message,1)           %水印图像的行数
38. Nm=size(message,2)           %水印图像的列数
39.
40. figure(1)
41. imshow(message,[]);
42. title('水印');
43.
44. % 将水印图像矩阵转换为 0, 1 组成的向量
45. message=reshape(message,1,Mm*Nm);
46.
47. % 检查水印信息是否过大
48. if (length(message) > max_message)
49.     error('水印太大')
50. end
51.
52. % 将 message_vector 置为全 1 向量, 并将 message 写入
53. message_vector=ones(1,max_message);
54. message_vector(1:length(message))=message;
55.
56. % 将 cover_object 写入 watermarked image
57. watermarked_image=cover_object;
58.
59. key=1100;
60. % 重置随机数发生器状态为 key
61. rand('state',key),
62.
63. % 生成伪随机数
64. pn_sequence_zero=round(2*(rand(1,sum(sum(midband)))-0.5));
65.
66. % 图像分块并嵌入
67. x=1;
68. y=1;
69. for (kk = 1:length(message_vector))
70.     % 分块进行 DCT 变换
71.     dct_block=dct2(cover_object(y:y+blocksize-1,x:x+blocksize-1));

```

```

72.
73.         % 如果 message_vector==0 并且 midband==1 , 那么 嵌
    入 pn_sequence_zero
74.     ll=1;
75.     if (message_vector(kk)==0)
76.         for ii=1:blocksize
77.             for jj=1:blocksize
78.                 if (midband(jj,ii)==1)
79.                     dct_block(jj,ii)=dct_block(jj,ii)+k*pn_sequen
    ce_zero(ll);
80.                     ll=ll+1;
81.                 end
82.             end
83.         end
84.     end
85.
86.     % 逆 DCT 变换
87.     watermarked_image(y:y+blocksize-1,x:x+blocksize-1)=idct2(dct_
    block);
88.
89.     % 移动到下一块
90.     if (x+blocksize) >= Nc
91.         x=1;
92.         y=y+blocksize;
93.     else
94.         x=x+blocksize;
95.     end
96. end
97.
98. % 转 换 为 uint8 并 将 watermarked_image_uint8 写 入
    watermarked_image_uint8.bmp
99. watermarked_image_uint8=uint8(watermarked_image);
100. imwrite(watermarked_image_uint8,'watermarked_image_uint8.bmp','bm
    p');
101.
102. % 显示运行时间
103. elapsed_time=cputime-start_time,
104.
105. % 计算 psnr
106. A=watermarked_image_uint8;X=cover_object;
107. [height width]=size(X);
108. X=double(X);
109. A=double(A);
110. sigma1=0;

```



```

111. for i=1:height
112. for j=1:width
113. sigma1=sigma1+(X(i,j)-A(i,j))^2;
114. end
115. end
116. mse=(sigma1/(height*width)); %均方误差
117. psnr=10*log10((255^2)/mse)
118.
119. % 显示嵌入水印图象与原始图象
120. figure(2)
121. subplot(1,2,1);
122. imshow(watermarked_image_uint8,[])
123. title('嵌入水印图像')
124. subplot(1,2,2)
125. imshow(cover_object,[]);
126. title('原始图像');

```

提取

```

1. %Project: Threshold-Based Correlation in DCT mid-band
2. % Uses two PN sequences; one for a "0" and another for
   a "1"
3. % 水印提取
4.
5. clear all;
6.
7. % 保存开始时间
8. start_time=cputime;
9.
10. blocksize=8; % 设置块的大小
11. midband=[ 0,0,0,1,1,1,1,0; % defines the mid-band frequencies
   of an 8x8 dct
12.          0,0,1,1,1,1,0,0;
13.          0,1,1,1,1,0,0,0;
14.          1,1,1,1,0,0,0,0;
15.          1,1,1,0,0,0,0,0;
16.          1,1,0,0,0,0,0,0;
17.          1,0,0,0,0,0,0,0;
18.          0,0,0,0,0,0,0,0 ];
19.
20. % 读入嵌入水印图像
21. file_name='watermarked_image_uint8.bmp';
22. watermarked_image=double(imread(file_name));

```

```

23.
24. % 嵌入水印图像的行列数
25. Mw=size(watermarked_image,1);          %行数
26. Nw=size(watermarked_image,2);          %列数
27.
28. % 确定最大嵌入信息量
29. max_message=Mw*Nw/(blocksize^2);
30.
31. % 读入水印
32. file_name='nd.png';
33. orig_watermark=double(imread(file_name));
34.
35. % 水印的行列数
36. Mo=size(orig_watermark,1); %行数
37. No=size(orig_watermark,2); %列数
38.
39.
40. key=1100;
41.
42. % 置随机数发生器的状态为 key
43. rand('state',key);
44.
45. % 产生伪随机序列
46. pn_sequence_zero=round(2*(rand(1,sum(sum(midband)))-0.5));
47.
48. % process the image in blocks
49. x=1;
50. y=1;
51. for (kk = 1:max_message)
52.
53.     % 对块进行 DCT 变换
54.     dct_block=dct2(watermarked_image(y:y+blocksize-1,x:x+blocksize-1));
55.
56.     % extract the middle band coefficients
57.     ll=1;
58.     for ii=1:blocksize
59.         for jj=1:blocksize
60.             if (midband(jj,ii)==1)
61.                 sequence(ll)=dct_block(jj,ii);
62.                 ll=ll+1;
63.             end
64.         end
65.     end

```

```
66.
67.     % 计算相关性系数
68.     correlation(kk)=corr2(pn_sequence_zero,sequence);
69.
70.     % 移动到下一块
71.     if (x+blocksize) >= Nw
72.         x=1;
73.         y=y+blocksize;
74.     else
75.         x=x+blocksize;
76.     end
77. end
78.
79. % 如果相关性系数大于平均值那么置 '0', 反之置 '1'
80. for (kk=1:max_message)
81.     if (correlation(kk) > mean(correlation(1:max_message)))
82.         message_vector(kk)=0;
83.     else
84.         message_vector(kk)=1;
85.     end
86. end
87.
88. % 重新排列提取水印
89.
90. message=reshape(message_vector(1:Mo*No),Mo,No);
91.
92. % 显示运行时间
93. elapsed_time=cputime-start_time,
94.
95. % 显示提取水印与原始水印
96.
97. figure(3)
98. imshow(message,[])
99. title('提取水印')
100. figure(4)
101. imshow(orig_watermark,[])
102. title('原始水印');
```

六、实验数据及处理结果

1、在原始图片 1.bmp 上运行 LSB 嵌入算法，得到含有隐藏信息的图像 Hide_image1.bmp；在原始图像 2.bmp 上运行 DCT 嵌入算法，得到含有隐藏信息的图像 Hide_image2.bmp；



图 1 LSB 嵌入算法



图 2 DCT 嵌入算法

2、使用 StirMark 工具，进行攻击测试：

①将待攻击的图像放置在文件夹 **Media\Input\Images\Set1** 中。

> StirMarkBenchmark_4_0_129 > Media > Input > Images > Set1

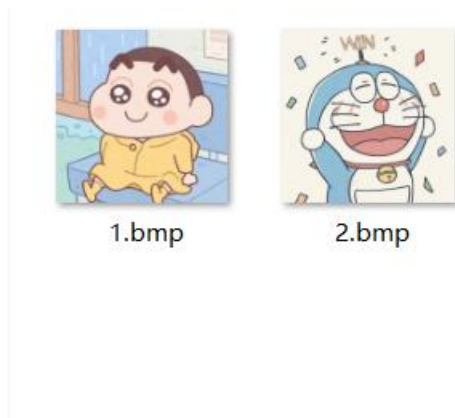


图 3 待攻击图像

②双击 **Bin\Benchmark\StirMark Benchmark** 应用程序，程序会自动运行。
程序运行时，会有下面的界面，显示的是攻击测试类型：

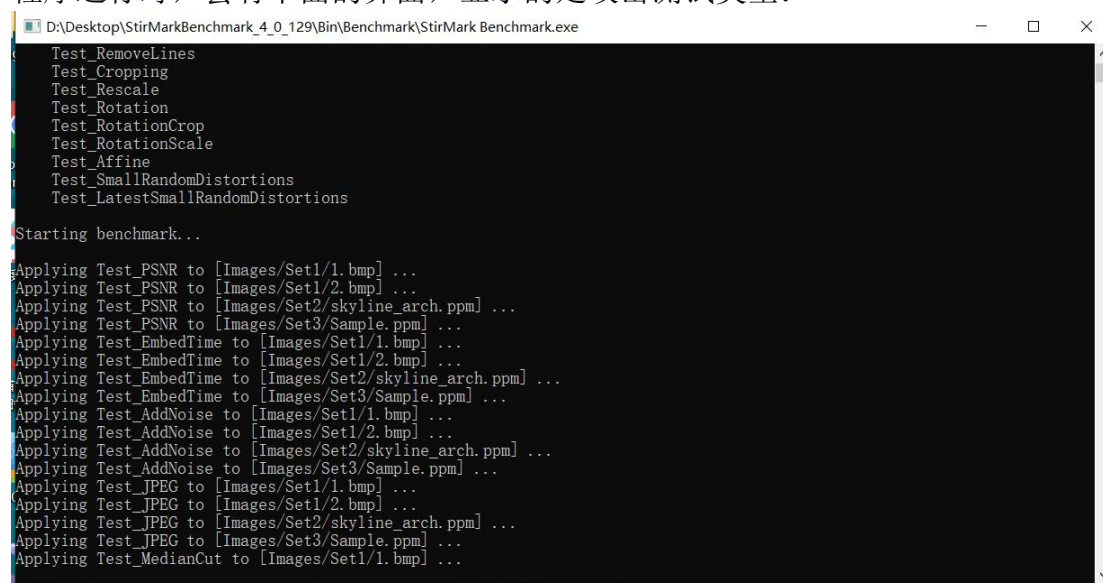


图 4 攻击测试类型

若想修改攻击测试的参数，可以在 **Profiles\SMBsettings** 里进行修改：

```
SMBsettings.ini - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

;-----
; Definition of the parameters for the tests
;-----

[Test_PSNR]
; Strength of the watermark embedding
start=0
end=100
step=10

[Test_EmbedTime]
; Number of embeddings with random key per media
; This is used to compute the average embedding and extraction time
list=5

[Test_AddNoise]
; Noise level is normalised from 0 to 100
; 0 gives identity and 100 a completely random image
start=0;
end=100;
step=20;
```

攻击测试后的图像放置在 Media\Output\Images\Set1 中，如图：

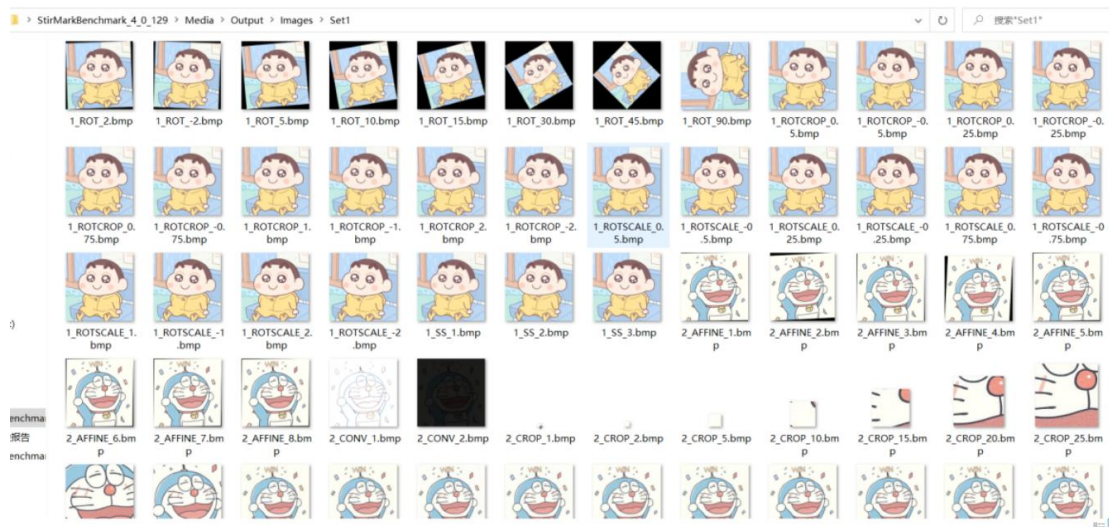



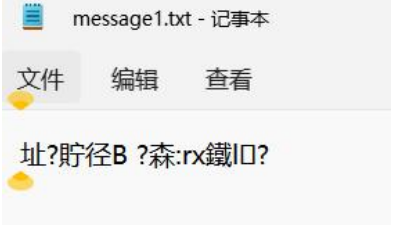

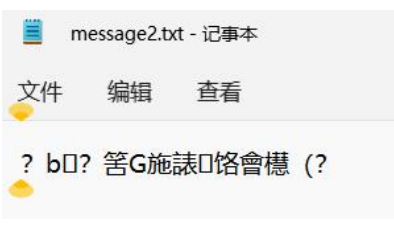


图 6 攻击测试后的图像







1、选用五种攻击测试的图像，分别对应使用 LSB 提取算法和 DCT 提取算法提取信息，比较提取信息是否与隐写信息相符。
隐写信息 hidden.txt:



图 7 隐写信息

表 1 提取信息

攻击测试类型及参数	算 法 类型	攻击测试后的图像	提取信息
AFFINE_1	LSB		
	DCT		
JPEG_50	LSB		
	DCT		

LATESTNRDDIST_1	LSB		<div>message3.txt - 记事本</div> <div>文件 编辑 查看</div> <div>la諉雌失G柒Z#陵\$?顔?</div>
	DCT		<div>message8.txt - 记事本</div> <div>文件 编辑 查看</div> <div>屢口虐罅OK</div>
MEDIAN_5	LSB		<div>message4.txt - 记事本</div> <div>文件 编辑 查看</div> <div>卜瑚口@·翩? 4? 里纓蝸</div>
	DCT		<div>message9.txt - 记事本</div> <div>文件 编辑 查看</div> <div>? 口敷baiH尅</div>
PSNR_20	LSB		<div>message5.txt - 记事本</div> <div>文件 编辑 查看</div> <div>duA节口f映E口H?</div>
	DCT		<div>Message6 (3).txt - 记事本</div> <div>文件 编辑 查看</div> <div>β003119100 丁俊</div>

由上表可知，LSB 提取算法在五种攻击测试中均无法成功提取信息，DCT 提取算法在前五种攻击测试中均无法成功提取信息，在参数为 20 的 PSNR 攻击测

试中提取出隐藏信息的数字部分。

由于时间有限，可推测，在本次实验中所使用的 LSB 算法和 DCT 算法都无法抵抗 AFFINE、JPEG、LATESTRNDDIST、MEDIAN、PSNR 攻击测试。如果能进行改进，或许可以提高其鲁棒性和安全性。

七、思考讨论题或体会或对改进实验的建议

经过本次实验，我熟悉了 StirMark 工具的使用方法，了解了 StirMark 是一个在数字水印研究领域非常有名的测试工具，能够检测水印算法的鲁棒性。此次实验通过之前学习的针对 LSB 算法和 DCT 算法使用 StirMark 工具对隐写图像进行了攻击测试，取用其中五种攻击策略对应图像提取隐写信息，了解了这两种不同隐写算法对不同攻击的抵抗能力。

八、参考资料

信息隐藏技术