



信息隐藏实验报告

实验(二)：扩频水印和 Monte-Carlo 仿真

姓 名：_____

学 号：_____

同组成员：_____

专 业：_____

二〇二二年十二月

第一章 总述

1. 实验分工与完成情况

实验分工如下表所示。

(表 1 实验分工与完成情况)

任务点	内容	完成情况	主要贡献者	贡献率
1. 图像的 DCT 系数上利用加性扩频水印嵌入信息				
1-1	在图像的 DCT 系数上嵌入信息，宿主图像为常见的 512×512 大小的 Lena 灰度图，嵌入信息为一二进制 Logo 图(即该图只有两种颜色:黑和白)，该图大小为 32×32，共 1024 个 Bit。这些图使用课件中提供的图片。 在图像的 DCT 系数上利用加性扩频水印嵌入信息。	√		50%
1-2	利用线性相关器解码，统计一下总错误率。	√		
1-3	实验一下不同的嵌入强度对作品知觉效果和解码效果的影响，并且绘制实验曲线。	√		
1-4	水印作品受到攻击：使用图像处理软件等加入噪声、改变对比度、JPEG 压缩等攻击，然后看看解码的效果（实验曲线）。	√		
2. Monte-Carlo 仿真验证 ASS 水印检测理论结果的正确性				
2-1	无攻击情况下：宿主信号 X 服从期望为 GGD 分布(中心为 0，标准偏差 STD=10)，嵌入强度 a=1.8，N=1000。实验要求达到的精度 1E-6。 （1）不同形状参数下的性能：绘制 ROC 曲线（包括形状参数分别为 c=2.0、1.0、0.5 时系统理论性能和实验性能）； （2）不同嵌入强度下的性能：绘制	√		50%

	ROC 曲线 (强度 $a=1.5$, $a=1.8$ 时的理论和实验性能 ($c=2.0$))。			
2-2	受到攻击情况下: 宿主信号 X 服从期望为高斯分布 (中心为 0, 标准偏差 $STD=10$), 嵌入强度 $a=1.8$, $N=1000$ 。实验要求达到的精度 $1E-6$ 。 (1) 噪声期望为 0, STD 分别为 2.5、5; (2) 绘制此时的 ROC 曲线 (未受攻击时的理论性能, 受到攻击时的理论和实验性能)。	√		

2. 符号说明

符号	含义
μ	均值 (中心)
σ	标准差 (STD)
c	GGD 分布的形状参数
β	GGD 分布的尺度参数
β_{Gamma}	指数分布的尺度参数
X_i	原始 (宿主) 作品
S_i	加入水印后的作品
Y_i	受到攻击后的作品
W_i	二元水印序列
$L(S)$	决策统计量
Ψ	决策阈值
$_{\text{the}} P_{\text{fa}}$	理论 P_{fa}
$_{\text{the}} P_{\text{m}}$	理论 P_{m}
$_{\text{exp}} P_{\text{fa}}$	实验 P_{fa}
$_{\text{exp}} P_{\text{m}}$	实验 P_{m}

3. 作业提交的文件结构

```
---2053182-王润霖-实验(二)报告.pdf
---任务 1: 水印嵌入源代码及图像数据
    ---添加噪声、改变对比度攻击的图像和代码（等效于 Photoshop 的工作）
        ---output
        ---AddNoise.m
        ---ChangeContrast.m
        ---CompressionJpeg.m
        ---lena_embed.bmp
    ---bitmap_image.cpp
    ---bitmap_image.h
    ---DCT.cpp
    ---DCT.h
    ---LENA.BMP
    ---main.cpp
    ---ss_watermark.cpp
    ---ss_watermark.h
    ---tj-logo.bmp
---任务 2: Monte-Carlo 仿真实验
    ---A 无攻击-a1.5-1.8-输出文件
        ---nonattack_exp_pm_1.5.txt
        ---nonattack_exp_pm_1.8.txt
        ---nonattack_the_pfa.txt
        ---nonattack_the_pm_1.5.txt
        ---nonattack_the_pm_1.8.txt
    ---A 无攻击-c0.5-1.0-2.0-输出文件
        ---nonattack_exp_pm_1.8(c0.5).txt
        ---nonattack_exp_pm_1.8(c1.0).txt
        ---nonattack_exp_pm_1.8(c2.0).txt
        ---nonattack_the_pfa.txt
        ---nonattack_the_pm_1.8(c0.5).txt
        ---nonattack_the_pm_1.8(c1.0).txt
        ---nonattack_the_pm_1.8(c2.0).txt
    ---B 有攻击-输出文件
        ---attack_exp_pm_2.5.txt
        ---attack_exp_pm_5.0.txt
        ---attack_the_pfa.txt
        ---attack_the_pm_2.5.txt
        ---attack_the_pm_5.0.txt
        ---nonattack_exp_pm_1.8.txt
        ---nonattack_the_pfa.txt
        ---nonattack_the_pm_1.8.txt
    ---embedding.cpp
    ---embedding.h
    ---embedding.cpp
    ---embedding.h
    ---monte_carlo.cpp
```

```
---random_Gaussian.cpp  
---random_Gaussian.h
```

第二章 Monte-Carlo 仿真

第一部分 相关内容介绍（背景知识）

1.1 Monte-Carlo 仿真

Monte-Carlo 仿真方法又称统计试验法，它是一种采用统计抽样理论近似地求解数学、物理及工程问题的方法。它解决问题的基本思想是，首先建立与描述该问题有相似性的概率模型，然后对模型进行随机模拟或统计抽样，再利用所得的结果求出特征量的统计值作为原问题的近似解，并对解的精度作出某些估计。Monte-Carlo 仿真方法的主要理论基础是概率论中的大数定律，其主要手段为随机变量的抽样分析。

Monte-Carlo 仿真就是通过随机数发生器产生原始数据（模拟宿主数据 X ），而水印的嵌入和检测就是在这些数据中进行的。因此，用 Monte-Carlo 仿真可以验证理论结果的正确性。

必须通过大量的数据来进行达到足够的精度。在这次实验中，要求达到 $1e-6$ 的精度，因此，我们需要生成 10^6 组数据。

1.2 常见分布函数

（1）高斯分布

高斯分布随机分布中最常见的一种，又称为正态分布。正态曲线呈钟型，两头低，中间高，左右对称因其曲线呈钟形，因此人们又经常称之为钟形曲线。若随机变量 X 服从一个数学期望为 μ 、方差为 σ^2 的正态分布，记为 $N(\mu, \sigma^2)$ 。其概率密度函数为正态分布的期望值 μ 决定了其位置，其标准差 σ 决定了分布的幅度。当 $\mu = 0, \sigma = 1$ 时的正态分布是标准正态分布。

（2）广义高斯分布

广义高斯分布 (GGD) 被经常使用与图像/视频信号的统计分析，其形状参数常被用为图像的特征进行分类或回归。如在图像质量评价任务中，Anish Mittal 等人提出的 BRISQUE 模型利用 GGD 拟合归一化后图像 (MSCN) 的分布，利用 GGD 参数作为一组特征。

1.3 扩频水印嵌入技术

扩频水印是一种直接将加性 (additive) 或乘性 (multiplicative) 水印信号嵌入到宿主信号中的技术。

在本实验中，我们使用加性水印信号。

1.4 高斯噪声攻击

高斯噪声攻击就是在水印作品上加入服从高斯分布的噪声，假设 $V = (V_1, V_2, \dots, V_N)$ 为攻击噪声，当然为了简化起见， V_i 服从独立等同分布的高斯分布，期望为 0，方差为 σ^2 ，且 V 独立于 X 。

第二部分 实验目的、内容和原理

2.1 无攻击情况下的情形

2.1.1 实验目的

通过 Monte-Carlo 仿真实验，模拟在宿主数据中的水印嵌入和检测情形；同时，比较不同形状参数 c 、不同嵌入强度 a 情形下的系统理论性能和实验性能，绘制 ROC 曲线。

2.1.2 实验内容

无攻击情况下：宿主信号 X 服从期望为 GGD 分布（中心为 0，标准偏差 $STD=10$ ），嵌入强度 $a=1.8$ ， $N=1000$ 。实验要求达到的精度 $1E-6$ 。

(1) 不同形状参数下的性能：绘制 ROC 曲线（包括形状参数分别为 $c=2.0$ 、 1.0 、 0.5 时系统理论性能和实验性能）；

(2) 不同嵌入强度下的性能：绘制 ROC 曲线（强度 $a=1.5$ ， $a=1.8$ 时的理论和实验性能 ($c=2.0$)）。

2.1.3 实验原理

2.1.3.1 生成 $c=2.0$ ；中心=0；STD=10 的 GGD 分布数据 X_i

根据题意，我们需要生成形状参数为 2.0、1.0、0.5 共三种情况下的宿主数据 X_i ；我们知道，当形状参数为 2.0 时，GGD 分布实际等同于高斯分布；当形状参数为 1.0、0.5 时，可以根据 Gamma 分布产生随机数。

利用 Box-Muller 方法，产生高斯分布随机数：

Box-Muller，一般是要得到服从正态分布的随机数，基本思想是先得到服从均匀分布的随机数再将服从均匀分布的随机数转变为服从正态分布。

需要选取两个服从 0-1 均匀分布的随机数，两个随机数分别为 U_1 、 U_2 ；通过 Box-Muller 变换公式，使得 X 、 Y 满足：

$$\begin{aligned} X &= \cos(2\pi U_1) \sqrt{-2 \ln U_2} \\ Y &= \sin(2\pi U_1) \sqrt{-2 \ln U_2} \end{aligned} \quad (1)$$

通过(1)式，得到服从高斯分布 $N(0, 1)$ 的随机数 X 、 Y 。

要使中心=0、STD=10，我们令 $\mu=0$ 、 $\sigma=10$ ：

$$X' = \mu + \sigma X \quad (2)$$

通过(2)式, 得到服从高斯分布 $N(\mu, \sigma^2)$ 的随机数 X' 、 Y' 。

2.1.3.2 生成 $c=1.0$ 、 $c=0.5$; 中心=0; STD=10 的 GGD 分布数据 X_i

利用指数分布, 产生 GGD 分布随机数。

如果 $c = 1.0$, 那么随机变量 E 就是服从形状参数为 1.0 的 Gamma 分布, 而形状参数为 1 的 Gamma 分布就是指数分布。

如果 $c = 0.5$, 那么随机变量 E 的 Gamma 分布的形状参数为 2.0, 尺度参数为 $1/\beta^{0.5}$ 。

产生指数分布的原理如下:

根据产生随机变量的逆变换法; 根据定理: 设 $F(x)$ 是任一连续的分布函数, 如果 $\mu \sim U(0,1)$ [均匀分布] 且 $\eta \sim F(x)$ 。

$$P(\eta \leq x) = P(F^{-1}(\mu) \leq x) = P(\mu \leq F(x)) = F(x) \quad (3)$$

证明: 由于 $\mu \sim U(0,1)$, 则有式子(3); 所以: $\eta \sim F(x)$ 。

$$x = -\beta \ln(\mu) \quad (4)$$

通过逆变换法产生指数分布随机数: 首先产生均匀分布的随机数 $\mu \sim U(0,1)$; 然后由逆变换公式(4)计算 $x = -\beta \ln(\mu)$ 。则 $x \sim E(\beta)$ 。

要使中心=0、STD=10, 我们需要通过给定的 STD, 计算 GGD 分布的尺度参数 β 。计算方法如式(5)所示, 其中, σ_x 即为 STD=10。

$$\beta = \frac{1}{\sigma_x} \left(\frac{\Gamma(3/c)}{\Gamma(1/c)} \right)^{1/2} \quad (5)$$

然后, 根据 GGD 分布的尺度参数 β , 推导出指数分布中的尺度参数 β_{Gamma} , 如式(6)所示。

$$\beta_{\text{Gamma}} = \frac{1}{\beta^c} \quad (6)$$

由此, 我们就可以生成 $c=1.0$ 、 $c=0.5$; 中心=0; STD=10 的 GGD 分布数据 X_i 了。

2.1.3.3 生成数据 W_i

W_i 是一个二元水印序列, 其中, $W_i=+1$ 或者 -1 (实际可以为任何实数)。

W_i 需要满足式(7):

$$\sum_{i=1}^N w_i = 0 \quad (7)$$

也就是说, w 中 $+1$ 和 -1 的个数相等。这里, 为了方便起见, 我们让 W_i 就等于 $+1$ 、 -1 、 $+1$ 、 -1 、……。

2.1.3.4 计算数据 S_i

由于我们使用加性水印嵌入的方式，因此，在嵌入水印时， s_i 可通过式(8)求出。

$$s_i = x_i + a \cdot w_i \quad (8)$$

其中， a 是水印嵌入强度，在实验中，我们将会用到 $a=1.8$ 、 $a=1.5$ 两种水印嵌入强度。

2.1.3.5 计算 Q 函数和 Q 的反函数(Q_inv)

我们知道，Q 函数可以通过 erfc 函数求得：

erfc 函数表达式如式(9)：

$$\text{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^{+\infty} \exp(-t^2/2) dt \quad (9)$$

通过式(10)求得 Q 函数：

$$Q(x) = \frac{1}{2} \text{erfc}\left(\frac{x}{\sqrt{2}}\right) \quad (10)$$

根据题目，给定了 ierfc 的代码，而 Q 的反函数 Q_{inv} 也容易通过 ierfc 函数求得，如式(11)：

$$Q^{-1}(Y) = \sqrt{2} \text{ierfc}(2Y) \quad (11)$$

2.1.3.6 计算阈值和理论 P_m

接下来的总体思路是：先自己定一组理论 P_{fa} 作为自变量；然后用每一个理论 P_{fa} ，都能算出对应的理论 P_m 和阈值；用阈值进行检测，得到实际 P_{fa} 和 P_m ；最后画 P_{fa} （理论）为 x 轴， P_m （理论、实际）为 y 轴的曲线；我们自己设的一个横坐标是对应一个阈值，然后检测 10^6 组数据对于一个阈值的实际 P_m ，得到图像上的一个点。

我们先规定一组理论 P_{fa} ，由于需要 $1e-6$ 的精度，因此，我们要使其在坐标轴上呈现 \log_{10} 分布。

规定 $P_{fa}=1e-6$ 、 $2e-6$ 、 $3e-6$ 、 \dots 、 $9e-6$ 、 $1e-5$ 、 $2e-5$ 、 $3e-5$ 、 \dots 、 $9e-5$ 、 \dots 、 1 ，共 55 个数据点。

由于我们知道式(12)：

$$P_{fa} = Q\left[\frac{\sqrt{N}\psi}{\sigma_x}\right] \quad (12)$$

所以，可以根据式(13)计算阈值：

$$\Psi = \frac{\sigma_x Q^{-1}(P_{fa})}{\sqrt{N}} \quad (13)$$

同样地，我们可以根据式(12)推导出理论 P_m

$$p_m = 1 - Q \left[Q^{-1}(p_{fa}) - \frac{\sqrt{Na}}{\sigma_x} \right] \quad (14)$$

2.1.3.7 水印检测、计算实验 Pm

我们假设检验过程中： H_1 表示作品中存在水印 w 、 H_0 表示作品中不存在水印 w 。

则有式(15)：

$$\begin{aligned} H_1 &\Rightarrow S_i = X_i + a \cdot w_i \\ H_0 &\Rightarrow S_i = X_i \end{aligned} \quad (15)$$

ASS 可以用线性相关器进行检测，其定义为式(16)：

$$\begin{aligned} L(S) &= \frac{1}{N} \sum_{i=1}^N S_i \cdot w_i \\ L(S|H_1) &= \frac{1}{N} \sum_{i=1}^N X_i \cdot w_i + a \\ L(S|H_0) &= \frac{1}{N} \sum_{i=1}^N X_i \cdot w_i \end{aligned} \quad (16)$$

检测器的统计决策过程通常可以表示成式(17)：

$$\begin{aligned} L(S) &> \Psi \Rightarrow H_1 \\ L(S) &< \Psi \Rightarrow H_0 \end{aligned} \quad (17)$$

因此，我们可以通过式(18)求得实验 Pm 的值：

$$\begin{aligned} p_m &= P(L(S) < \Psi | H_1) \\ &= P \left[\frac{1}{N} \sum_{i=1}^N X_i w_i + a < \Psi \right] \end{aligned} \quad (18)$$

2.1.3.8 改变参数绘制 ROC 曲线

根据题目要求，分别求解出在 $a=1.8$ 时 $c=2.0$ 、 $c=1.0$ 、 $c=0.5$ 和在 $c=2.0$ 时 $a=1.8$ 、 $a=1.5$ 条件下的阈值、理论 Pfa、实验 Pfa、理论 Pm、实验 Pm；并按照 \log_{10} 从 $1e-6$ 至 1 的坐标绘制 ROC 曲线，即可完成本节实验。

2.2 有攻击情况下的情形

2.2.1 实验目的

通过 Monte-Carlo 仿真实验，模拟在宿主数据中的水印嵌入并被攻击时的情形；同时，比较不同高斯噪声强度（以标准偏差度量）下的未受攻击时的理论性能、受到攻击时的理论性能和实验性能，绘制 ROC 曲线。

2.2.2 实验内容

受到攻击情况下：宿主信号 X 服从期望为高斯分布(中心为 0，标准偏差

STD=10)，嵌入强度 $a=1.8$ ， $N=1000$ 。实验要求达到的精度 $1E-6$ 。

(1) 噪声期望为 0，STD 分别为 2.5、5；

(2) 绘制此时的 ROC 曲线（未受攻击时的理论性能，受到攻击时的理论和实验性能）。

2.2.3 实验原理

我们只需要在 3.1 节的基础上，继续讨论受到攻击时的运算即可。

2.2.3.1 计算 Y_i 的值

我们用 V_i 表示攻击的高斯噪声，则在受到攻击的情况下，有式(19)：

$$\begin{aligned} H_0 &\Rightarrow Y_i = S_i + V_i = X_i + V_i \\ H_1 &\Rightarrow Y_i = S_i + V_i = X_i + a w_i + V_i \end{aligned} \quad (19)$$

2.2.3.2 计算阈值

通过式(23)可以推导出在有攻击情况下，阈值的表达式，如式(20)所示：

$$\Psi = \frac{\sqrt{\sigma_X^2 + \sigma_V^2} Q^{-1}(\text{the } P_{fa})}{\sqrt{N}} \quad (20)$$

2.2.3.3 计算实验 P_m 和理论 P_m

如果用线性相关器检测，有式(21)：

$$\begin{aligned} H_1 &\Rightarrow L(Y | H_1) = \bar{X} + a + \bar{V} \\ H_0 &\Rightarrow L(Y | H_0) = \bar{X} + \bar{V} \end{aligned} \quad (21)$$

在式(20)中，有式(22)：

$$\begin{aligned} \bar{X} &= \frac{1}{N} \sum_{i=1}^N X_i w_i \\ \bar{V} &= \frac{1}{N} \sum_{i=1}^N V_i w_i \end{aligned} \quad (22)$$

我们通过式(23)计算实验 P_m ：

$$\begin{aligned} \text{exp } P_m &= P[L(Y | H_1) < \Psi] \\ &= P[\bar{X} + \bar{V} + a < \Psi] \end{aligned} \quad (23)$$

通过式(24)计算理论 P_m ：

$$\text{the } P_m = 1 - Q \left[Q^{-1}(\text{the } P_{fa}) - \frac{\sqrt{Na}}{\sqrt{\sigma_X^2 + \sigma_V^2}} \right] \quad (24)$$

第三部分 实验过程与参数说明

3.1 参数说明

在头文件中，已经详细介绍了每一个参数的含义和各个函数的功能，并在注释中书写了主要思想。

```
const int N = 1000;           // 数据个数 N=1000
const int Xn = 55;           // 横坐标个数
const double sigma = 10.0;    // 标准偏差 STD=10
const int cyclenums = 10000; // 达到 1e-6 的精度，需要 10^6 组数据

class embed {
protected:
    double a;                 // 嵌入强度 a
    double x_data[N];         // 数据 Xi
    double w_data[N];         // 数据 Wi
    double v_data[N];         // 数据 Vi（有攻击情形）
    double s_h1_data[N];      // h1 情形下的 Si
    double s_h0_data[N];      // h0 情形下的 Si
    int pm_nums[Xn];          // Pm 的频数，用于计算实验 Pm

    /*
        先自己定一组理论 pfa 作为自变量；然后用每一个理论 pfa，都能算出对应的理论 pm
        和阈值，
        用阈值进行检测，得到实际 pfa 和 pm；最后画 pfa（理论）为 x 轴，pm（理论、实际）
        为 y 轴的曲线；
        我们自己设的一个横坐标是对应一个阈值，然后检测 10^6 组数据对于一个阈值的实际
        pm，得到图像上的一个点。
    */

    double the_pfa[Xn] = { 1e-6, 2e-6, 3e-6, 4e-6, 5e-6, 6e-6, 7e-6, 8e-6, 9e-6,
                           1e-5, 2e-5, 3e-5, 4e-5, 5e-5, 6e-5, 7e-5, 8e-5, 9e-5,
                           1e-4, 2e-4, 3e-4, 4e-4, 5e-4, 6e-4, 7e-4, 8e-4, 9e-4,
                           1e-3, 2e-3, 3e-3, 4e-3, 5e-3, 6e-3, 7e-3, 8e-3, 9e-3,
                           1e-2, 2e-2, 3e-2, 4e-2, 5e-2, 6e-2, 7e-2, 8e-2, 9e-2,
                           1e-1, 2e-1, 3e-1, 4e-1, 5e-1, 6e-1, 7e-1, 8e-1, 9e-1, 1 }; //（规
        定的）理论 Pfa，55 个横坐标

    double exp_pfa[Xn];        // 实验 Pfa
    double the_pm[Xn];         // 理论 Pm
    double exp_pm[Xn];         // 实验 Pm
    double threshold[Xn];      // 阈值

public:
    void init(double in_a, int status); // 参数初始化
    void input_data(random_ggd gen);    // 数据初始化（GGD 分布，无攻击情形）
    void input_data(random_Gaussian gen); // 数据初始
        化（高斯分布，无攻击情形）
    void input_data(random_Gaussian gen, random_Gaussian noise); // 数据初始
```

<p>化（高斯分布，有攻击情形）</p> <pre> void xdata(random_ggd gen); // 计算 Xi void xdata(random_Gaussian gen); // 计算 Xi void vdata(random_Gaussian noise); // 计算 Vi（有攻击情形） void wdata(); // 计算 Wi void sdata(); // 计算 Si（无攻击情形） void sdata_attacked(); // 计算 Si（有攻击情形） void simulate(random_ggd gen); // 嵌入仿真（GGD 分布，无攻击情形） void simulate(random_Gaussian gen); // 嵌入仿真（高斯分布，无攻击情形） void simulate(random_Gaussian gen, random_Gaussian noise); // 嵌入仿真 （高斯分布，有攻击情形） double cal_q(const double x); // 计算 Q 函数 double cal_qinv(const double y); // 计算 Q 函数的反函数 double ierfc(const double y); // 计算 ierfc 函数 void result(double in_a); // 文件输出结果（无攻击情形） void result_attacked(double in_a, double noise); // 文件输出 结果（有攻击情形） }; </pre>	
<pre> const double euler = 2.718281828459045; const double dist_max = 1000000000.0; class random_ggd { protected: int num_s; // 生成服从 GGD 分布的随机数个数 double in_c; // 输入形状参数 c double in_beta; // 输入尺度参数 β double sum = 0; double miu = 0; double sigma_square = 0; double ex_abs; double R; double est_c; // 矩估计形状参数 public: double data[1000]; // 生成的 GGD 分布随机数 random_ggd(int num, double c, double beta); void generate_ggd(); void moment_estimation(double data[]); // 使用实验一作业中的方法三：通过取 对数，从而直接使用 gammln 函数 double gammln(double xx); }; </pre>	
<pre> class random_Gaussian { protected: </pre>	

```

    int num_sample;      // 生成高斯分布随机数个数
    double in_ave;       // 输入高斯分布  $\mu$ 
    double in_var;       // 输入高斯分布  $\sigma$ 
    double c_ave;        // 参数估计  $\mu$ 
    double c_var;        // 参数估计  $\sigma$ 
public:
    double data[1000];   // 生成的高斯分布随机数
    random_Gaussian(int num_s, double ave, double in_var); // 根据输入信息初始
    化
    void generate_Gaussian(); // 生成高斯分布
    N( $\mu$ ,  $\sigma$ )随机数
};

```

3.2 实验过程

3.2.1 无攻击情况下的情形

(1) 生成 $c=2.0$; 中心=0; STD=10 的 GGD 分布数据 X_i

```

const double ave = 0;      // 高斯分布中心
const double var = 10;     // 高斯分布标准差

random_Gaussian gen(num, ave, var);
gen.generate_Gaussian();
emb.simulate(gen);

x_data[i] = gen.data[i];

```

(2) 生成 $c=1.0$ 、 $c=0.5$; 中心=0; STD=10 的 GGD 分布数据 X_i

```

beta = 1.0 / sigma * sqrt(pow(euler, gammln(3.0 / c)) / pow(euler, gammln(1.0 / c)));
// beta: GGD 分布的尺度参数

random_ggd gen(num, c, beta);
gen.generate_ggd();
emb.simulate(gen);

x_data[i] = gen.data[i];

```

(3) 生成数据 W_i

```

//简单一点可以取+1, -1, +1, -1...
for (int i = 0; i < N; i++) {
    w_data[i] = i % 2 == 0 ? 1 : -1;
}

```

(4) 计算数据 S_i

```

// 计算  $S_i$  (无攻击情形)
void embed::sdata()
{
    for (int i = 0; i < N; i++) {
        s_h1_data[i] = x_data[i] + a * w_data[i];
        s_h0_data[i] = x_data[i];
    }
}

```

```

    }
}

// 计算 Si (有攻击情形)
void embed::sdata_attacked()
{
    for (int i = 0; i < N; i++) {
        s_h1_data[i] = x_data[i] + a * w_data[i] + v_data[i];
        s_h0_data[i] = x_data[i] + v_data[i];
    }
}

```

(5) 计算 Q 函数和 Q 的反函数(Q_inv)

```

// 计算 Q 函数
double embed::cal_q(const double x)
{
    return 0.5 * erfc(x / (sqrt(2.0)));
}

```

```

// 计算 Q 函数的反函数
double embed::cal_qinv(const double y)
{
    return sqrt(2.0) * ierfc(2 * y);
}

```

(6) 计算阈值和理论 Pm

```

threshold[i] = sigma * cal_qinv(the_pfa[i]) / (sqrt(N)); // 计算阈值
the_pm[i] = 1 - cal_q(cal_qinv(the_pfa[i]) - sqrt(N) * a / sigma); // 计算理论 Pm

```

(7) 水印检测、计算实验 Pm

```

double Sigma_SiWi = 0; // 计算 L(S)=1/N*Sigma(i=1,N) {Si*wi}
for (int i = 0; i < N; i++) {
    Sigma_SiWi += s_h1_data[i] * w_data[i];
}
double LS = Sigma_SiWi / N;

for (int i = 0; i < Xn; i++) { // 计算实验 Pm
    if (LS < threshold[i]) {
        pm_nums[i]++; // 计算 Pm 的频数
    }
}

```

(8) 改变参数绘制 ROC 曲线

3.2.2 有攻击情况下的情形

(1) 计算 Yi 的值

```

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

```

```
s_h1_data[i] = x_data[i] + a * w_data[i] + v_data[i];  
s_h0_data[i] = x_data[i] + v_data[i];  
}
```

(2) 计算阈值

```
threshold[i] = sqrt(sigma * sigma + 2.5 * 2.5) * cal_qinv(the_pfa[i]) / (sqrt(N));  
// 计算阈值
```

```
threshold[i] = sqrt(sigma * sigma + 5.0 * 5.0) * cal_qinv(the_pfa[i]) / (sqrt(N));  
// 计算阈值
```

(3) 计算实验 Pm 和理论 Pm

```
double Sigma_SiWi = 0;           // 计算  $L(S)=1/N*\Sigma(i=1,N)\{Si*wi\}$   
for (int i = 0; i < N; i++) {  
    Sigma_SiWi += s_h1_data[i] * w_data[i];  
}  
double LY = Sigma_SiWi / N;
```

```
for (int i = 0; i < Xn; i++) { // 计算实验 Pm  
    if (LY < threshold[i]) {  
        pm_nums[i]++;           // 计算 Pm 的频数  
    }  
}
```

```
the_pm[i] = 1 - cal_q(cal_qinv(the_pfa[i]) - sqrt(N) * a / sqrt(sigma * sigma + 2.5  
* 2.5)); // 计算理论 Pm  
the_pm[i] = 1 - cal_q(cal_qinv(the_pfa[i]) - sqrt(N) * a / sqrt(sigma * sigma + 5.0  
* 5.0)); // 计算理论 Pm
```

3.3 输入输出样例

根据控制台的提示，输入相应的参数或选项，执行对应任务，具体如下。

首先，根据是否有攻击，输入 A 或 B，A 代表无攻击、B 代表有攻击；

如果输入 A，则需要输入形状参数 c，c 为 2.0、1.0 或 0.5，如果输入 2.0，则计算嵌入强度 a=1.8、a=1.5 时的两种结果；如果输入 1.0，则计算嵌入强度 a=1.8 下的一种情形，如果输入 0.5，则计算嵌入强度 a=1.8 下的一种情形。

如果输入 B，则不需要再次输入任何内容，程序自动分别计算原始状态、高斯噪声 STD=2.5、高斯噪声 STD=5.0 三种情形下的数据情况。

程序完成了实验任务所规定的所有参数情形。

```
C:\WINDOWS\system32\cmd.exe
随机数个数N=1000
标准偏差STD=10
请选择攻击情况 (A: 无攻击, B: 有攻击) B

完整进度条: =====

正计算{攻击情况B: 有攻击、a=1.8、STD=0(原始状态)}条件下的情形.....
当前进度条: =====

正计算{攻击情况B: 有攻击、a=1.8、STD=2.5}条件下的情形.....
当前进度条: =====

正计算{攻击情况B: 有攻击、a=1.8、STD=5.0}条件下的情形.....
当前进度条: =====
```

(图 1 控制台交互界面样例)

为了提高程序交互的友好性，在控制台界面输出了实时进度条。

每一个进度条对应 10^6 组数据，需要执行约 10 分钟。

如需检验实验所有任务，则需要 7×10^6 组数据，需要执行约总计 70 分钟。

第四部分 实验结果与分析讨论

4.1 无攻击情况下，观察不同形状参数下的性能

(一) 要求：无攻击情况下：宿主信号 X 服从期望为 GGD 分布(中心为 0，标准偏差 $STD=10$)，嵌入强度 $a=1.8$ ， $N=1000$ 。实验要求达到的精度 $1E-6$ 。

不同形状参数下的性能：绘制 ROC 曲线（包括形状参数分别为 $c=2.0$ 、 1.0 、 0.5 时系统理论性能和实验性能）。

(二) 实验数据（部分）：如图 2 所示。

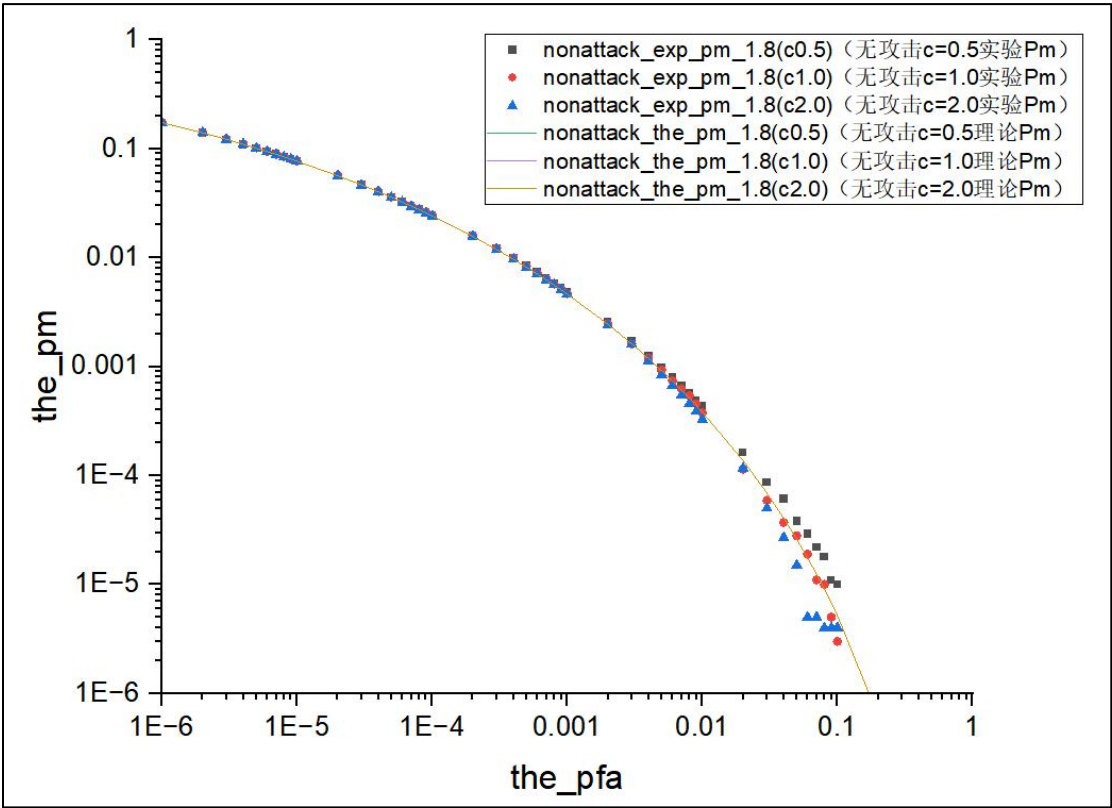
(三) 完整的实验数据见附件：“A 无攻击-c0.5-c1.0-c2.0-输出文件”文件夹下的所有文件。

	A(X)	B(Y)	C(Y)	D(Y)	E(Y)	F(Y)	G(Y)
Long Name	the_pfa	the_pm					
Units							
Comments		nonattack_exp_pm_1.8(c0.5)	nonattack_exp_pm_1.8(c1.0)	nonattack_exp_pm_1.8(c2.0)	nonattack_the_pm_1.8(c0.5)	nonattack_the_pm_1.8(c1.0)	nonattack_the_pm_1.8(c2.0)
F(x)=							
1	1E-8	0.173389	0.174659	0.174097	0.173949	0.173949	0.173949
2	2E-8	0.139446	0.140742	0.140411	0.139911	0.139911	0.139911
3	3E-8	0.121222	0.122684	0.122198	0.121866	0.121866	0.121866
4	4E-8	0.109482	0.110487	0.110167	0.109927	0.109927	0.109927
5	5E-8	0.100821	0.101539	0.101390	0.101168	0.101168	0.101168
6	6E-8	0.094061	0.094611	0.094504	0.094337	0.094337	0.094337
7	7E-8	0.086596	0.089045	0.088941	0.088791	0.088791	0.088791
8	8E-8	0.084071	0.084435	0.084226	0.084156	0.084156	0.084156
9	9E-8	0.080088	0.080585	0.080406	0.080198	0.080198	0.080198
10	1E-5	0.076780	0.077100	0.076933	0.076760	0.076760	0.076760
11	2E-5	0.056281	0.056953	0.056476	0.056526	0.056526	0.056526
12	3E-5	0.046433	0.046782	0.046363	0.046548	0.046548	0.046548
13	4E-5	0.040280	0.040545	0.040176	0.040258	0.040258	0.040258
14	5E-5	0.035792	0.036026	0.035682	0.035811	0.035811	0.035811
15	6E-5	0.032450	0.032682	0.032385	0.032448	0.032448	0.032448
16	7E-5	0.029793	0.030039	0.029680	0.029787	0.029787	0.029787
17	8E-5	0.027667	0.027915	0.027491	0.027613	0.027613	0.027613
18	9E-5	0.025840	0.026132	0.025717	0.025794	0.025794	0.025794
19	1E-4	0.024308	0.024593	0.024131	0.024243	0.024243	0.024243
20	2E-4	0.015919	0.016067	0.015781	0.015698	0.015698	0.015698
21	3E-4	0.012082	0.012240	0.011897	0.011896	0.011896	0.011896
22	4E-4	0.009878	0.009958	0.009659	0.009660	0.009660	0.009660
23	5E-4	0.008412	0.008369	0.008141	0.008162	0.008162	0.008162
24	6E-4	0.007334	0.007230	0.007079	0.007079	0.007079	0.007079
25	7E-4	0.006484	0.006372	0.006220	0.006255	0.006255	0.006255
26	8E-4	0.005793	0.005719	0.005624	0.005603	0.005603	0.005603
27	9E-4	0.005277	0.005157	0.005057	0.005075	0.005075	0.005075
28	0.001	0.004837	0.004688	0.004638	0.004636	0.004636	0.004636
29	0.002	0.002541	0.002468	0.002396	0.002447	0.002447	0.002447
30	0.003	0.001706	0.001613	0.001599	0.001618	0.001618	0.001618
31	0.004	0.001253	0.001196	0.001134	0.001183	0.001183	0.001183
32	0.005	0.000964	0.000930	0.000946	0.000916	0.000916	0.000916
33	0.006	0.000795	0.000741	0.000671	0.000736	0.000736	0.000736
34	0.007	0.000664	0.000622	0.000547	0.000609	0.000609	0.000609
35	0.008	0.000567	0.000541	0.000456	0.000513	0.000513	0.000513
36	0.009	0.000484	0.000456	0.000389	0.000440	0.000440	0.000440

(图 2: 实验数据 (部分) -SS 算法在未受攻击情况不同形状参数下的理论性能和实验性能, 宿主信号 $\sigma X = 10$, 形状分别为 0.5、1.0、2.0, $N = 1000$, $a=1.8$)

(四) 实验结果: 如图 3 所示。

(五) 分析讨论: 在未受攻击情况, 不同形状参数下的理论性能和实验性能无明显差异。



(图 3: ROC 图像-SS 算法在未受攻击情况不同形状参数下的理论性能和实验性能, 宿主信号 $\sigma X = 10$, 形状分别为 0.5、1.0、2.0, $N = 1000$, $a=1.8$)

4.2 无攻击情况下，观察不同嵌入强度下的性能

(一) 要求：无攻击情况下：宿主信号 X 服从期望为 GGD 分布(中心为 0，标准偏差 $STD=10$)，嵌入强度 $a=1.8$ ， $N=1000$ 。实验要求达到的精度 $1E-6$ 。

不同嵌入强度下的性能：绘制 ROC 曲线（强度 $a=1.5$ ， $a=1.8$ 时的理论和实验性能($c=2.0$)）。

(二) 实验数据（部分）：如图 4 所示。

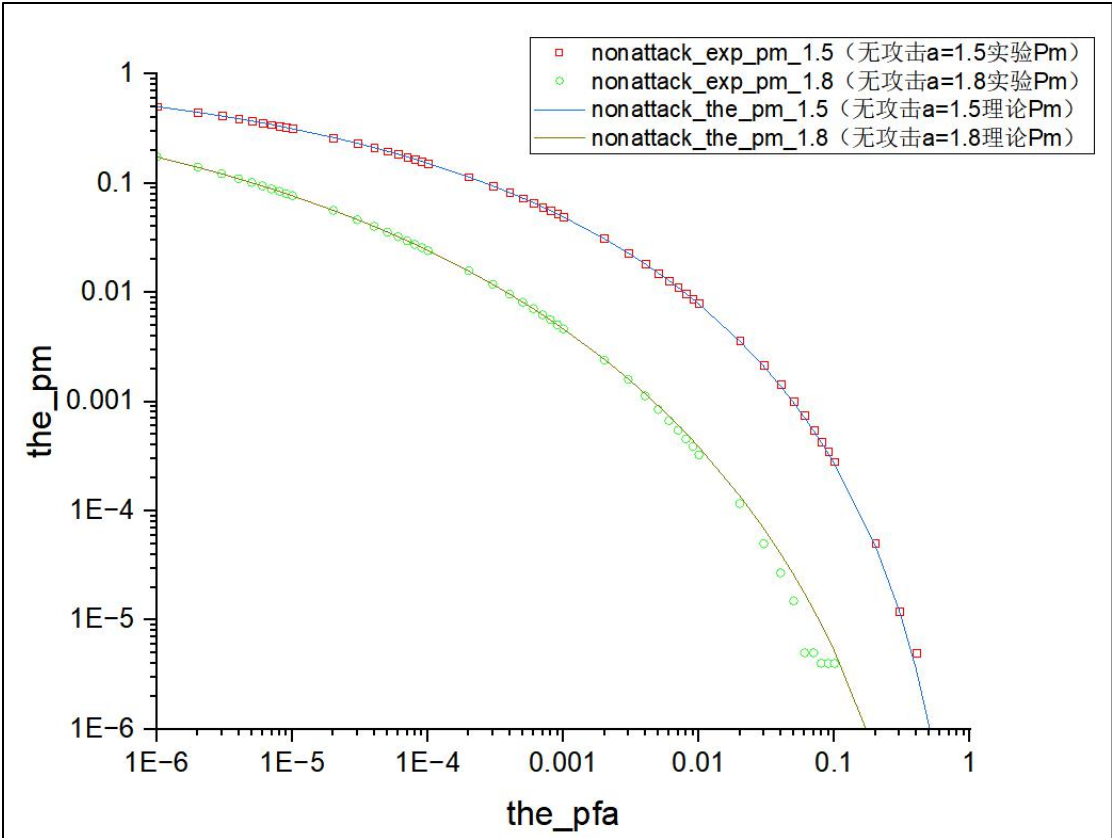
(三) 完整的实验数据见附件：“A 无攻击-a1.5-1.8-输出文件”文件夹下的所有文件。

	A(X)	B(Y)	C(Y)	D(Y)	E(Y)
Long Name	the_pfa	the_pm			
Units					
Comments		nonattack_exp_pm_1.5 (nonattack_exp_pm_1.8 (nonattack_the_pm_1.5 (nonattack_the_pm_1.8 (
F(x)=					
1	1E-6	0.504086	0.174097	0.503992	0.173949
2	2E-6	0.447677	0.140411	0.447479	0.139911
3	3E-6	0.414274	0.122198	0.414094	0.121866
4	4E-6	0.390431	0.110167	0.390417	0.109927
5	5E-6	0.371994	0.101390	0.372120	0.101168
6	6E-6	0.357326	0.094504	0.357246	0.094337
7	7E-6	0.344898	0.088941	0.344742	0.088791
8	8E-6	0.334090	0.084226	0.333974	0.084156
9	9E-6	0.324772	0.080406	0.324533	0.080198
10	1E-5	0.316356	0.076933	0.316138	0.076760
11	2E-5	0.262493	0.056476	0.262409	0.056526
12	3E-5	0.232523	0.046363	0.232510	0.046548
13	4E-5	0.212107	0.040176	0.212140	0.040258
14	5E-5	0.196924	0.035682	0.196878	0.035811
15	6E-5	0.184961	0.032385	0.184782	0.032448
16	7E-5	0.175013	0.029680	0.174830	0.029787
17	8E-5	0.166786	0.027491	0.166421	0.027613
18	9E-5	0.159490	0.025717	0.159172	0.025794
19	1E-4	0.153139	0.024131	0.152823	0.024243
20	2E-4	0.114491	0.015781	0.114424	0.015698
21	3E-4	0.094928	0.011897	0.094794	0.011896
22	4E-4	0.082272	0.009659	0.082170	0.009660
23	5E-4	0.073190	0.008141	0.073127	0.008162
24	6E-4	0.066352	0.007079	0.066222	0.007079
25	7E-4	0.060918	0.006220	0.060719	0.006255
26	8E-4	0.056426	0.005624	0.056199	0.005603
27	9E-4	0.052669	0.005057	0.052399	0.005075
28	0.001	0.049386	0.004638	0.049147	0.004636
29	0.002	0.031284	0.002396	0.031073	0.002447
30	0.003	0.023101	0.001599	0.022987	0.001618
31	0.004	0.018248	0.001134	0.018249	0.001183
32	0.005	0.015096	0.000846	0.015095	0.000916
33	0.006	0.012905	0.000671	0.012832	0.000736
34	0.007	0.011200	0.000547	0.011123	0.000609
35	0.008	0.009834	0.000456	0.009785	0.000513
36	0.009	0.008766	0.000389	0.008708	0.000440

(图 4：实验数据（部分）-SS 算法在未受攻击情况不同嵌入强度下的理论性能和实验性能，宿主信号 $\sigma X = 10$ ，嵌入强度分别为 1.8、1.5， $N = 1000$ ， $c=2.0$)

(四) 实验结果：如图 5 所示。

(五) 分析讨论：在 $c=2.0$ 、未受攻击情况下，嵌入强度越大，在相同的误检概率下，漏检概率越小。



(图 5: ROC 图像-SS 算法在未受攻击情况不同嵌入强度下的理论性能和实验性能, 宿主信号 $\sigma X = 10$, 嵌入强度分别为 1.8、1.5, $N = 1000$, $c=2.0$)

4.3 有攻击情况下，观察不同 STD 高斯噪声的影响

(一) 要求：受到攻击情况下：宿主信号 X 服从期望为高斯分布(中心为 0，标准偏差 $STD=10$)，嵌入强度 $a=1.8$, $N=1000$ 。实验要求达到的精度 $1E-6$ 。

(1) 噪声期望为 0，STD 分别为 2.5、5；

(2) 绘制此时的 ROC 曲线（未受攻击时的理论性能，受到攻击时的理论和实验性能）。

(二) 实验数据（部分）：如图 6 所示。

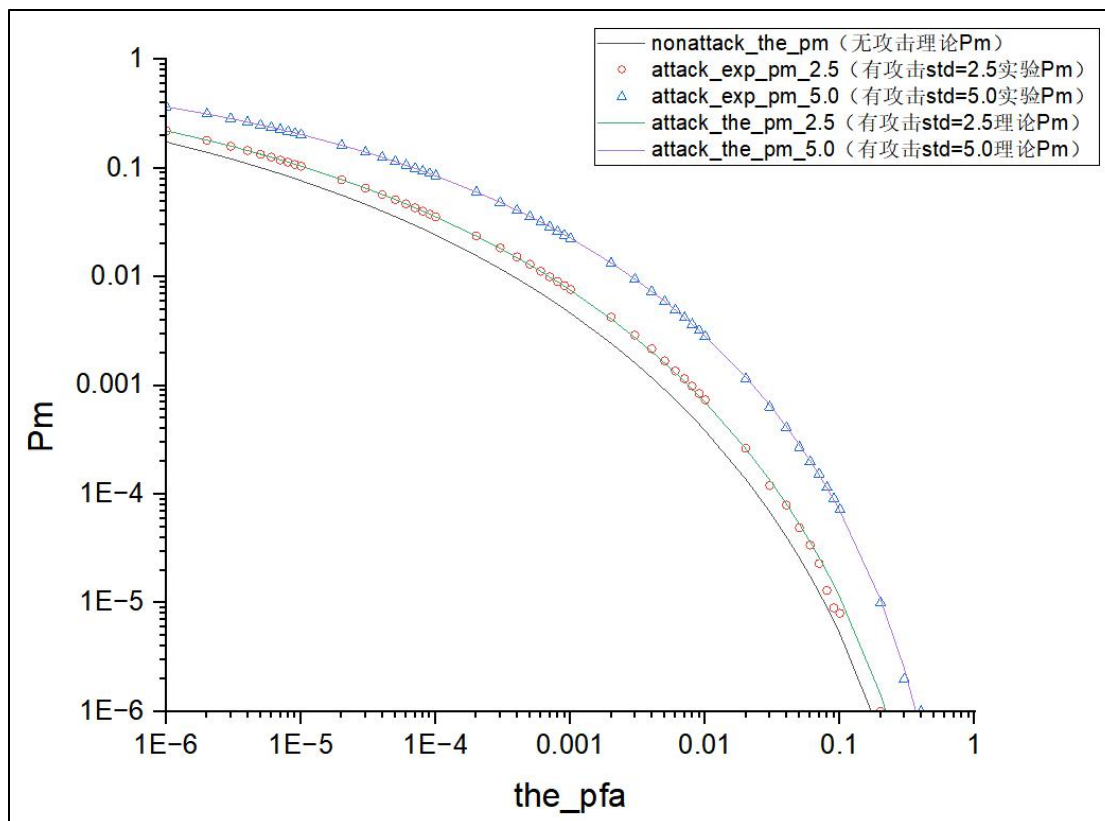
(三) 完整的实验数据见附件：“B 有攻击-输出文件”文件夹下的所有文件。

	A(X)	B(Y)	C(Y)	D(Y)	E(Y)	F(Y)
Long Name	the_pfa	the_pm				
Units						
Comments		nonattack_the_pm (attack_exp_pm_2.5 (attack_exp_pm_5.0 (有	attack_the_pm_2.5 (attack_the_pm_5.0 (
F(x)=						
6	6E-6	0.094337	0.125749	0.237180	0.126196	0.237743
7	7E-6	0.088791	0.118858	0.226783	0.119341	0.227439
8	8E-6	0.084156	0.113105	0.218067	0.113582	0.218663
9	9E-6	0.080198	0.108164	0.210482	0.108640	0.211042
10	1E-5	0.076760	0.103885	0.203778	0.104330	0.204323
11	2E-5	0.056526	0.078261	0.162186	0.078583	0.162634
12	3E-5	0.046548	0.065454	0.140115	0.065606	0.140437
13	4E-5	0.040258	0.057271	0.125470	0.057312	0.125739
14	5E-5	0.035811	0.051305	0.114951	0.051387	0.114958
15	6E-5	0.032448	0.046810	0.106453	0.046867	0.106558
16	7E-5	0.029787	0.043194	0.099684	0.043266	0.099746
17	8E-5	0.027613	0.040180	0.094106	0.040307	0.094061
18	9E-5	0.025794	0.037740	0.089315	0.037817	0.089213
19	1E-4	0.024243	0.035636	0.085075	0.035684	0.085008
20	2E-4	0.015698	0.023838	0.060253	0.023736	0.060441
21	3E-4	0.011896	0.018474	0.048358	0.018285	0.048502
22	4E-4	0.009660	0.015263	0.040835	0.015028	0.041073
23	5E-4	0.008162	0.012973	0.035631	0.012820	0.035880
24	6E-4	0.007079	0.011303	0.031827	0.011207	0.031992
25	7E-4	0.006255	0.010049	0.028791	0.009969	0.028946
26	8E-4	0.005603	0.009047	0.026293	0.008985	0.026479
27	9E-4	0.005075	0.008289	0.024233	0.008181	0.024432
28	0.001	0.004636	0.007648	0.022529	0.007510	0.022700
29	0.002	0.002447	0.004259	0.013352	0.004097	0.013449
30	0.003	0.001618	0.002905	0.009460	0.002765	0.009555
31	0.004	0.001183	0.002176	0.007308	0.002052	0.007362
32	0.005	0.000916	0.001686	0.005937	0.001608	0.005946
33	0.006	0.000736	0.001364	0.004970	0.001306	0.004954
34	0.007	0.000609	0.001153	0.004201	0.001089	0.004220
35	0.008	0.000513	0.000987	0.003645	0.000925	0.003656
36	0.009	0.000440	0.000843	0.003205	0.000798	0.003210
37	0.01	0.000382	0.000736	0.002833	0.000697	0.002848
38	0.02	0.000137	0.000264	0.001155	0.000262	0.001193
39	0.03	0.000069	0.000120	0.000634	0.000136	0.000663
40	0.04	0.000041	0.000079	0.000410	0.000081	0.000418
41	0.05	0.000026	0.000049	0.000270	0.000053	0.000284

(图 6: 实验数据 (部分) -SS 算法在未受攻击时的理论性能、受到攻击时的理论和实验性能, 宿主信号 $\sigma_X = 10$, 嵌入强度为 1.8, $N = 1000$, 攻击噪声为高斯噪声(期望为 0, 标准偏差为 2.5、5.0))

(四) 实验结果: 如图 7 所示。

(五) 分析讨论: 在相同的误检概率下, 漏检概率在受攻击时增大了, 且高斯噪声 STD 越高, 漏检概率增大地越多。



(图 7: ROC 图像-SS 算法在未受攻击时的理论性能、受到攻击时的理论和实验性能, 宿主信号 $\sigma X = 10$, 嵌入强度为 1.8, $N = 1000$, 攻击噪声为高斯噪声(期望为 0, 标准偏差为 2.5、5.0))

第五部分 实验结论

本次实验首先订正了上次实验(一)中出现的问题, 使得能够正确产生高斯分布和 GGD 分布的随机数。

本次实验与上次实验关系密切, 在生成 X_i 时, 需要产生不同形状参数下的随机数, 并且, 还需要通过规定的 STD 求出需要的尺度参数。

实验中, 完成了规定了所有任务, 发现理论 P_m 和实验 P_m 在精度为 $1e-6$ 的条件下, 达到了很高的拟合程度。观察 ROC 图像, 可以明显感受到: 漏检概率和误检概率这两个系统性能的指标是相互矛盾的, 而由于误检概率更加重要, 我们首先根据确定的误检概率计算阈值。然后计算理论 P_m 和实验 P_m ; 当我们增大嵌入强度 a 时, 便得到了更大的水印能量, 可以降低漏检概率。

当嵌入水印受到高斯噪声攻击时, 观察 ROC 图像, 可以明显感受到: 在相同的误检概率下, 漏检概率在受攻击时增大了, 且高斯噪声 STD 越高, 漏检概率增大地越多。

第三章 扩频水印

第一部分 相关内容介绍（背景知识）

1.1 DCT 变换

DCT 变换的全称是离散余弦变换 (Discrete Cosine Transform)，主要用于将数据或图像的压缩，能够将空域的信号转换到频域上，具有良好的去相关性的性能。DCT 变换本身是无损的，但是在图像编码等领域给接下来的量化、哈弗曼编码等创造了很好的条件，同时，由于 DCT 变换是对称的，所以，我们可以在量化编码后利用 DCT 反变换，在接收端恢复原始的图像信息。DCT 变换在当前的图像分析已经压缩领域有着极为广大的用途，我们常见的 JPEG 静态图像编码以及 MJPEG、MPEG 动态编码等标准中都使用了 DCT 变换。

1.2 感官失真

感官失真主要指原始作品和水印作品的差别，大部分应用场合都要求这种差别足够小，以至小到不能被人感知的程度。

我们使用均方差 (Mean Squared Error (MSE)) 来度量水印嵌入引起的失真。这种度量能在一定程度上反应失真的大小。

第二部分 实验目的、内容和原理

2.1 实验目的

通过实验，学习并掌握扩频水印嵌入技术在图像的信息隐藏中的应用；熟悉线性相关器解码的操作；探究水印的嵌入强度对作品知觉效果和解码效果的影响；并观察当水印作品受到攻击时的解码效果。

2.2 实验内容

在图像的 DCT 系数上嵌入信息，宿主图像为常见的 512×512 大小的 Lena 灰度图，嵌入信息为一二进制 Logo 图 (即该图只有两种颜色：黑和白)，该图大小为 32×32 ，共 1024 个 Bit。这些图使用课件中提供的图片。

(1) 在图像的 DCT 系数上利用加性扩频水印嵌入信息。

(2) 利用线性相关器解码，统计一下总错误率。

(3) 实验一下不同的嵌入强度对作品知觉效果和解码效果的影响，并且绘制实验曲线。

(4) 水印作品受到攻击：使用图像处理软件等加入噪声、改变对比度、JPEG 压缩等攻击，然后看看解码的效果（实验曲线）。

2.3 实验原理

2.3.1 BMP 图像文件存储格式

2.3.1.1 BMP 格式介绍

BMP 是 Bitmap (位图) 的简称, 是 Windows 操作系统中的标准图像文件格式。其特点是由于几乎不进行压缩, 所以包含的图像信息较丰富, 但同时也到之占用的磁盘空间较大。

2.3.1.2 BMP 文件格式

位图文件由 4 个部分组成:

- (1) 位图头文件 (bitmap-file header)
- (2) 位图信息头 (bitmap-information header)
- (3) 颜色表 (color table): 使用索引来表示图像, 此时颜色表就是索引与其对应的颜色之间的映射表 (即通过索引值, 结合颜色表, 找到对应的像素信息)
- (4) 位图数据 (data bits)

2.3.1.3 位图头文件与位图信息头

BMP 文件中, 数据的存储方式为小端方式 (little endian), 即假设一个数据需要多个字节来表示, 那么数据的存放字节的顺序为“低地址存放地位数据, 高地址存放高位数据”。在十六进制中, 一个数字占 4 位, 因此每个字节可以存储 2 个十六进制数字。

位图头文件的结构体可以通过以下代码理解:

```
typedef struct tagBITMAPFILEHEADER
{
    UINT16 bfType;    // 19778, 必须是 BM 字符串, 对应的十六进制为 0x4d42, 十进制
    // 为 19778, 否则不是 bmp 格式文件
    DWORD bfSize;    // 文件大小 以字节为单位 (2-5 字节)
    UINT16 bfReserved1; // 保留, 必须设置为 0 (6-7 字节)
    UINT16 bfReserved2; // 保留, 必须设置为 0 (8-9 字节)
    DWORD bfOffBits;    // 从文件头到像素数据的偏移 (10-13 字节)
} BITMAPFILEHEADER;
```

同样地, 可以定义位图信息头的结构体, 在后文的代码中体现。

2.3.2 DCT 原理

2.3.2.1 DCT 变换简介

傅里叶变换表明, 任何信号都能表示为多个不同振幅和频率的正弦或者余弦信号的叠加。如果采用的是余弦函数, 则信号分解过程称为余弦变换; 若输入信号是离散的, 则称之为离散余弦变换 (Discrete Cosine Transform, DCT)。

2.3.2.2 DCT 性质

离散余弦变换具有两点性质, 第一个为可分离性, 第二个为能量集中性。

DCT 的可分离性就是可以将二维 DCT 拆分为两个方向的一维 DCT。

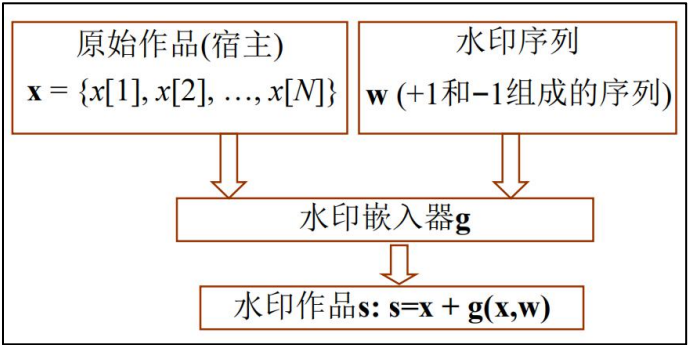
图像编码中，图像以矩阵形式存储，是一个二维数组，对图像进行二维 DCT，可以将时域图像转为频域能量分布图，实现**能量集中**从而去除空间冗余。

图像中，低频分量可以看作是基本图像，高频分量为边缘轮廓细节信息（也可能是噪声）。绝大多数能量都包含了大量平坦区域，因此大部分能量集中于低频区域（左上角），从而达到去除空间冗余的目的。（其中，DCT 变换尺寸越大，DCT 去相关性能越好，但是 DCT 的计算复杂度会随之增大）。

2.3.3 扩频水印嵌入与解码

2.3.3.1 水印嵌入原理

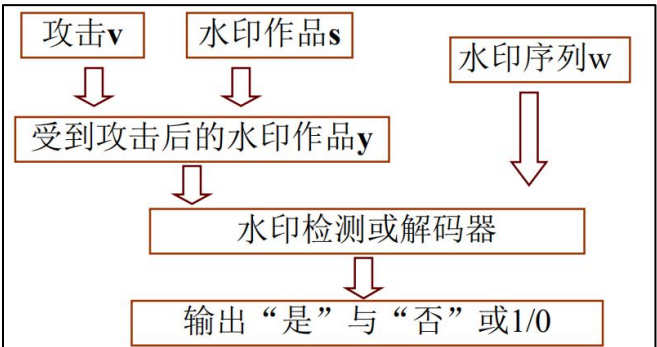
在第二章中，已经较详细地介绍了扩频水印嵌入的原理，此处不再赘述，课件中的流程图清晰地展现了在水印作品中嵌入水印序列的方法。



（图 8：水印嵌入原理流程图）

2.3.3.2 水印解码原理

水印解码是对嵌入信息的解码，是有水印检测或解码器完成的，解码结果通常是 1 或者 0；其解码流程如下图所示。



（图 9：水印解码原理流程图）

第三部分 实验过程与参数说明

3.1 参数说明

在头文件中，已经详细介绍了每一个参数的含义和各个函数的功能，如下面

代码所示。

```
class bitmap_image
{
protected:
    string bitmap_image_name;           //图像名字
    BITMAPINFOHEADER bitmap_image_message; //图像数据信息
    int bitmap_image_width;             //图像宽度
    int bitmap_image_height;            //图像高度
    int bitmap_image_len_width;         //位图数据每行字节数
    short bitmap_image_type;            //图像类型
    int bitmap_image_color_num;         //颜色位数
    RGBQUAD* pcolortable;              //颜色表
    unsigned char* bitmap_image_data;   //位图数据

    void initial_data();                //初始化图像参数信息

public:
    int width() const;                 //返回图片的宽度
    int height() const;                //返回图片的高度, 位图信息列数
    int len_width() const;             //返回位图信息列数
    unsigned char* image_data();       //返回位图数据
    ~bitmap_image();
    bool read_image(string image_name); //读取图像数据信息
    bool preserve_image(string image_name, unsigned char* image_data); //存储图像
    信息
};

class ss_watermark {
protected:
    string name_image;                 //嵌入图像文件名
    string name_mark;                  //嵌入水印文件名
    bitmap_image image;                //嵌入图像文件
    bitmap_image mark;                 //嵌入水印文件
    bitmap_image de_image;             //需要解码图像文件
    int image_width;                   //图像数据宽度
    int image_height;                  //图像数据高度
    int mark_width;                    //水印数据宽度
    int mark_height;                   //水印数据高度
    double strength;                   //嵌入强度
    vector<double> image_data;          //原始图像数据
    vector<double> image_embed;         //嵌入扩频水印后图像数据
    vector<double> decode_image;        //需要解码图像信息
    vector<int> mark_image;            //原始嵌入水印数据信息
    vector<int> mark_decode;           //对嵌入图像解码后获得水印数据信息
};
```

```

    void embed_8x8_image(double in_data[8][8], double out_data[8][8], int b); //
对 8x8 图像块进行水印嵌入
    double decode_8x8_image(double data[8][8]);
//对 8x8 图像块进行水印解码
    double count_error(); //
统计水印解码错误率
    unsigned char* transform_bit_to_byte(); //
将水印存储数据转为 bmp 存储格式
public:
    ss_watermark(string n_image, string n_mark);
    void embed_watermasking(); //水印嵌入
    void decode_watermasking(); //水印解码
    void watermark_embed_and_decode(); //水印嵌入及解码并统计解码错误
率
};

class DCT {
protected:
    int modification(double a); //修正函数，在 DCT 反变换过程中使用, 调整图像像素
值在指定范围内，求其最近整数
public:
    void dc_transform(double (* input)[NUM], double(* output)[NUM]); //DCT 正变换
    void Inverse_dc_transform(double(*input)[NUM], double(*output)[NUM]); //DCT
反变换
};

```

3.2 实验过程

3.2.1 BMP 文件读取与写回

扩频水印的嵌入与解码主要涉及：

- (1) 对 BMP 存储的位图数据进行操作；
- (2) 保存水印嵌入成功图像；
- (3) 保存水印解码后水印图像。

因此，程序需要在文件信息头读取图像宽度、高度等信息，以及颜色表和位图数据信息。读取内容的存储，主要利用动态空间的申请实现。

由于函数较长，这里仅附上函数名及其功能的说明。

```

//读取图像数据信息
bool bitmap_image::read_image(string image_name)

//存储图像信息
bool bitmap_image::preserve_image(string image_name, unsigned char* image_data)

//返回图片的宽度
int bitmap_image::width() const //返回图片的宽度
//返回图片的高度，位图信息列数
int bitmap_image::height() const //返回图片的高度

```

```
//返回位图信息列数
int bitmap_image::len_width() const
//返回位图数据
unsigned char* bitmap_image::image_data()
```

3.2.2 DCT 正反变换

根据 DCT 正反变换的原理公式，对输入 $N \times N$ 的矩阵数据进行计算。

由于本次实验是对图像数据进行 DCT 正反变换，而 DCT 变换后数据类型为 double。故需对 DCT 反变换后结果进行修正，使其代表像素值介于 0-255 之间的整数，以便减少误差。

对 DCT 变换的测试数据如下图所示：

```
原始图像数据
129 101 114 125 126 115 105 96
87 115 131 147 149 155 123 113
74 134 159 178 175 164 149 137
151 143 177 86 201 189 165 150
119 141 175 201 207 186 162 144
107 130 165 189 192 171 104 125
97 119 149 171 172 145 117 96
88 107 136 156 155 129 97 75
DCT正变换:
1111.25 -28.258 -167.823 9.25743 -6.75002 -22.5831 16.2066 14.6897
-16.1038 -20.2233 53.7266 7.88658 -10.3111 2.0685 0.59835 3.82877
-139.409 37.8089 -1.05176 -5.08508 18.8591 26.5915 -12.1636 -23.4602
6.30635 21.4798 -3.97389 2.2137 30.335 10.5256 6.78681 -15.1816
1.49998 13.3247 41.1375 29.4242 -1.5 2.99174 7.85531 31.0559
-6.34548 9.69114 36.5114 17.7704 -7.17044 -8.85925 8.7917 30.0236
-6.65708 6.13494 -15.4136 -18.443 12.2126 -0.765948 -0.69822 -14.0008
14.1601 -9.42643 -30.9966 -27.1123 8.21141 25.3771 -27.2915 -22.1311
DCT反变换:
129 101 114 125 126 115 105 96
87 115 131 147 149 155 123 113
74 134 159 178 175 164 149 137
151 143 177 86 201 189 165 150
119 141 175 201 207 186 162 144
107 130 165 189 192 171 104 125
97 119 149 171 172 145 117 96
88 107 136 156 155 129 97 75
```

(图 10: DCT 变换测试数据)

3.2.3 嵌入水印

(1) 首先将整个图像分成 8×8 的小块，假设这样共得到 M 小块。例如：512 \times 512 的 Lena 图，则 $M=4096$ 。

(2) 对每小块进行 DCT 变换，取出每块中的中频系数，假设每块我们选取 K 个系数用来嵌入数据。按照这样做，共得到 MK 个 DCT 系数可用于水印嵌入，这些系数就是扩频算法中的 $x=\{x_1, x_2, \dots, x_{M \times K}\}$ 。

(3) 根据水印嵌入公式进行水印嵌入，后对数据进行 DCT 反变换，利用得到数据替换原始对应位置数据信息。嵌入数据时，为了保证足够的健壮性，扩频算法通常需要使用多个 (N 个) 系数来嵌入一位 b 。读取所需嵌入的数据 (LOGO)，将黑色表示成 -1，白色表示成 1。

(4) 将 s_i 放回原来的位置。

3.2.4 水印解码

读取输入的已嵌入水印图像信息，将其划分为 8×8 的小块，进行 DCT 正变换，利用线性相关器对选取中频系数的进行水印解码，并保存解码所得水印数据信息，同时，计算错误率。

第四部分 实验结果与分析讨论



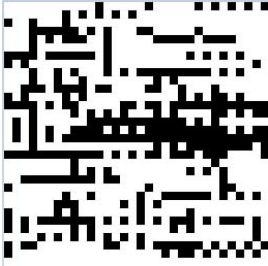

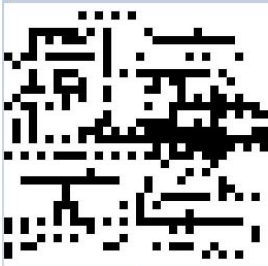

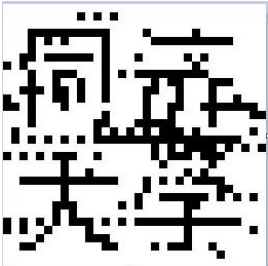

4.1 在图像的 DCT 系数上嵌入信息、统计总错误率

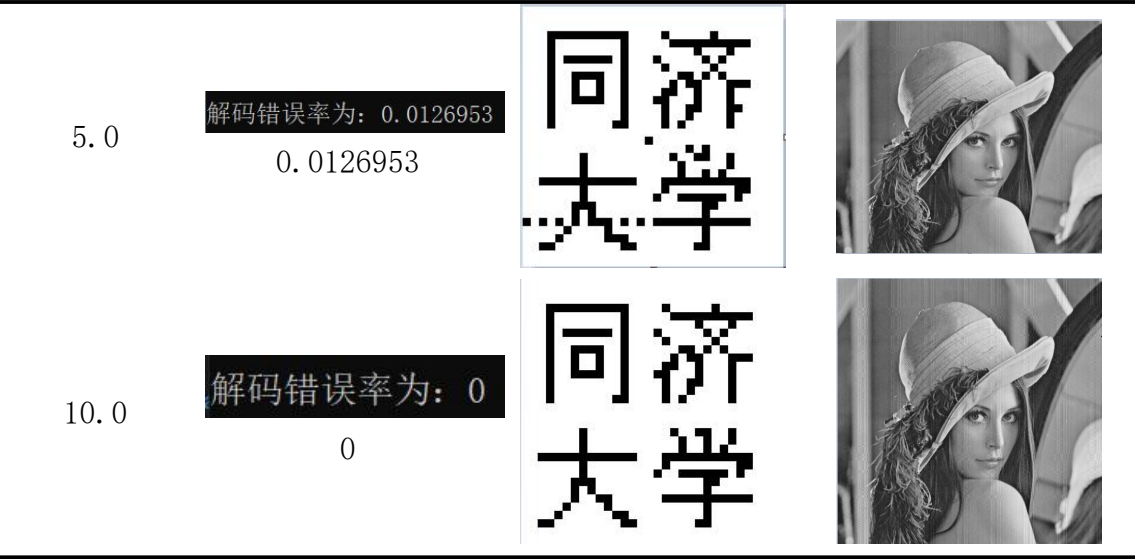
这两个实验任务点均可在任务点 4.2 的实验结果中得以体现。

4.2 实验不同的嵌入强度对作品知觉效果和解码效果的影响，并绘制实验曲线

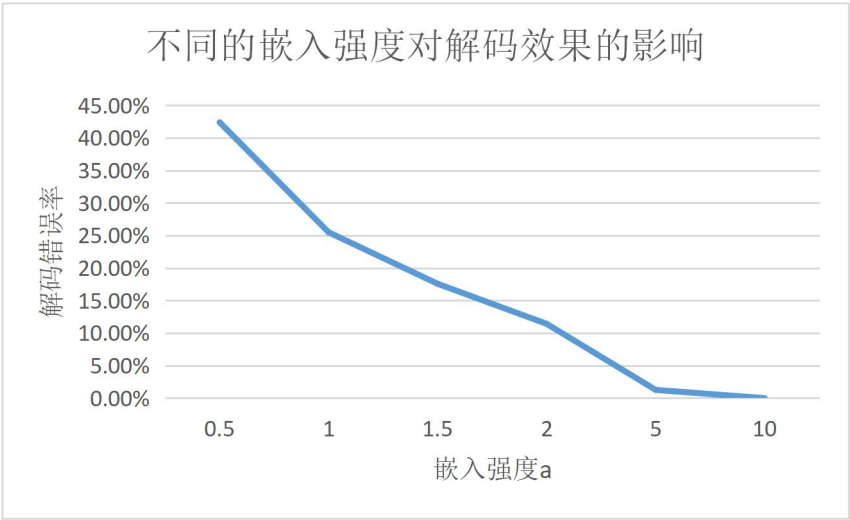
所选解码图像为未受到攻击的嵌入水印的图像。

（表 3：不同的嵌入强度对作品知觉效果和解码效果的影响）

嵌入强度 α	解码错误率	作品解码效果	作品知觉效果
0.5	解码错误率为：0.423828 0.423828		
1.0	解码错误率为：0.254883 0.254883		
1.5	解码错误率为：0.175781 0.175781		
2.0	解码错误率为：0.114258 0.114528		



实验曲线如下图所示：



（图 11：不同的嵌入强度对作品知觉效果和解码效果的影响）

实验结论：观察实验结果和实验曲线，可以发现，当嵌入强度增大时，解码错误率呈现了明显的下降趋势，但与此同时，图像的知觉效果给人的失真感约明显。

4.3 水印作品受到攻击，并绘制实验曲线

使用图像处理软件等加入噪声、改变对比度、JPEG 压缩等攻击，然后看看解码的效果（实验曲线）。

为了便于实验调参，我们编写了 matlab 代码加入高斯噪声、改变对比度。

4.3.1 加入高斯噪声

实验结果如下表所示。

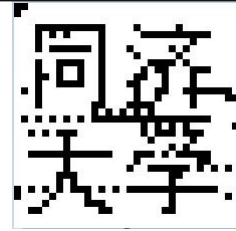
（表 4：不同程度高斯噪声攻击对水印解码效果的影响）

噪声强度	解码错误率	解码效果
------	-------	------

未加入

解码错误率为: 0.0712891

0.0712891



1%

解码错误率为: 0.22168

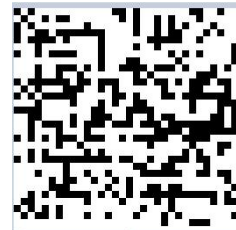
0.22168



2%

解码错误率为: 0.249023

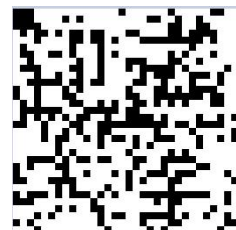
0.249023



3%

解码错误率为: 0.235352

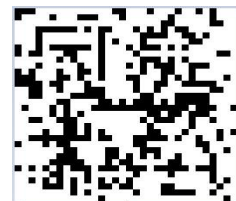
0.235352



5%

解码错误率为: 0.22168

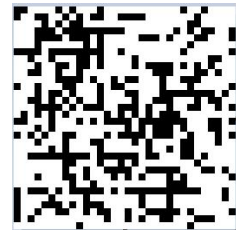
0.22168



7%

解码错误率为: 0.242188

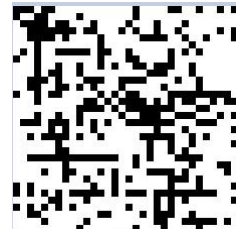
0.242188



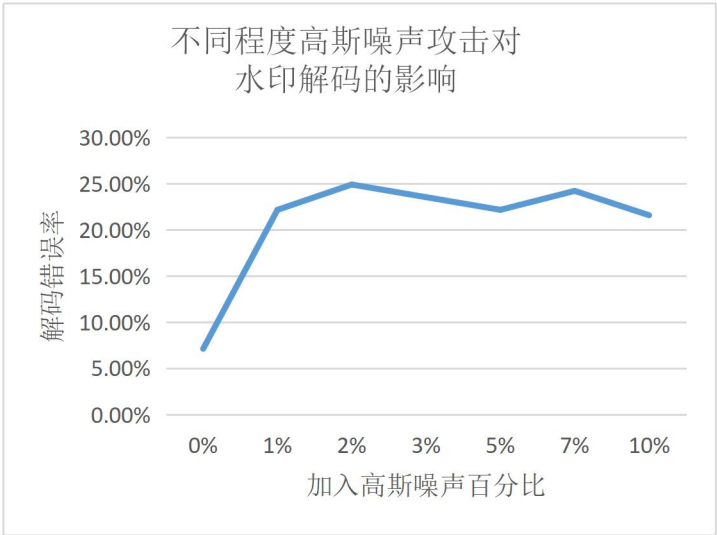
10%

解码错误率为: 0.21582

0.21582



绘制实验曲线如下图所示:



(图 12: 不同程度高斯噪声攻击对水印解码效果的影响)

实验结论：当加入高斯噪声后，图像的解码错误率有了明显增加；但是，高斯噪声攻击的百分比与解码错误率并不呈现明显的线性关系。

4.3.2 改变对比度

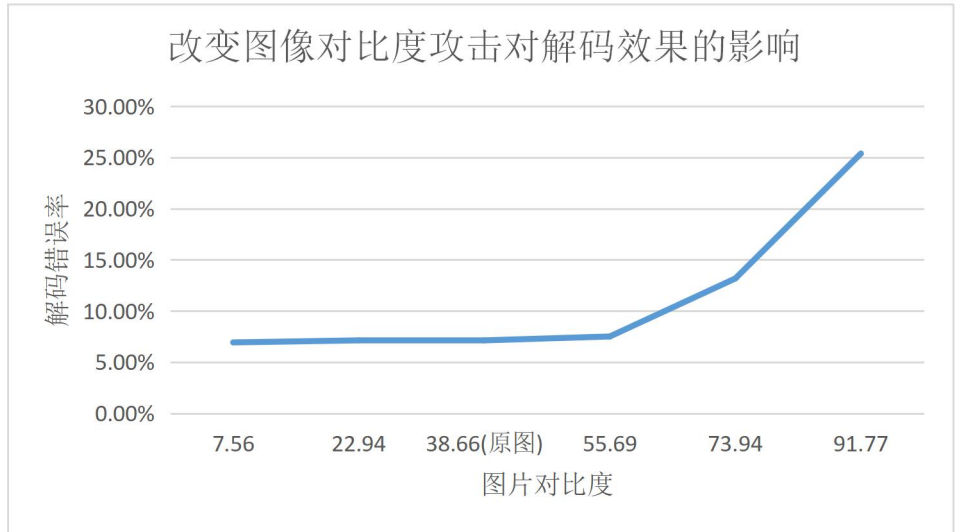
实验结果如下表所示。

(表 5: 改变图像对比度攻击对水印解码效果的影响)

对比度	解码错误率	解码效果
7.56	<div>解码错误率为: 0.0693359</div> 0.0693359	
22.94	<div>解码错误率为: 0.0712891</div> 0.0712891	
38.66 (原图)	<div>解码错误率为: 0.0712891</div> 0.0712891	

55.69	解码错误率为: 0.0751953 0.0751953	
73.94	解码错误率为: 0.131836 0.131836	
91.77	解码错误率为: 0.253906 0.253906	

绘制实验曲线如下图所示：



（图 13：改变图像对比度攻击对水印解码效果的影响）

实验结论：当进行改变图像对比度攻击时，图片对比度越高，解码错误率也越高，表现了明显的相关关系。

4.4 探究：选取不同个数 N 的 DCT 系数嵌入同一位水印 b 对作品知觉效果和解码效果的影响

嵌入强度：1.54，解码的嵌入水印图像未收到攻击。

（表 6：探究：选取不同个数 N 的 DCT 系数嵌入同一位水印 b 对作品知觉效果和解码效果的影响）

选取 N	解码错误率	作品解码效果	作品知觉效果
------	-------	--------	--------

K=8, N=32	解码错误率为: 0.248047 0.248047		
K=22, N=88	解码错误率为: 0.175781 0.175781		
K=34, N=136	解码错误率为: 0.141602 0.141602		

第六部分 实验结论

在本次实验中，首先完成了在图像的 DCT 系数上嵌入信息、统计总错误率的工作，通过实验，探究了嵌入强度对解码错误率和知觉效果的影响。当嵌入强度增大时，解码错误率呈现了明显的下降趋势，但与此同时，图像的知觉效果给人的失真感约明显。

然后，我们还通过加入高斯噪声攻击、改变图像对比度攻击，探究其对解码错误率的影响。当加入高斯噪声后，图像的解码错误率有了明显增加，但是，高斯噪声攻击的百分比与解码错误率并不呈现明显的线性关系；当进行改变图像对比度攻击时，图片对比度越高，解码错误率也越高，表现了明显的相关关系。

最后，我们还探究观察了选取不同个数 N 的 DCT 系数嵌入同一位水印 b 对作品知觉效果和解码效果的影响。

第七部分 参考文献

- [1] Monte-Carlo 实验：除课件给定的 `ierfc` 函数外，未引用他人成果（论文、程序），所有计算原理均由课件介绍或推导。
- [2] 水印嵌入实验：DCT 部分程序参考于 <https://blog.csdn.net/BigDream123/article/details/104395587> 离散余弦变换（DCT）的 C++ 实现，已在理解的基础上进行较大修改。