南开大学
Nankai University

南　　开　　大　　学

网 络 空 间 安 全 学 院

信息隐藏实验报告

实验 1：可视密码学

艾明旭 戴伊涵 刘璞睿 温博淳

年级：2021 级

指导教师：李朝晖

2024 年 3 月 26 日

目录

一、 二值图像的可视密钥分享方案

(一) 实验简述

Shamir 的二值信息分存方案，是一种典型的二值图像的可视密钥分享方案。

其主要思路是：原始图像的每个黑或自像素被 2 个子块所替代，其中每个子块由 2x2 个黑、白像素构成，这样就生成两幅膨胀了的图形，这两幅图像的叠加得到放大了 4 倍且对比度有所降低的原始图像。在本次实验中，我们所选择的密钥分配方案如下。

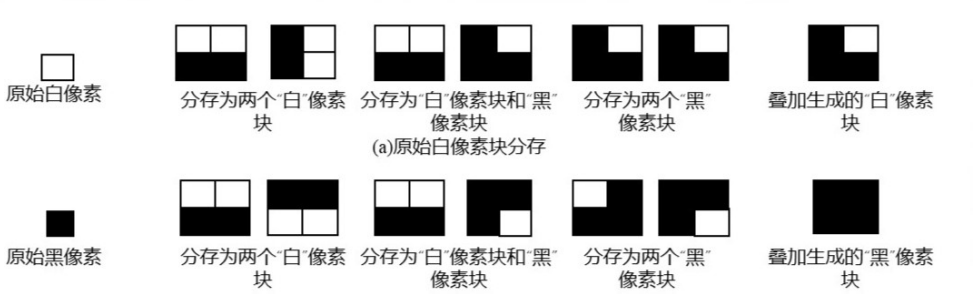

图 1: 密钥分配方案

(二) MATLAB 代码及注释

Algorithm 1 Image Encryption and Decryption Process

$x \leftarrow \text{IMREAD}('..bmp')$ ▷ 读取图片
$y \leftarrow \text{IMRESIZE}(x, [256, 256])$ ▷ 调整图片大小
$I \leftarrow \text{RGB2GRAY}(y)$ ▷ 将图片转换为灰度图
$a \leftarrow \text{IMBINARIZE}(I)$ ▷ 将灰度图二值化
$A \leftarrow \text{ZEROS}(512, 512)$ ▷ 创建一个全零矩阵 A
$B \leftarrow \text{ZEROS}(512, 512)$ ▷ 创建一个全零矩阵 B
$[height, width] \leftarrow \text{SIZE}(a)$ ▷ 获取二值化图片的尺寸
for $i = 1$ **to** $height$ **do**
    for $j = 1$ **to** $width$ **do**
        $random \leftarrow \text{RAND} \times 4$ ▷ 生成一个 0 到 4 的随机数
        if $a(i, j) = 1$ **then** ▷ 如果当前像素是白色的
            根据随机数的范围，对矩阵 A 和 B 进行不同的赋值操作
        else if $a(i, j) = 0$ **then** ▷ 如果当前像素是黑色的
            根据随机数的范围，对矩阵 A 和 B 进行不同的赋值操作
        end if
    end for
end for
$\text{IMWRITE}(A,'.1.png','png')$ ▷ 将矩阵 A 保存为图片
$\text{IMWRITE}(B,'.2.png','png')$ ▷ 将矩阵 B 保存为图片
$I \leftarrow \text{AND}(A, B)$ ▷ 对矩阵 A 和 B 进行逻辑与操作
$C \leftarrow \text{ZEROS}(256, 256)$ ▷ 创建一个全零矩阵 C
for $i = 1$ **to** $height$ **do**
    for $j = 1$ **to** $width$ **do**

if $(I(2i-1,2j-1)=1) \vee (I(2i-1,2j)=1) \vee (I(2i,2j-1)=1) \vee (I(2i,2j)=1)$ **then**
    $C(i,j) \leftarrow 1$ ▷ 如果合并后的图片中有白色像素，则将 C 中对应位置设为 1
end if
end for
end for

(三) 运行效果

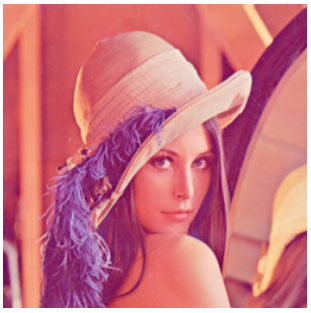

图 2: 调整大小后的图片 (256*256)

图 3: 二值化后的图片 (256*256)

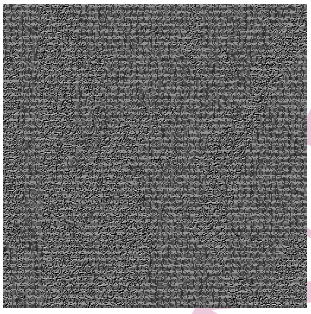

图 4: 子密钥 1(512*512)

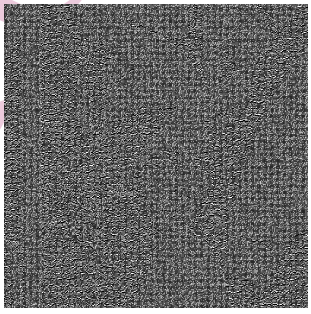

图 5: 子密钥 2(512*512)

图 6: 合并后的中等灰度图像 (512*512)

图 7: 经过缩放得到的原图.(256*256)

第一步：图像基础处理 我们需要把原始图片的大小调整为 256*256，并且通过彩色图像转化为灰度图像，再转化为二值图像使得原始图像可以表示为一个 0/1 矩阵。(见图 1、图 2)

第二步：子密钥生成 我们根据二值化图像的黑/白两种情况，结合随机数的生成确定一种密钥分配方案，分别为两个子密钥矩阵生成相应位置的 1/0 数字。注意，原始图像的一个像素会导致

子密钥四个像素的生成。（见图 3、图 4）

第三步：子密钥叠加，恢复原始图像 我们将两个子密钥进行逻辑或操作，来模拟叠加的过程，然后显示合并后的图片。（见图 5）根据密钥分享方案，通过一定的判断条件，将合并后的图片进行压缩，恢复出原始图像。（见图 6）

第四步：对比恢复出的图像和原始图像 在我们恢复出原始图像后，将其与原始图像进行对比，发现二者完全相同。（见图 7、图 8）并且，单独从两张子密钥中，难以用肉眼发现任何原始图像信息。这说明了我们实现了对图像信息的隐藏，也完成了图像信息的恢复。

图 8: 恢复出的图像 (256*256)

图 9: 二值化后的原图 (256*256)

二、 灰度图像的可视密钥分享方案

(一) 实验简述

灰度图像要进行可视密钥的分享，需要借鉴二值图像。因此，需要用到连续调图像转化为半色调图像的技术。然后再使用和上述二值图像相同的方法，进行可视秘钥的分享。

连续调图像 通常指在一幅图像上，其由淡到浓或由浅到深的色调变化是以单位面积成像物质颗粒密度来构成的，其深浅、浓淡是呈现无极变化的。

半色调图像 通常是指经过特殊加工后的印刷品上的由浅到深或由淡到浓的色调变化是由网点大小来表现的，由于网点在空间上是有一定距离而呈离散型分布的，并且由于加网的级数总有一定的限制，在图像的层次变化上是不能象连续调图像一样实现无极变化，故称为半色调图像。

(二) 伪代码及注释

Algorithm 2 灰度图像的可视密钥分享方案

$input\_img \leftarrow \text{IMREAD}('lena.bmp')$ ▷ 读取图片

$input\_img \leftarrow \text{IMRESIZE}(input\_img, [256, 256])$ ▷ 调整图片大小为 256x256

$im \leftarrow \text{RGB2GRAY}(input\_img)$ ▷ 将图片转换为灰度图

$K \leftarrow im$

$I \leftarrow \text{ZEROS}(\text{size}(K))$ ▷ 创建与 K 相同大小的零矩阵

$[height, width] \leftarrow \text{SIZE}(K)$ ▷ 获取图片的高度和宽度

Define error diffusion coefficients $a = \frac{7}{16}, b = \frac{3}{16}, c = \frac{5}{16}, d = \frac{1}{16}$

for $i = 1$ **to** $width$ **do**

    for $j = 1$ **to** $height$ **do**

```
        if K(i,j) > 127 then
            I(i,j) ← 255                                  ▷ 将像素值设为 255（白色）
        else
            I(i,j) ← 0                                    ▷ 将像素值设为 0（黑色）
        end if
        error ← (K(i,j) − I(i,j))                         ▷ 计算误差
        if j > 1 ∧ j < height ∧ i < width then
            K(i,j+1) ← K(i,j+1) + error · a
            K(i+1,j−1) ← K(i+1,j−1) + error · b
            K(i+1,j) ← K(i+1,j) + error · c
            K(i+1,j+1) ← K(i+1,j+1) + error · d
        end if
    end for
end for
im_bin ← IMBINARIZE(im)                                   ▷ 将灰度图转换为二值图
Create figures and display images I and im_bin
A, B ← ZEROS(512, 512)                                    ▷ 创建两个全零矩阵
for i = 1 to height do
    for j = 1 to width do
        random ← RAND ·4                                  ▷ 生成一个 0 到 4 的随机数
        Based on random, assign values to matrices A and B accordingly
    end for
end for
Save matrices A and B as images
Display and save the logical AND of A and B as an image
C ← ZEROS(256, 256)                                       ▷ 创建一个全零矩阵
for i = 1 to height do
    for j = 1 to width do
        if any of I(2i−1, 2j−1), I(2i−1, 2j), I(2i, 2j−1), I(2i, 2j) is 1 then
            C(i,j) ← 1
        end if
    end for
end for
Display and save matrix C as an image
```

(三) 运行效果

第一步：图像基础处理 我们需要把原始图片的大小调整为 256*256，并且通过彩色图像转化为灰度图像，再通过误差扩散法，对灰度图像进行半色调处理。另外，我们还可以再生成一个二值图像，对比二值图像和经过半色调处理的灰度图像。（如图 10、图 11）

第二步：子密钥生成 我们根据二值化图像的黑/白两种情况，结合随机数的生成确定一种密钥分配方案，分别为两个子密钥矩阵生成相应位置的 1/0 数字。注意，原始图像的一个像素会导致子密钥四个像素的生成。（见图 12、图 13）

第三步：子密钥叠加，恢复原始图像 我们将两个子密钥进行逻辑或操作，来模拟叠加的过程，然后显示合并后的图片。（见图 14）根据密钥分享方案，通过一定的判断条件，将合并后的图片进行压缩，恢复出原始图像。（见图 15）

第四步：对比恢复出的图像和原始图像 在我们恢复出图像后，将其与经过半色调处理的灰度图像进行对比，发现二者完全相同。并且，单独从两张子密钥中，难以用肉眼发现任何原始图像信息。这说明了我们实现了对图像信息的隐藏，也完成了图像信息的恢复。

图 10: 二值化处理的图像 (256*256)

图 11: 半色调化处理的图像 (256*256)

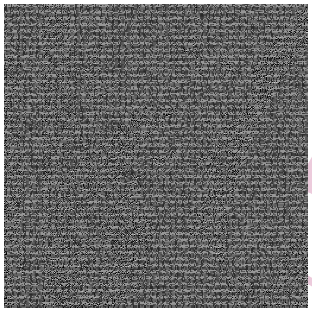

图 12: 子密钥 1(512*512)

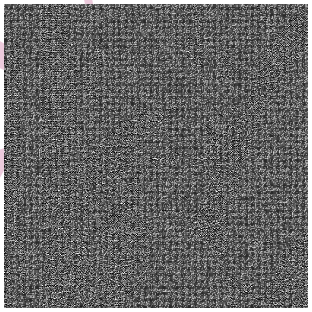

图 13: 子密钥 2(512*512)

图 14: 合并后的中等灰度图像 (512*512)

图 15: 经过缩放得到的原图.(256*256)

三、 彩色图像的可视密钥分享方案

(一) 实验简述

彩色图像不同于黑白二值图像和灰度图像，每个像素点不是由一个单值构成，而是由红、绿、蓝三个分量构成，对应的它在矩阵的存储是一个三维矩阵。每一维矩阵代表红、绿、蓝三个分量

中的一个分量。

思路 将彩色图像的每个分量当做一张图片来处理，即把一张彩色图像看做红、绿、蓝三个分量上的三张图片。对每一张图片按照“灰度图像”进行半色调处理，然后对每一张图片进行信息分存。R、G、B 分量分别分存到两张子图中，最后将得到的子图进行合并可以得到两张彩色子图。

(二) 伪代码及注释

Algorithm 3 Advanced Image Processing for Sub-Images and Overlap

$L \leftarrow \text{imread}('lena.bmp')$ ▷ 读取图片
$L \leftarrow \text{imresize}(L, [256, 256])$ ▷ 调整图片大小
$[height, width] \leftarrow \text{size}(L)$ ▷ 获取原图像大小
$width \leftarrow width/3$ ▷ 排除颜色通道数
for $t \leftarrow 1$ **to** 3 **do** ▷ 对每个颜色通道进行操作
    for $i \leftarrow 1$ **to** $height$ **do** ▷ 遍历高度
        for $j \leftarrow 1$ **to** $width$ **do** ▷ 遍历宽度
            if $L(i, j, t) > 127$ **then** ▷ 判断像素值是否大于 127
                $out \leftarrow 255$ ▷ 设为白色
            else
                $out \leftarrow 0$ ▷ 设为黑色
            end if
            $error \leftarrow L(i, j, t) - out$ ▷ 计算误差
            if $j > 1$ **and** $i < height$ **and** $j < width$ **then** ▷ 边界检查
                $L(i, j+1, t) \leftarrow L(i, j+1, t) + error \times \frac{7}{16}$
                $L(i+1, j, t) \leftarrow L(i+1, j, t) + error \times \frac{5}{16}$
                $L(i+1, j-1, t) \leftarrow L(i+1, j-1, t) + error \times \frac{3}{16}$
                $L(i+1, j+1, t) \leftarrow L(i+1, j+1, t) + error \times \frac{1}{16}$
            end if
            $L(i, j, t) \leftarrow out$ ▷ 更新当前像素值
        end for
    end for
end for
Display and save the halftone processed image L
$[red, green, blue] \leftarrow [L(:, :, 1), L(:, :, 2), L(:, :, 3)]$ ▷ 获取颜色通道
$[sub\_height, sub\_width] \leftarrow [2 \times height, 2 \times width]$ ▷ 计算子图大小
$A, B \leftarrow \text{zeros}(sub\_height, sub\_width, 3), \text{zeros}(sub\_height, sub\_width, 3)$ ▷ 初始化子图
for $t \leftarrow 1$ **to** 3 **do** ▷ 对每个颜色通道进行操作
    for $i \leftarrow 1$ **to** $height$ **do**
        for $j \leftarrow 1$ **to** $width$ **do**
            Based on $L(i, j, t)$, apply random patterns to A and B
        end for
    end for
end for
Display and save sub-images A and B
$overlap \leftarrow \text{zeros}(2 \times height, 2 \times width, 3)$ ▷ 初始化合并后的图片

for $t \leftarrow 1$ **to** 3 **do** ▷ 对每个颜色通道进行操作
    $overlap(:,:,t) \leftarrow \text{and}(A(:,:,t), B(:,:,t))$ ▷ 合并子图 A 和 B
end for
Display and save the overlapped image *overlap*
$minipic \leftarrow \text{zeros}(height, width, 3)$ ▷ 初始化缩小处理后的复原图
for $t \leftarrow 1$ **to** 3 **do** ▷ 对每个颜色通道进行操作
    for $i \leftarrow 1$ **to** *height* **do**
        for $j \leftarrow 1$ **to** *width* **do**
            if $overlap(2*i-1, 2*j-1, t) == 1$ **or** $overlap(2*i-1, 2*j, t) == 1$ **or** $overlap(2*i, 2*j-1, t) == 1$ **or** $overlap(2*i, 2*j, t) == 1$ **then**
                $minipic(i, j, t) \leftarrow 1$ ▷ 设为 1
            else
                $minipic(i, j, t) \leftarrow 0$ ▷ 设为 0
            end if
        end for
    end for
end for
Display and save the reduced version of the overlapped image *minipic*

(三) 运行效果

第一步：图像基础处理 我们需要把原始图片的大小调整为 256*256，再通过误差扩散法，对灰度图像进行半色调处理。对比原图像和经过半色调处理的灰度图像，发现其像素分布各异且无规律性，色调丰富，视觉效果较好。

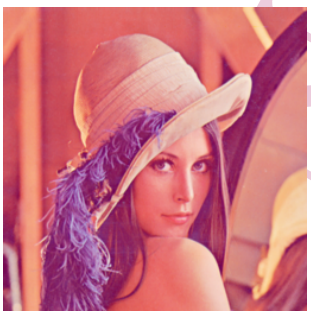

图 16: 调整大小后的原图片 (256*256)

图 17: 半色调化处理的图像 (256*256)

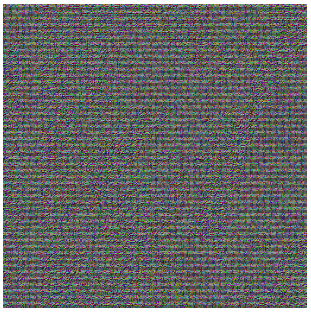

图 18: 子密钥 1(512*512)

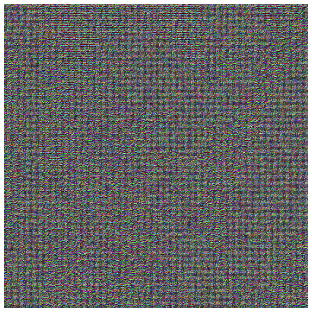

图 19: 子密钥 2(512*512)

第二步：子密钥生成 相比前面提到的灰度图像，我们分 R、G、B 三个通道，在每个通道上根据二值化图像的黑/白两种情况，结合随机数的生成确定一种密钥分配方案，分别为两个子密钥矩阵生成相应位置的 1/0 数字。注意，原始图像的一个像素会导致子密钥四个像素的生成。（见图 18、图 19）

第三步：子密钥叠加，恢复原始图像 我们将两个子密钥进行逻辑或操作，来模拟叠加的过程，然后显示合并后的图片。（见图 20）根据密钥分享方案，通过一定的判断条件，将合并后的图片进行压缩，恢复出原始图像。（见图 21）

第四步：对比恢复出的图像和经过半色调处理的图像 在我们恢复出图像后，将其与经过半色调处理的彩色图像进行对比，发现二者完全相同。并且，单独从两张子密钥中，难以用肉眼发现任何原始图像信息。这说明了我们实现了对图像信息的隐藏，也完成了图像信息的恢复。

图 20: 合并后的图像 (512*512)

图 21: 经过缩放得到的原图 (256*256)

四、 (t,n) 可视密钥分享方案

(一) 实验简述

1. (k,n) 可视加密主要思想

(k, n) 可视加密（Visual Cryptography）是一种加密技术，其主要思想是将一幅秘密图像分割成多个子图像，称为分享图像，使得单独观察任何一个分享图像都不会泄露出原始图像的信息，但是当将这些分享图像叠加在一起时，原始图像的信息将会显现出来。在 (k, n) 可视加密中，其中 k 张分享图像中的任意 k-1 张都无法还原出原始图像，只有当至少收集齐 n 张分享图像时，才能还原出原始图像。

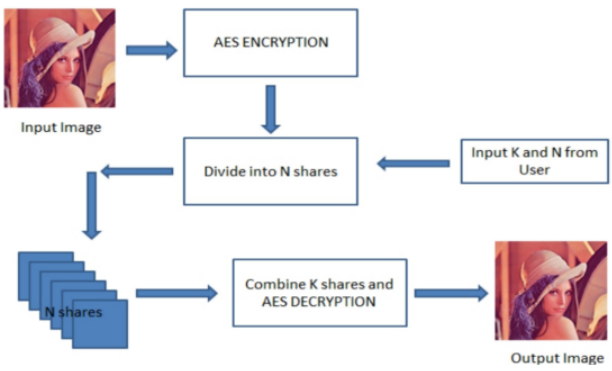

图 22: （k，n）加密算法示意

这种方法的优势在于，不需要进行复杂的数学运算，只需简单的视觉叠加操作即可恢复原始信息。这使得可视加密在某些特定场景下非常实用，比如需要安全传输敏感图像但又不希望直接暴露给单个接收者的情况下。

2. 实现方法

1. 图像读取与预处理：从文件中读取图像（这里使用了 lena.bmp），将图像转换为灰度图像，简化处理。
2. 多项式生成：使用多项式生成函数 polynomial 生成 n 个随机多项式，对每个多项式，随机生成 r-1 个系数，构建多项式，计算多项式在像素点处的值，并将其加到原始图像数据上。对结果取模，确保像素值在合理范围内。
3. 保存生成的图像：将生成的图像保存为 JPEG 格式。
4. 解码：使用解码函数 decode 解码生成的 n 个多项式，恢复原始图像数据。对每个像素点进行解码，通过拉格朗日插值恢复原始像素值。

其中有两个重要的函数：

多项式生成函数 随机生成多项式的系数矩阵，存储在 coef 中。构建基向量，计算每个像素点对应的基函数值，并将其加到原始图像数据上。对结果取模，确保像素值在合理范围内。

解码函数 使用拉格朗日插值法恢复原始像素值，对每个像素点进行解码，通过多项式插值计算原始像素值。

(二) 伪代码及注释

下面代码以 k=3，n=5 为例。

Algorithm 4 Image Processing Algorithm

Read image from file 'lena.bmp' and convert to grayscale
Flatten the image data
Set parameters: $n = 5$, $r = 3$
Generate polynomial images using POLYNOMIAL$(img_flattened, n, r)$
for $i \leftarrow 1$ to n **do**
    Save generated image i as 'test2_i.jpeg'
end for
Decode generated images using DECODE$(gen_imgs(1:r,:), index, r, n)$
Save decoded image as 'test2_origin.jpeg'

Algorithm 5 Polynomial Image Generation Function

function POLYNOMIAL(img, n, r)
    $num\_pixels \leftarrow \text{size}(img, 1)$
    $coef \leftarrow \text{randi}([0, 250], num\_pixels, r-1)$
    $gen\_imgs \leftarrow \text{zeros}(n, num\_pixels)$
    for $i \leftarrow 1$ to n **do**
        $base \leftarrow \text{zeros}(1, r-1)$
        for $j \leftarrow 1$ to $r-1$ **do**

$base(j) \leftarrow i^j$
end for
$base \leftarrow coef \times base'$
$img \leftarrow \text{double}(img)$
$img\_ \leftarrow img + \text{mod}(base, 251)$
$img\_ \leftarrow \text{mod}(img\_, 251)$
$gen\_imgs(i, :) \leftarrow img\_'$
end for
return *gen_imgs*
end function

Algorithm 6 Decoding Function

function DECODE($imgs, index, r$)
assert $\text{size}(imgs, 1) \geq r$
$dim \leftarrow \text{size}(imgs, 2)$
$origin\_img \leftarrow \text{zeros}(1, dim)$
for $i \leftarrow 1$ to dim **do**
if $\text{mod}(i, 10000) = 0$ **then**
disp('Decoding ' + i + 'th pixel')
end if
$y \leftarrow imgs(:, i)'$
$pixel \leftarrow \text{mod}(\text{lagrange}(index, y, 0), 251)$
$origin\_img(i) \leftarrow pixel$
end for
return *origin_img*
end function

(三) 运行效果

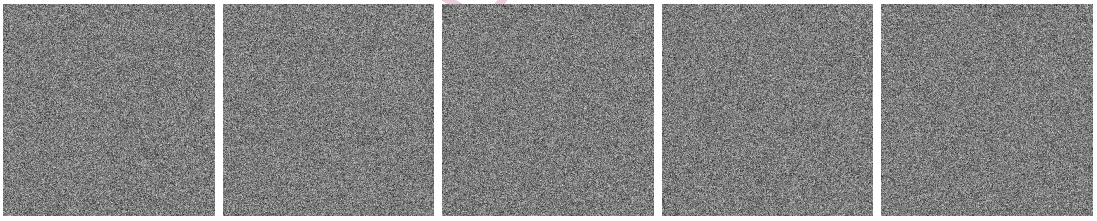

图 23: 子密钥 1 图 24: 子密钥 2 图 25: 子密钥 3 图 26: 子密钥 4 图 27: 子密钥 5

将任意四个子密钥两元组进行重叠，发现均不能还原图像：

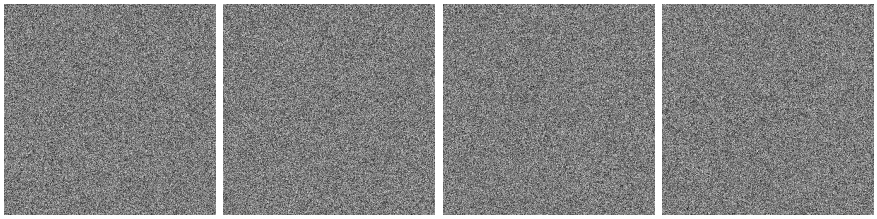

图 28: 1 和 2 图 29: 2 和 3 图 30: 4 和 1 图 31: 5 和 3

将任意子密钥三元组进行重叠，发现均可以还原图像：

图 32: 三元组进行解密的结果

五、 叠像术

(一) 实验简述

要实现叠像术，首先要讨论二值图像和灰度图像叠像术的实现。

二值图像叠像术的实现 在可视密码学的基础上，需要再找两张图片进行子图的隐藏。选取两张和原图尺寸一样的图片来隐藏。对三张图片中一一对应的每个像素点：原始图像像素点可能为黑或白，分存原始子图 1 像素点可能为黑或白，分存原始子图 2 像素点也可能为黑或白。具体的实现方法在前面已经进行了详细讨论，这里不再赘述。

灰度图像叠像术的实现 有了灰度图像的可是密码技术的基础，灰度图像的叠像术处理相对变得简单一些。首先我们将要隐藏的灰度图像和用于掩饰的两张灰度图像进行半色调处理，处理得到的图像的像素点的值只有 0 和 255 两个。参照二值图像叠像术的实现，我们可以将灰度图像的 255 视为黑白图像的“1”，对其进行类似的二值化处理。前面已经具体介绍，这里不再赘述。

(二) 伪代码及注释

Algorithm 7 叠像术误差扩散算法

```
scale_factor ← 255                                          ▷ 设置缩放因子为 255
im_out ← zeros(size(im_gray))                               ▷ 初始化输出图像
for i ← 1 to size(im_gray, 1) do
    for j ← 1 to size(im_gray, 2) do
        old_gray ← im_gray(i, j)                            ▷ 获取当前像素灰度值
        new_gray ← round(old_gray / scale_factor) * scale_factor   ▷ 计算新的灰度值
        im_out(i, j) ← new_gray                             ▷ 更新输出图像
        quant_error ← old_gray - new_gray                   ▷ 计算量化误差
        if j < size(im_gray, 2) then
            im_gray(i, j+1) += quant_error * 7 / 16         ▷ 传播误差至右侧像素
        end if
        if i < size(im_gray, 1) and j > 1 then
            im_gray(i+1, j-1) += quant_error * 3 / 16       ▷ 传播误差至左下像素
        end if
        if i < size(im_gray, 1) then
            im_gray(i+1, j) += quant_error * 5 / 16         ▷ 传播误差至下方像素
        end if
```

```
        if i < size(im_gray, 1) and j < size(im_gray, 2) then
            im_gray(i+1, j+1) += quant_error * 1 / 16          ▷ 传播误差至右下像素
        end if
    end for
end for
```

Algorithm 8 叠像术将彩色图像变成灰度图像并进行处理

```
for i = 1 to 256 do
    for j = 1 to 256 do
        if A(i, j) == 0 then
            random ← rand() * 4
            if B(i, j) == 0 and C(i, j) == 0 then
                if random ≥ 0 and random < 1 then
                    Y(2i−1, 2j−1) ← 0, Y(2i−1, 2j) ← 255, Y(2i, 2j−1) ← 0, Y(2i, 2j) ← 0
                    Z(2i−1, 2j−1) ← 255, Z(2i−1, 2j) ← 0, Z(2i, 2j−1) ← 0, Z(2i, 2j) ← 0
                else if random ≥ 1 and random < 2 then
                    // 类似的条件语句以及适当的赋值
                else if random ≥ 2 and random < 3 then
                    // 类似的条件语句以及适当的赋值
                else
                    // 类似的条件语句以及适当的赋值
                end if
            else if B(i, j) == 0 and C(i, j) == 255 then
                // 类似的条件语句以及适当的赋值
            else if B(i, j) == 255 and C(i, j) == 0 then
                // 类似的条件语句以及适当的赋值
            else if B(i, j) == 255 and C(i, j) == 255 then
                // 类似的条件语句以及适当的赋值
            end if
        else if A(i, j) == 255 then
            // 白色情况下的类似条件语句
        end if
    end for
end for
```

Algorithm 9 叠像术原始图像恢复

```
X1 = And(Y1, Z1)                                   ▷ 使用 Y1 和 Z1 计算 X1
X2 = And(Y2, Z2)                                   ▷ 使用 Y2 和 Z2 计算 X2
X3 = And(Y3, Z3)                                   ▷ 使用 Y3 和 Z3 计算 X3
for i = 1 to 512 do
    for j = 1 to 512 do
        for k = 1 to 3 do
            if k == 1 then
                X[i, j, k] = X1[i, j]              ▷ 使用 X1 设置 X 分量
                Y[i, j, k] = Y1[i, j]              ▷ 使用 Y1 设置 Y 分量
```

```
            Z[i,j,k] = Z1[i,j]                          ▷ 使用 Z1 设置 Z 分量
        else if k == 2 then
            X[i,j,k] = X2[i,j]                          ▷ 使用 X2 设置 X 分量
            Y[i,j,k] = Y2[i,j]                          ▷ 使用 Y2 设置 Y 分量
            Z[i,j,k] = Z2[i,j]                          ▷ 使用 Z2 设置 Z 分量
        else if k == 3 then
            X[i,j,k] = X3[i,j]                          ▷ 使用 X3 设置 X 分量
            Y[i,j,k] = Y3[i,j]                          ▷ 使用 Y3 设置 Y 分量
            Z[i,j,k] = Z3[i,j]                          ▷ 使用 Z3 设置 Z 分量
        end if
    end for
  end for
end for
subplot(1,3,1)                                          ▷ 为 X 创建子图
imshow(X)                                               ▷ 显示 X
subplot(1,3,2)                                          ▷ 为 Y 创建子图
imshow(Y)                                               ▷ 显示 Y
subplot(1,3,3)                                          ▷ 为 Z 创建子图
imshow(Z)                                               ▷ 显示 Z
```

(三) 运行效果

第一步：误差扩散法 误差扩散法是一种比较流行且效果较好的半色调技术，基本思想是先按照一定的扫描路径临界值量化图像像素，然后以一定的方式扩散到相邻未处理的像素上，目的是希望能够保存平均的阶调值相同，并且企图使阶调的分布局部化，已达到保留连续影像细部的资讯，但因为误差扩散将黑白均匀的分散，造成高频资讯部份遗失，使得高频品质较差。

其工作原理如图 27 所示，Pi(i,j) 是原始影像灰阶值，Po(i,j) 是处理后的输出值，v(i,j)= Pi(i,j)+ 误差修正值 (即误差扩散滤波器之输出)，e(i,j)= Po(i,j)-v(i,j)，t 是临界值，一般选用的临界值是取 0-255 的中间值 127，(i,j) 是像素点座标，若 v(i,j)>t 则 Po(i,j)=255(即输出白点)，反之 Po(i,j)=0(即输出黑点)。误差扩散滤波器 h(i,j) 以加权的方式将误差值 e(i,j) 分散至邻近像素的影像值，如此可以调整因量化所造成的明暗度偏差，使输出影像的整体视觉效果能更近似原始的输入影像。

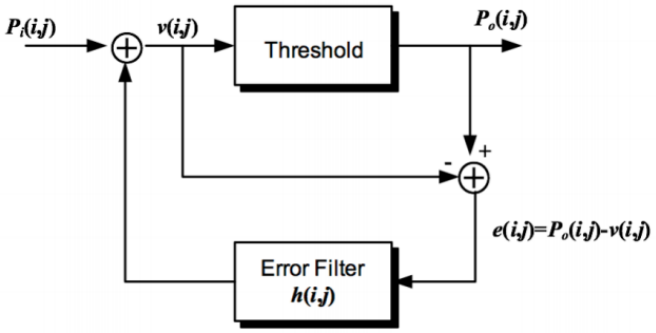

图 33: 误差扩散法原理

图 34: 要加密的原图和两个分布子图半色调处理结果

第二步：信息分存与恢复 对于半色调处理过的图片，仍然把红、绿、蓝三个分量分开考虑，把每一个分量看成一张图片。彩色图像不同于黑白二值图像和灰度图像，可以看作是三个不同分量维度的图像的合并，即 R,G,B 三个图像的合并。每一维矩阵代表红、绿、蓝三个分量中的一个分量。每个子图像当做灰度图像来处理。每个像素不是由一个单值构成的，而是由红、蓝、绿三种分量构成，对应它在矩阵的存储是一个三维矩阵。

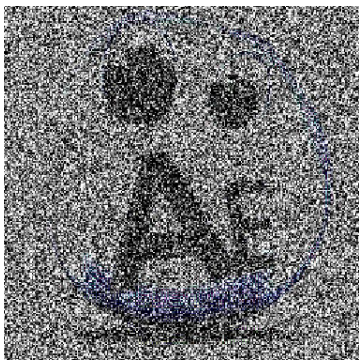

图 35: 分布后的子图

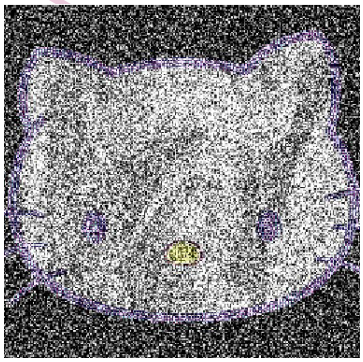

图 36: 分布后的子图

图 37: 两个字图合并结果

第三步：原始图像恢复 将 RGB 三个分量叠加得到彩色子图合并得到原始图像，如图 32

图 38: 合并子图得到的原始图像

六、 附录-Matlab 代码

(一) 二值图像的可视密钥分享方案

Listing 1: 二值图像的可视密钥分享方案代码

```
x=imread('.\lena.bmp'); % 读取图片
y=imresize(x, [256, 256]); % 调整图片大小
I=rgb2gray(y); % 将图片转换为灰度图
a=imbinarize(I); % 将灰度图二值化
figure(1); % 创建图形窗口
imshow(x);title('原始图片'); % 显示原始图片
imwrite(x,'.\test1\原始图片.png','png');
figure(2); % 创建另一个图形窗口
imshow(y);title('调整大小后的图片'); % 显示调整大小后的图片
imwrite(y,'.\test1\调整大小后的图片.png','png');
figure(3); % 创建另一个图形窗口
imshow(a);title('二值化后的图片'); % 显示二值化后的图片
imwrite(a,'.\test1\二值化后的图片.png','png');
A=zeros(512,512); % 创建一个全零矩阵A
B=zeros(512,512); % 创建一个全零矩阵B
[height,width]=size(a); % 获取二值化图片的尺寸
for i=1: height % 遍历图片的每一行
    for j=1: width % 遍历图片的每一列
        if(a(i,j)==1) % 如果当前像素是白色的
            random=rand()*4; % 生成一个0到4的随机数
            % 根据随机数的范围, 对矩阵A和B进行不同的赋值操作
            if(random>0&&random<=1)
                A(2*i-1,2*j-1)=1;A(2*i-1,2*j)=1;A(2*i,2*j-1)=0;A(2*i,2*j)=0;
                B(2*i-1,2*j-1)=0;B(2*i-1,2*j)=1;B(2*i,2*j-1)=0;B(2*i,2*j)=1;
            end
            if(random>1&&random<=2)
                A(2*i-1,2*j-1)=1;A(2*i-1,2*j)=1;A(2*i,2*j-1)=0;A(2*i,2*j)=0;
                B(2*i-1,2*j-1)=0;B(2*i-1,2*j)=1;B(2*i,2*j-1)=0;B(2*i,2*j)=0;
            end
            if(random>2&&random<3)
                A(2*i-1,2*j-1)=0;A(2*i-1,2*j)=1;A(2*i,2*j-1)=0;A(2*i,2*j)=0;
```

```
                B(2*i-1,2*j-1)=1;B(2*i-1,2*j)=1;B(2*i,2*j-1)=0;B(2*i,2*j)=0;
            end
            if(random>3&&random<=4)
                A(2*i-1,2*j-1)=0;A(2*i-1,2*j)=1;A(2*i,2*j-1)=0;A(2*i,2*j)=0;
                B(2*i-1,2*j-1)=0;B(2*i-1,2*j)=1;B(2*i,2*j-1)=0;B(2*i,2*j)=0;
            end
        end
        if(a(i,j)==0) % 如果当前像素是黑色的
            random=rand()*4; % 生成一个0到4的随机数
            % 根据随机数的范围，对矩阵A和B进行不同的赋值操作
            if(random>0&&random<=1)
                A(2*i-1,2*j-1)=1;A(2*i-1,2*j)=1;A(2*i,2*j-1)=0;A(2*i,2*j)=0;
                B(2*i-1,2*j-1)=0;B(2*i-1,2*j)=0;B(2*i,2*j-1)=1;B(2*i,2*j)=1;
            end
            if(random>1&&random<=2)
                A(2*i-1,2*j-1)=1;A(2*i-1,2*j)=1;A(2*i,2*j-1)=0;A(2*i,2*j)=0;
                B(2*i-1,2*j-1)=0;B(2*i-1,2*j)=0;B(2*i,2*j-1)=0;B(2*i,2*j)=1;
            end
            if(random>2&&random<=3)
                A(2*i-1,2*j-1)=0;A(2*i-1,2*j)=0;A(2*i,2*j-1)=0;A(2*i,2*j)=1;
                B(2*i-1,2*j-1)=1;B(2*i-1,2*j)=1;B(2*i,2*j-1)=0;B(2*i,2*j)=0;
            end
            if(random>3&&random<=4)
                A(2*i-1,2*j-1)=1;A(2*i-1,2*j)=0;A(2*i,2*j-1)=0;A(2*i,2*j)=0;
                B(2*i-1,2*j-1)=0;B(2*i-1,2*j)=0;B(2*i,2*j-1)=0;B(2*i,2*j)=1;
            end
        end
    end
end
imwrite(A,'.\test1\subkey1.png','png'); % 将矩阵A保存为图片
imwrite(B,'.\test1\subkey2.png','png'); % 将矩阵B保存为图片
figure(4); % 创建图形窗口
imshow(A);title('子密钥1'); % 显示子密钥1
figure(5); % 创建另一个图形窗口
imshow(B);title('子密钥2'); % 显示子密钥2
I=and(A,B); % 对矩阵A和B进行逻辑或操作
figure(6); % 创建另一个图形窗口
imshow(I);title('合并后的四倍大小的中等灰度原图'); % 显示合并后的图片
imwrite(I,'.\test1\合并后的四倍大小的中等灰度原图.png','png');
C=zeros(256,256); % 创建一个全一矩阵C
for i=1:height % 遍历图片的每一行
    for j=1:width % 遍历图片的每一列
        if((I(2*i-1,2*j-1)==1)||(I(2*i-1,2*j)==1)||(I(2*i,2*j-1)==1)||(I(2*i
            ,2*j)==1))
            C(i,j)=1; % 如果合并后的图片中有白色像素，则将C中对应位置设为1
        end
    end
end
```

```
figure(7); % 创建图形窗口
imshow(C);title('经过缩放得到的原图'); % 显示经过处理后得到的原图
imwrite(C,'.\test1\经过缩放得到的原图.png','png');
```

(二) 灰度图像的可视密钥分享方案

Listing 2: 灰度图像的可视密钥分享方案代码

```
input_img = imread('lena.bmp'); % 读取图片
input_img = imresize(input_img, [256, 256]); % 调整图片大小为256x256
im = rgb2gray(input_img); % 将图片转换为灰度图
% 半色调化处理
K = im;
I = zeros(size(K)); % 创建与K相同大小的零矩阵
[height,width]=size(K); % 获取图片的高度和宽度
a = 7/16; % 定义误差扩散系数
b = 3/16;
c = 5/16;
d = 1/16;
for i = 1:width % 遍历每一列
    for j = 1:height % 遍历每一行
        if K(i,j) > 127 % 判断像素值是否大于127
            I(i,j) = 255; % 将像素值设为255（白色）
        else
            I(i,j) = 0; % 将像素值设为0（黑色）
        end
        error = (K(i,j) - I(i,j)); % 计算误差
        % 误差扩散
        if j > 1 && j < height && i < width
            K(i,j+1) = K(i,j+1) + error * a;
            K(i+1,j-1) = K(i+1,j-1) + error * b;
            K(i+1,j) = K(i+1,j) + error * c;
            K(i+1,j+1) = K(i+1,j+1) + error * d;
        end
    end
end
% 二值化处理
im_bin = imbinarize(im); % 将灰度图转换为二值图
% 显示输出图像
figure(1);
subplot(1,2,1);imshow(I);title("半色调化处理的图像");
imwrite(I,'.\test2\半色调化处理的图像.png','png'); % 保存半色调化处理的图像
subplot(1,2,2);imshow(im_bin);title("二值化处理的图像");
imwrite(im_bin,'.\test2\二值化处理的图像.png','png'); % 保存二值化处理的图像
A=zeros(512,512); % 创建一个全零矩阵A
B=zeros(512,512); % 创建一个全零矩阵B
[height,width]=size(I); % 获取二值化图片的尺寸
for i=1: height % 遍历图片的每一行
```

```
    for j=1: width % 遍历图片的每一列
        if(I(i,j)==255) % 如果当前像素是白色的
            random=rand()*4; % 生成一个0到4的随机数
            % 根据随机数的范围，对矩阵A和B进行不同的赋值操作
            if(random>0&&random<=1)
                A(2*i-1,2*j-1)=1;A(2*i-1,2*j)=1;A(2*i,2*j-1)=0;A(2*i,2*j)=0;
                B(2*i-1,2*j-1)=0;B(2*i-1,2*j)=1;B(2*i,2*j-1)=0;B(2*i,2*j)=1;
            end
            if(random>1&&random<=2)
                A(2*i-1,2*j-1)=1;A(2*i-1,2*j)=1;A(2*i,2*j-1)=0;A(2*i,2*j)=0;
                B(2*i-1,2*j-1)=0;B(2*i-1,2*j)=1;B(2*i,2*j-1)=0;B(2*i,2*j)=0;
            end
            if(random>2&&random<3)
                A(2*i-1,2*j-1)=0;A(2*i-1,2*j)=1;A(2*i,2*j-1)=0;A(2*i,2*j)=0;
                B(2*i-1,2*j-1)=1;B(2*i-1,2*j)=1;B(2*i,2*j-1)=0;B(2*i,2*j)=0;
            end
            if(random>3&&random<=4)
                A(2*i-1,2*j-1)=0;A(2*i-1,2*j)=1;A(2*i,2*j-1)=0;A(2*i,2*j)=0;
                B(2*i-1,2*j-1)=0;B(2*i-1,2*j)=1;B(2*i,2*j-1)=0;B(2*i,2*j)=0;
            end
        end
        if(I(i,j)==0) % 如果当前像素是黑色的
            random=rand()*4; % 生成一个0到4的随机数
            % 根据随机数的范围，对矩阵A和B进行不同的赋值操作
            if(random>0&&random<=1)
                A(2*i-1,2*j-1)=1;A(2*i-1,2*j)=1;A(2*i,2*j-1)=0;A(2*i,2*j)=0;
                B(2*i-1,2*j-1)=0;B(2*i-1,2*j)=0;B(2*i,2*j-1)=1;B(2*i,2*j)=1;
            end
            if(random>1&&random<=2)
                A(2*i-1,2*j-1)=1;A(2*i-1,2*j)=1;A(2*i,2*j-1)=0;A(2*i,2*j)=0;
                B(2*i-1,2*j-1)=0;B(2*i-1,2*j)=0;B(2*i,2*j-1)=0;B(2*i,2*j)=1;
            end
            if(random>2&&random<=3)
                A(2*i-1,2*j-1)=0;A(2*i-1,2*j)=0;A(2*i,2*j-1)=0;A(2*i,2*j)=1;
                B(2*i-1,2*j-1)=1;B(2*i-1,2*j)=1;B(2*i,2*j-1)=0;B(2*i,2*j)=0;
            end
            if(random>3&&random<=4)
                A(2*i-1,2*j-1)=1;A(2*i-1,2*j)=0;A(2*i,2*j-1)=0;A(2*i,2*j)=0;
                B(2*i-1,2*j-1)=0;B(2*i-1,2*j)=0;B(2*i,2*j-1)=0;B(2*i,2*j)=1;
            end
        end
    end
end
imwrite(A,'.\test2\subkey1.png','png'); % 将矩阵A保存为图片
imwrite(B,'.\test2\subkey2.png','png'); % 将矩阵B保存为图片
figure(4); % 创建图形窗口
imshow(A);title('子密钥1'); % 显示子密钥1
figure(5); % 创建另一个图形窗口
```

```
89  imshow(B);title('子密钥2'); % 显示子密钥2
90  I=and(A,B); % 对矩阵A和B进行逻辑与操作
91  figure(6); % 创建另一个图形窗口
92  imshow(I);title('合并后的四倍大小的中等灰度原图'); % 显示合并后的图片
93  imwrite(I,'.\test2\合并后的四倍大小的中等灰度原图.png','png');
94  C=zeros(256,256); % 创建一个全一矩阵C
95  for i=1:height % 遍历图片的每一行
96      for j=1:width % 遍历图片的每一列
97          if((I(2*i-1,2*j-1)==1)||(I(2*i-1,2*j)==1)||(I(2*i,2*j-1)==1)||(I(2*i
                ,2*j)==1))
98              C(i,j)=1; % 如果合并后的图片中有白色像素，则将C中对应位置设为1
99          end
100     end
101 end
102 figure(7); % 创建图形窗口
103 imshow(C);title('经过缩放得到的原图'); % 显示经过处理后得到的原图
104 imwrite(C,'.\test2\经过缩放得到的原图.png','png');
105 input_img = imread('lena.bmp'); % 读取图片
106 input_img = imresize(input_img, [256, 256]); % 调整图片大小为256x256
107 im = rgb2gray(input_img); % 将图片转换为灰度图
108 % 半色调化处理
109 K = im;
110 I = zeros(size(K)); % 创建与K相同大小的零矩阵
111 [height,width]=size(K); % 获取图片的高度和宽度
112 a = 7/16; % 定义误差扩散系数
113 b = 3/16;
114 c = 5/16;
115 d = 1/16;
116 for i = 1:width % 遍历每一列
117     for j = 1:height % 遍历每一行
118         if K(i,j) > 127 % 判断像素值是否大于127
119             I(i,j) = 255; % 将像素值设为255（白色）
120         else
121             I(i,j) = 0; % 将像素值设为0（黑色）
122         end
123         error = (K(i,j) - I(i,j)); % 计算误差
124         % 误差扩散
125         if j > 1 && j < height && i < width
126             K(i,j+1) = K(i,j+1) + error * a;
127             K(i+1,j-1) = K(i+1,j-1) + error * b;
128             K(i+1,j) = K(i+1,j) + error * c;
129             K(i+1,j+1) = K(i+1,j+1) + error * d;
130         end
131     end
132 end
133 % 二值化处理
134 im_bin = imbinarize(im); % 将灰度图转换为二值图
135 % 显示输出图像
```

```
figure(1);
subplot(1,2,1);imshow(I);title("半色调化处理的图像");
imwrite(I,'.\test2\半色调化处理的图像.png','png'); % 保存半色调化处理的图像
subplot(1,2,2);imshow(im_bin);title("二值化处理的图像");
imwrite(im_bin,'.\test2\二值化处理的图像.png','png'); % 保存二值化处理的图像
A=zeros(512,512); % 创建一个全零矩阵A
B=zeros(512,512); % 创建一个全零矩阵B
[height,width]=size(I); % 获取二值化图片的尺寸
for i=1:height % 遍历图片的每一行
    for j=1:width % 遍历图片的每一列
        if(I(i,j)==255) % 如果当前像素是白色的
            random=rand()*4; % 生成一个0到4的随机数
            % 根据随机数的范围，对矩阵A和B进行不同的赋值操作
            if(random>0&&random<=1)
                A(2*i-1,2*j-1)=1;A(2*i-1,2*j)=1;A(2*i,2*j-1)=0;A(2*i,2*j)=0;
                B(2*i-1,2*j-1)=0;B(2*i-1,2*j)=1;B(2*i,2*j-1)=0;B(2*i,2*j)=1;
            end
            if(random>1&&random<=2)
                A(2*i-1,2*j-1)=1;A(2*i-1,2*j)=1;A(2*i,2*j-1)=0;A(2*i,2*j)=0;
                B(2*i-1,2*j-1)=0;B(2*i-1,2*j)=1;B(2*i,2*j-1)=0;B(2*i,2*j)=0;
            end
            if(random>2&&random<3)
                A(2*i-1,2*j-1)=0;A(2*i-1,2*j)=1;A(2*i,2*j-1)=0;A(2*i,2*j)=0;
                B(2*i-1,2*j-1)=1;B(2*i-1,2*j)=1;B(2*i,2*j-1)=0;B(2*i,2*j)=0;
            end
            if(random>3&&random<=4)
                A(2*i-1,2*j-1)=0;A(2*i-1,2*j)=1;A(2*i,2*j-1)=0;A(2*i,2*j)=0;
                B(2*i-1,2*j-1)=0;B(2*i-1,2*j)=1;B(2*i,2*j-1)=0;B(2*i,2*j)=0;
            end
        end
        if(I(i,j)==0) % 如果当前像素是黑色的
            random=rand()*4; % 生成一个0到4的随机数
            % 根据随机数的范围，对矩阵A和B进行不同的赋值操作
            if(random>0&&random<=1)
                A(2*i-1,2*j-1)=1;A(2*i-1,2*j)=1;A(2*i,2*j-1)=0;A(2*i,2*j)=0;
                B(2*i-1,2*j-1)=0;B(2*i-1,2*j)=0;B(2*i,2*j-1)=1;B(2*i,2*j)=1;
            end
            if(random>1&&random<=2)
                A(2*i-1,2*j-1)=1;A(2*i-1,2*j)=1;A(2*i,2*j-1)=0;A(2*i,2*j)=0;
                B(2*i-1,2*j-1)=0;B(2*i-1,2*j)=0;B(2*i,2*j-1)=0;B(2*i,2*j)=1;
            end
            if(random>2&&random<=3)
                A(2*i-1,2*j-1)=0;A(2*i-1,2*j)=0;A(2*i,2*j-1)=0;A(2*i,2*j)=1;
                B(2*i-1,2*j-1)=1;B(2*i-1,2*j)=1;B(2*i,2*j-1)=0;B(2*i,2*j)=0;
            end
            if(random>3&&random<=4)
                A(2*i-1,2*j-1)=1;A(2*i-1,2*j)=0;A(2*i,2*j-1)=0;A(2*i,2*j)=0;
                B(2*i-1,2*j-1)=0;B(2*i-1,2*j)=0;B(2*i,2*j-1)=0;B(2*i,2*j)=1;
```

```
            end
        end
    end
end
imwrite(A,'.\test2\subkey1.png','png'); % 将矩阵A保存为图片
imwrite(B,'.\test2\subkey2.png','png'); % 将矩阵B保存为图片
figure(4); % 创建图形窗口
imshow(A);title('子密钥1'); % 显示子密钥1
figure(5); % 创建另一个图形窗口
imshow(B);title('子密钥2'); % 显示子密钥2
I=and(A,B); % 对矩阵A和B进行逻辑与操作
figure(6); % 创建另一个图形窗口
imshow(I);title('合并后的四倍大小的中等灰度原图'); % 显示合并后的图片
imwrite(I,'.\test2\合并后的四倍大小的中等灰度原图.png','png');
C=zeros(256,256); % 创建一个全一矩阵C
for i=1:height % 遍历图片的每一行
    for j=1:width % 遍历图片的每一列
        if((I(2*i-1,2*j-1)==1)||(I(2*i-1,2*j)==1)||(I(2*i,2*j-1)==1)||(I(2*i
            ,2*j)==1))
            C(i,j)=1; % 如果合并后的图片中有白色像素，则将C中对应位置设为1
        end
    end
end
figure(7); % 创建图形窗口
imshow(C);title('经过缩放得到的原图'); % 显示经过处理后得到的原图
imwrite(C,'.\test2\经过缩放得到的原图.png','png');
```

(三) 彩色图像的可视密钥分享方案

Listing 3: (t,n) 可视密钥分享方案代码

```
L=imread('lena.bmp'); % 读取图片
L=imresize(L, [256, 256]); % 调整图片大小
[height,width]=size(L); % 获取原图像大小
width=width/3; % 排除颜色通道数
for t=1:3 % 对每个颜色通道进行操作
    for i=1: height % 遍历高度
        for j=1: width % 遍历宽度
            if L(i,j,t)>127 % 判断像素值是否大于127
                out=255; % 设为白色
            else
                out=0; % 设为黑色
            end
            error=L(i,j)-out; % 计算误差
            if j>1 && i<height && j<width % 边界检查
                L(i,j+1,t)=L(i,j+1,t)+error*7/16.0; % 右侧像素误差分配
                L(i+1,j,t)=L(i+1,j,t)+error*5/16.0; % 下方像素误差分配
                L(i+1,j-1,t)=L(i+1,j-1,t)+error*3/16.0; % 左下方像素误差分配
```

```
                L(i+1,j+1,t)=L(i+1,j+1,t)+error*1/16.0; % 右下方像素误差分配
                L(i,j,t)=out; % 更新当前像素值
            else
                L(i,j,t)=out; % 更新当前像素值
            end
        end
    end
end
figure(1);
imshow(L);title('半色调处理'); % 显示处理后的图片
imwrite(L,'.\test3\半色调处理后的图片.png','png'); % 保存处理后的图片
red=L(:,:,1); % 获取红色通道
green=L(:,:,2); % 获取绿色通道
blue=L(:,:,3); % 获取蓝色通道
sub_height=2*height;sub_width=2*width; % 计算子图大小
A=zeros(sub_height,sub_width); % 初始化子图A
B=zeros(sub_height,sub_width); % 初始化子图B
for t=1:3 % 对每个颜色通道进行操作
    for i=1: height % 遍历高度
        for j=1: width % 遍历宽度
            if(L(i,j,t)==255) % 如果像素值为255
                random= round(rand()*3); % 生成随机数
                switch random % 根据随机数选择模式
                    case 0
                        A(2*i-1,2*j-1,t)=1;A(2*i-1,2*j,t)=1;A(2*i,2*j-1,t)=0;
                            A(2*i,2*j,t)=0;
                        B(2*i-1,2*j-1,t)=0;B(2*i-1,2*j,t)=1;B(2*i,2*j-1,t)=0;
                            B(2*i,2*j,t)=1;
                    case 1
                        A(2*i-1,2*j-1,t)=1;A(2*i-1,2*j,t)=1;A(2*i,2*j-1,t)=0;
                            A(2*i,2*j,t)=0;
                        B(2*i-1,2*j-1,t)=0;B(2*i-1,2*j,t)=1;B(2*i,2*j-1,t)=0;
                            B(2*i,2*j,t)=0;
                    case 2
                        A(2*i-1,2*j-1,t)=0;A(2*i-1,2*j,t)=1;A(2*i,2*j-1,t)=0;
                            A(2*i,2*j,t)=0;
                        B(2*i-1,2*j-1,t)=1;B(2*i-1,2*j,t)=1;B(2*i,2*j-1,t)=0;
                            B(2*i,2*j,t)=0;
                    case 3
                        A(2*i-1,2*j-1,t)=0;A(2*i-1,2*j,t)=1;A(2*i,2*j-1,t)=0;
                            A(2*i,2*j,t)=0;
                        B(2*i-1,2*j-1,t)=0;B(2*i-1,2*j,t)=1;B(2*i,2*j-1,t)=0;
                            B(2*i,2*j,t)=0;
                end
            end
            if(L(i,j,t)==0) % 如果像素值为0
                random= round(rand()*3); % 生成随机数
                switch random % 根据随机数选择模式
```

```
                    case 0
                        A(2*i-1,2*j-1,t)=1;A(2*i-1,2*j,t)=1;A(2*i,2*j-1,t)=0;
                            A(2*i,2*j,t)=0;
                        B(2*i-1,2*j-1,t)=0;B(2*i-1,2*j,t)=0;B(2*i,2*j-1,t)=1;
                            B(2*i,2*j,t)=1;
                    case 1
                        A(2*i-1,2*j-1,t)=1;A(2*i-1,2*j,t)=1;A(2*i,2*j-1,t)=0;
                            A(2*i,2*j,t)=0;
                        B(2*i-1,2*j-1,t)=0;B(2*i-1,2*j,t)=0;B(2*i,2*j-1,t)=0;
                            B(2*i,2*j,t)=1;
                    case 2
                        A(2*i-1,2*j-1,t)=0;A(2*i-1,2*j,t)=0;A(2*i,2*j-1,t)=0;
                            A(2*i,2*j,t)=1;
                        B(2*i-1,2*j-1,t)=1;B(2*i-1,2*j,t)=1;B(2*i,2*j-1,t)=0;
                            B(2*i,2*j,t)=0;
                    case 3
                        A(2*i-1,2*j-1,t)=1;A(2*i-1,2*j,t)=0;A(2*i,2*j-1,t)=0;
                            A(2*i,2*j,t)=0;
                        B(2*i-1,2*j-1,t)=0;B(2*i-1,2*j,t)=0;B(2*i,2*j-1,t)=0;
                            B(2*i,2*j,t)=1;
                end
            end
        end
    end
end
figure(2);
imshow(A);title('子图1'); % 显示子图A
imwrite(A,'.\test3\subkey1.png','png'); % 保存子图A
figure(5);
imshow(B);title('子图2'); % 显示子图B
imwrite(B,'.\test3\subkey2.png','png'); % 保存子图B
overlap=zeros(2*height,2*width); % 初始化合并后的图片
for t=1:3 % 对每个颜色通道进行操作
    overlap(:,:,t)=and(A(:,:,t),B(:,:,t)); % 合并子图A和B
end
figure(3);
imshow(overlap);title('合并后的图片'); % 显示合并后的图片
imwrite(overlap,'.\test3\合并后的图片.png','png'); % 保存合并后的图片
minipic=zeros(height,width); % 初始化缩小处理后的复原图
for t=1:3 % 对每个颜色通道进行操作
    for i=1:height % 遍历高度
        for j=1:width % 遍历宽度
            if(overlap(2*i-1,2*j-1,t)==1||overlap(2*i-1,2*j,t)==1||overlap(2*
                i,2*j-1,t)==1||overlap(2*i,2*j,t)==1) % 判断像素是否应该为1
                minipic(i,j,t)=1; % 设为1
            else
                minipic(i,j,t)=0; % 设为0
            end
```

```
        end
    end
end
figure(4);
imshow(minipic);title('缩小处理后的复原图'); % 显示缩小处理后的复原图
imwrite(minipic,'.\test3\缩小处理后的复原图.png','png'); % 保存缩小处理后的复原图
```

(四) (t,n) 可视密钥分享方案

Listing 4: (t,n) 可视密钥分享方案代码

```
% 读取图像
path = 'lena.bmp';
img = imread(path);
img_gray = rgb2gray(img);
img_flattened = img_gray(:);
% disp(img_flattened);

% 多项式生成
n = 5;
r = 3;
gen_imgs = polynomial(img_flattened, n, r);

% 保存生成的图像
for i = 1:n
    img_save = reshape(gen_imgs(i,:), size(img_gray));
    imwrite(uint8(img_save), strcat('test2_', num2str(i), '.jpeg'));
end

% 解码
index = 1:r;
origin_img = decode(gen_imgs(1:r,:), index, r, n);
imwrite(uint8(reshape(origin_img, size(img_gray))), 'test2_origin.jpeg');

% 多项式生成函数
function gen_imgs = polynomial(img, n, r)
    % 获取图像中像素的数量
    num_pixels = size(img, 1);
    % disp(num_pixels);

    % 随机生成多项式的系数矩阵, 范围在 [0, 251)
    coef = randi([0, 250], num_pixels, r - 1);
    % disp(coef);

    % 存储生成的图像序列
    gen_imgs = zeros(n, num_pixels);

```

```
    % 对于每个要生成的多项式
    for i = 1:n
        % 构建基向量 [i^1, i^2, ..., i^(r-1)]
        base = zeros(1, r - 1); % 初始化基函数数组
        for j = 1:r - 1
            base(j) = i^j; % 计算每个指数值并存储到数组中
        end
        % 计算每个像素点对应的基函数值, 并将其加到原始图像数据上
        base = coef * base';
        img = double(img);
        img_ = img + mod(base, 251);

        % 对结果取模, 确保像素值在合理范围内
        img_ = mod(img_, 251);

        % 将处理后的图像存储到生成的图像序列中
        gen_imgs(i, :) = img_';
    end
end


% 解码函数
function origin_img = decode(imgs, index, r, ~)
    assert(size(imgs, 1) >= r);
    dim = size(imgs, 2);
    origin_img = zeros(1, dim);

    for i = 1:dim
        if mod(i, 10000) == 0
            disp(['Decoding ', num2str(i), 'th pixel']);
        end
        y = imgs(:, i)';
        pixel = mod(lagrange(index,y,0), 251);
        origin_img(i) = pixel;
    end
end
```

（五） 叠像术

Listing 5: 叠像术代码

```
% And函数
function[X] = And(Y,Z)
for i = 1:256*2
    for j = 1:256*2
        if(Y(i,j)==255&&Z(i,j)==255)
            X(i,j)=255;
        else
```

```
            X(i,j)=0;
        end
    end
end
% Binary_Hide函数
function[Y,Z] = Binary_Hide(A, B, C)
for i = 1: 256
    for j = 1: 256
        if(A(i,j)==0) %黑色情况
            if( B(i,j)==0 && C(i,j)==0)
                random=rand()*4;
                if(random>=0&&random<1)
                    Y(2*i-1,2*j-1)=0;Y(2*i-1,2*j)=255;Y(2*i,2*j-1)=0;Y(2*i,2*
                        j)=0;
                    Z(2*i-1,2*j-1)=255;Z(2*i-1,2*j)=0;Z(2*i,2*j-1)=0;Z(2*i,2*
                        j)=0;
                elseif(random>=1&&random<2)
                    Y(2*i-1,2*j-1)=0;Y(2*i-1,2*j)=0;Y(2*i,2*j-1)=0;Y(2*i,2*j)
                        =255;
                    Z(2*i-1,2*j-1)=0;Z(2*i-1,2*j)=0;Z(2*i,2*j-1)=255;Z(2*i,2*
                        j)=0;
                elseif(random>=2&&random<3)
                    Y(2*i-1,2*j-1)=255;Y(2*i-1,2*j)=0;Y(2*i,2*j-1)=0;Y(2*i,2*
                        j)=0;
                    Z(2*i-1,2*j-1)=0;Z(2*i-1,2*j)=255;Z(2*i,2*j-1)=0;Z(2*i,2*
                        j)=0;
                else
                    Y(2*i-1,2*j-1)=0;Y(2*i-1,2*j)=0;Y(2*i,2*j-1)=255;Y(2*i,2*
                        j)=0;
                    Z(2*i-1,2*j-1)=0;Z(2*i-1,2*j)=0;Z(2*i,2*j-1)=0;Z(2*i,2*j)
                        =255;
                end
            end
            if( B(i,j)==0 && C(i,j)==255)
                random=rand()*4;
                if(random>=0&&random<1)
                    Y(2*i-1,2*j-1)=255;Y(2*i-1,2*j)=0;Y(2*i,2*j-1)=0;Y(2*i,2*
                        j)=0;
                    Z(2*i-1,2*j-1)=0;Z(2*i-1,2*j)=0;Z(2*i,2*j-1)=255;Z(2*i,2*
                        j)=255;
                elseif(random>=1&&random<2)
                    Y(2*i-1,2*j-1)=0;Y(2*i-1,2*j)=255;Y(2*i,2*j-1)=0;Y(2*i,2*
                        j)=0;
                    Z(2*i-1,2*j-1)=255;Z(2*i-1,2*j)=0;Z(2*i,2*j-1)=255;Z(2*i
                        ,2*j)=0;
                elseif(random>=2&&random<3)
                    Y(2*i-1,2*j-1)=0;Y(2*i-1,2*j)=0;Y(2*i,2*j-1)=255;Y(2*i,2*
                        j)=0;
```

```
43                  Z(2*i-1,2*j-1)=255;Z(2*i-1,2*j)=255;Z(2*i,2*j-1)=0;Z(2*i
                        ,2*j)=0;
44              else
45                  Y(2*i-1,2*j-1)=0;Y(2*i-1,2*j)=0;Y(2*i,2*j-1)=0;Y(2*i,2*j)
                        =255;
46                  Z(2*i-1,2*j-1)=0;Z(2*i-1,2*j)=255;Z(2*i,2*j-1)=255;Z(2*i
                        ,2*j)=0;
47              end
48          end
49          if( B(i,j)==255 && C(i,j)==0)
50              random=rand()*4;
51              if(random>=0&&random<1)
52                  Y(2*i-1,2*j-1)=255;Y(2*i-1,2*j)=255;Y(2*i,2*j-1)=0;Y(2*i
                        ,2*j)=0;
53                  Z(2*i-1,2*j-1)=0;Z(2*i-1,2*j)=0;Z(2*i,2*j-1)=0;Z(2*i,2*j)
                        =255;
54              elseif(random>=1&&random<2)
55                  Y(2*i-1,2*j-1)=0;Y(2*i-1,2*j)=255;Y(2*i,2*j-1)=0;Y(2*i,2*
                        j)=255;
56                  Z(2*i-1,2*j-1)=255;Z(2*i-1,2*j)=0;Z(2*i,2*j-1)=0;Z(2*i,2*
                        j)=0;
57              elseif(random>=2&&random<3)
58                  Y(2*i-1,2*j-1)=0;Y(2*i-1,2*j)=0;Y(2*i,2*j-1)=255;Y(2*i,2*
                        j)=255;
59                  Z(2*i-1,2*j-1)=0;Z(2*i-1,2*j)=255;Z(2*i,2*j-1)=0;Z(2*i,2*
                        j)=0;
60              else
61                  Y(2*i-1,2*j-1)=255;Y(2*i-1,2*j)=0;Y(2*i,2*j-1)=0;Y(2*i,2*
                        j)=255;
62                  Z(2*i-1,2*j-1)=0;Z(2*i-1,2*j)=0;Z(2*i,2*j-1)=255;Z(2*i,2*
                        j)=0;
63              end
64          end
65          %
                ------------------------------------------------------------%
66          %
                ------------------------------------------------------------%
67          if( B(i,j)==255 && C(i,j)==255)
68              random=rand()*4;
69              if(random>=0&&random<1)
70                  Y(2*i-1,2*j-1)=255;Y(2*i-1,2*j)=255;Y(2*i,2*j-1)=0;Y(2*i
                        ,2*j)=0;
71                  Z(2*i-1,2*j-1)=0;Z(2*i-1,2*j)=0;Z(2*i,2*j-1)=255;Z(2*i,2*
                        j)=255;
72              elseif(random>=1&&random<2)
73                  Y(2*i-1,2*j-1)=0;Y(2*i-1,2*j)=255;Y(2*i,2*j-1)=0;Y(2*i,2*
```

```
                    j)=255;
74                  Z(2*i-1,2*j-1)=255;Z(2*i-1,2*j)=0;Z(2*i,2*j-1)=255;Z(2*i
                        ,2*j)=0;
75              elseif(random>=2&&random<3)
76                  Y(2*i-1,2*j-1)=0;Y(2*i-1,2*j)=0;Y(2*i,2*j-1)=255;Y(2*i,2*
                        j)=255;
77                  Z(2*i-1,2*j-1)=255;Z(2*i-1,2*j)=255;Z(2*i,2*j-1)=0;Z(2*i
                        ,2*j)=0;
78              else
79                  Y(2*i-1,2*j-1)=255;Y(2*i-1,2*j)=0;Y(2*i,2*j-1)=0;Y(2*i,2*
                        j)=255;
80                  Z(2*i-1,2*j-1)=0;Z(2*i-1,2*j)=255;Z(2*i,2*j-1)=255;Z(2*i
                        ,2*j)=0;
81              end
82          end
83      end
84      if(A(i,j)==255) %白色情况
85          if( B(i,j)==0 && C(i,j)==0)
86              random=rand()*4;
87              if(random>=0&&random<1)
88                  Y(2*i-1,2*j-1)=0;Y(2*i-1,2*j)=0;Y(2*i,2*j-1)=0;Y(2*i,2*j)
                        =255;
89                  Z(2*i-1,2*j-1)=0;Z(2*i-1,2*j)=0;Z(2*i,2*j-1)=0;Z(2*i,2*j)
                        =255;
90              elseif(random>=1&&random<2)
91                  Y(2*i-1,2*j-1)=0;Y(2*i-1,2*j)=0;Y(2*i,2*j-1)=255;Y(2*i,2*
                        j)=0;
92                  Z(2*i-1,2*j-1)=0;Z(2*i-1,2*j)=0;Z(2*i,2*j-1)=255;Z(2*i,2*
                        j)=0;
93              elseif(random>=2&&random<3)
94                  Y(2*i-1,2*j-1)=255;Y(2*i-1,2*j)=0;Y(2*i,2*j-1)=0;Y(2*i,2*
                        j)=0;
95                  Z(2*i-1,2*j-1)=255;Z(2*i-1,2*j)=0;Z(2*i,2*j-1)=0;Z(2*i,2*
                        j)=0;
96              else
97                  Y(2*i-1,2*j-1)=0;Y(2*i-1,2*j)=255;Y(2*i,2*j-1)=0;Y(2*i,2*
                        j)=0;
98                  Z(2*i-1,2*j-1)=0;Z(2*i-1,2*j)=255;Z(2*i,2*j-1)=0;Z(2*i,2*
                        j)=0;
99              end
100         end
101         if( B(i,j)==0 && C(i,j)==255)
102             random=rand()*4;
103             if(random>=0&&random<1)
104                 Y(2*i-1,2*j-1)=0;Y(2*i-1,2*j)=0;Y(2*i,2*j-1)=0;Y(2*i,2*j)
                        =255;
105                 Z(2*i-1,2*j-1)=0;Z(2*i-1,2*j)=0;Z(2*i,2*j-1)=255;Z(2*i,2*
                        j)=255;
```

```
106                 elseif(random>=1&&random<2)
107                     Y(2*i-1,2*j-1)=0;Y(2*i-1,2*j)=0;Y(2*i,2*j-1)=255;Y(2*i,2*
                            j)=0;
108                     Z(2*i-1,2*j-1)=0;Z(2*i-1,2*j)=255;Z(2*i,2*j-1)=255;Z(2*i
                            ,2*j)=0;
109                 elseif(random>=2&&random<3)
110                     Y(2*i-1,2*j-1)=0;Y(2*i-1,2*j)=255;Y(2*i,2*j-1)=0;Y(2*i,2*
                            j)=0;
111                     Z(2*i-1,2*j-1)=255;Z(2*i-1,2*j)=255;Z(2*i,2*j-1)=0;Z(2*i
                            ,2*j)=0;
112                 else
113                     Y(2*i-1,2*j-1)=255;Y(2*i-1,2*j)=0;Y(2*i,2*j-1)=0;Y(2*i,2*
                            j)=0;
114                     Z(2*i-1,2*j-1)=255;Z(2*i-1,2*j)=0;Z(2*i,2*j-1)=255;Z(2*i
                            ,2*j)=0;
115                 end
116             end
117             if( B(i,j)==255 && C(i,j)==255)
118                 random=rand()*4;
119                 if(random>=0&&random<1)
120                     Y(2*i-1,2*j-1)=255;Y(2*i-1,2*j)=0;Y(2*i,2*j-1)=255;Y(2*i
                            ,2*j)=0;
121                     Z(2*i-1,2*j-1)=0;Z(2*i-1,2*j)=0;Z(2*i,2*j-1)=255;Z(2*i,2*
                            j)=255;
122                 elseif(random>=1&&random<2)
123                     Y(2*i-1,2*j-1)=0;Y(2*i-1,2*j)=0;Y(2*i,2*j-1)=255;Y(2*i,2*
                            j)=255;
124                     Z(2*i-1,2*j-1)=0;Z(2*i-1,2*j)=255;Z(2*i,2*j-1)=255;Z(2*i
                            ,2*j)=0;
125                 elseif(random>=2&&random<3)
126                     Y(2*i-1,2*j-1)=255;Y(2*i-1,2*j)=0;Y(2*i,2*j-1)=0;Y(2*i,2*
                            j)=255;
127                     Z(2*i-1,2*j-1)=255;Z(2*i-1,2*j)=255;Z(2*i,2*j-1)=0;Z(2*i
                            ,2*j)=0;
128                 else
129                     Y(2*i-1,2*j-1)=255;Y(2*i-1,2*j)=255;Y(2*i,2*j-1)=0;Y(2*i
                            ,2*j)=0;
130                     Z(2*i-1,2*j-1)=255;Z(2*i-1,2*j)=0;Z(2*i,2*j-1)=255;Z(2*i
                            ,2*j)=0;
131                 end
132             end
133             if( B(i,j)==255 && C(i,j)==0)
134                 random=rand()*4;
135                 if(random>=0&&random<1)
136                     Y(2*i-1,2*j-1)=255;Y(2*i-1,2*j)=255;Y(2*i,2*j-1)=0;Y(2*i
                            ,2*j)=0;
137                     Z(2*i-1,2*j-1)=255;Z(2*i-1,2*j)=0;Z(2*i,2*j-1)=0;Z(2*i,2*
                            j)=0;
```

```
                elseif(random>=1&&random<2)
                    Y(2*i-1,2*j-1)=0;Y(2*i-1,2*j)=0;Y(2*i,2*j-1)=255;Y(2*i,2*
                        j)=255;
                    Z(2*i-1,2*j-1)=0;Z(2*i-1,2*j)=0;Z(2*i,2*j-1)=255;Z(2*i,2*
                        j)=0;
                elseif(random>=2&&random<3)
                    Y(2*i-1,2*j-1)=255;Y(2*i-1,2*j)=0;Y(2*i,2*j-1)=0;Y(2*i,2*
                        j)=255;
                    Z(2*i-1,2*j-1)=0;Z(2*i-1,2*j)=0;Z(2*i,2*j-1)=0;Z(2*i,2*j)
                        =255;
                else
                    Y(2*i-1,2*j-1)=255;Y(2*i-1,2*j)=255;Y(2*i,2*j-1)=0;Y(2*i
                        ,2*j)=0;
                    Z(2*i-1,2*j-1)=0;Z(2*i-1,2*j)=255;Z(2*i,2*j-1)=0;Z(2*i,2*
                        j)=0;
                end
            end
        end
    end
end
% Halftone函数
function [im_out] = Halftone(im_gray)

% 定义点阵单元尺寸和比例因子
% cell_size = 8;  % 点阵单元尺寸
scale_factor = 255;  % 比例因子

% 初始化输出图像和误差矩阵
im_out = zeros(size(im_gray),'uint8');

% 执行误差扩散算法
for i = 1:size(im_gray, 1)
    for j = 1:size(im_gray, 2)
        % 计算当前像素值和最近的阈值
        old_gray = im_gray(i, j);
        new_gray = round(old_gray / scale_factor) * scale_factor;
        im_out(i, j) = new_gray;

        % 计算误差并进行扩散
        quant_error = old_gray - new_gray ;
        if j < size(im_gray, 2)
            im_gray(i, j+1) = im_gray(i, j+1) + quant_error * 7 / 16;
        end
        if i < size(im_gray, 1) && j > 1
            im_gray(i+1, j-1) = im_gray(i+1, j-1) + quant_error * 3 / 16;
        end
        if i < size(im_gray, 1)
            im_gray(i+1, j) = im_gray(i+1, j) + quant_error * 5 / 16;
```

```
        end
        if i < size(im_gray, 1) && j < size(im_gray, 2)
            im_gray(i+1, j+1) = im_gray(i+1, j+1) + quant_error * 1 / 16;
        end
    end
end
% 读取图片
A=imread('lena.bmp');
B=imread('b.jpg');
C=imread('c.jpg');


A1=zeros(256,'uint8'); A2=zeros(256,'uint8'); A3=zeros(256,'uint8');
B1=zeros(256,'uint8'); B2=zeros(256,'uint8'); B3=zeros(256,'uint8');
C1=zeros(256,'uint8'); C2=zeros(256,'uint8'); C3=zeros(256,'uint8');

for i = 1:256
    for j = 1:256
        for k = 1:3
            if k==1
                A1(i,j)=A(i,j,k);
                B1(i,j)=B(i,j,k);
                C1(i,j)=C(i,j,k);
            end
            if k==2
                A2(i,j)=A(i,j,k);
                B2(i,j)=B(i,j,k);
                C2(i,j)=C(i,j,k);
            end
            if k==3
                A3(i,j)=A(i,j,k);
                B3(i,j)=B(i,j,k);
                C3(i,j)=C(i,j,k);
            end
        end
    end
end
[A1] = Halftone(A1);[A2] = Halftone(A2);[A3] = Halftone(A3);
[B1] = Halftone(B1);[B2] = Halftone(B2);[B3] = Halftone(B3);
[C1] = Halftone(C1);[C2] = Halftone(C2);[C3] = Halftone(C3);


X=zeros(512,'uint8'); Y=zeros(512,'uint8'); Z=zeros(512,'uint8');

X1=zeros(512,'uint8'); Y1=zeros(512,'uint8'); Z1=zeros(512,'uint8');
X2=zeros(512,'uint8'); Y2=zeros(512,'uint8'); Z2=zeros(512,'uint8');
X3=zeros(512,'uint8'); Y3=zeros(512,'uint8'); Z3=zeros(512,'uint8');

```

```
[Y1,Z1]= Binary_Hide(A1,B1,C1);
[Y2,Z2]= Binary_Hide(A2,B2,C2);
[Y3,Z3]= Binary_Hide(A3,B3,C3);


X1 = And(Y1,Z1);
X2 = And(Y2,Z2);
X3 = And(Y3,Z3);


% 三个维度重新恢复为RGB图像

for i = 1:512
    for j = 1:512
        for k = 1:3
            if(k==1)
            X(i,j,k)=X1(i,j);
            Y(i,j,k)=Y1(i,j);
            Z(i,j,k)=Z1(i,j);
            end
            if(k==2)
            X(i,j,k)=X2(i,j);
            Y(i,j,k)=Y2(i,j);
            Z(i,j,k)=Z2(i,j);
            end
            if(k==3)
            X(i,j,k)=X3(i,j);
            Y(i,j,k)=Y3(i,j);
            Z(i,j,k)=Z3(i,j);
            end
        end
    end
end

subplot(1,3,1);
imshow(X);
subplot(1,3,2);
imshow(Y);
subplot(1,3,3);
imshow(Z);
```