南開大學
Nankai University

网络空间安全学院
信息隐藏技术报告

姓名：姜涵 李晓彤 丁奕可 胡亚飞 孙蕗

学号：2113630 2112075 2113345 2111690 2112060

专业：信息安全

指导教师：李朝晖

2024 年 5 月 27 日

目录

1 论文选择

在本次实验中，我们选择了《基于自适应哈夫曼编码的密文可逆信息隐藏算法》这篇学术论文作为研究对象，该论文发表于《计算机学报》2021 年第 4 期，作者团队包括吴友情、郭玉堂、汤进、罗斌和殷赵霞，他们分别来自安徽大学多模态认知计算安徽省重点实验室和合肥师范学院计算机学院。本文针对云存储与隐私保护背景下的密文域可逆信息隐藏技术进行深入探讨，旨在提高嵌入率和安全性，解决现有方法在有效载荷和图像质量间平衡不佳的问题。

2 文章研读

2.1 主要动机

这篇文章的主要动机是针对当前密文域可逆信息隐藏技术中有效载荷低和安全性不足的问题，提出一种改进方法，旨在提升信息隐藏容量和安全性的同时确保无损恢复原始图像质量。文章关注在云存储和隐私保护需求背景下，如何在加密图像中嵌入秘密信息而又不影响图像内容的正常解密与还原，且嵌入信息能够准确无误提取。

2.2 主要贡献

文章基于自适应的哈夫曼编码方案设计了一种 RRBE 的 RDHEI 算法，主要贡献包括以下 3 个方面；

1. 可以探索更大的数据嵌入空间，从而获得更高的有效载荷；
2. 根据概率分布分配不同长度的码字，可使平均码长最短，实现了最优编码的选择；
3. 对不同的图像采用不同的码字，增强了信息嵌入过程的安全性

2.3 主要方法

在本文提出的整个信息隐藏流程中，共有三个角色。

总体来说，原始图像所有者可以将图像进行加密。利用自然图像相邻像素间的相关性对原始明文图像进行像素值预测。从最高到最低有效位，每对原始像素值和预测像素值的相同比特位被存储到位图中并进行自适应哈夫曼编码标记。利用流密码对图像进行加密，将已编码压缩的位图嵌入到加密图像中。

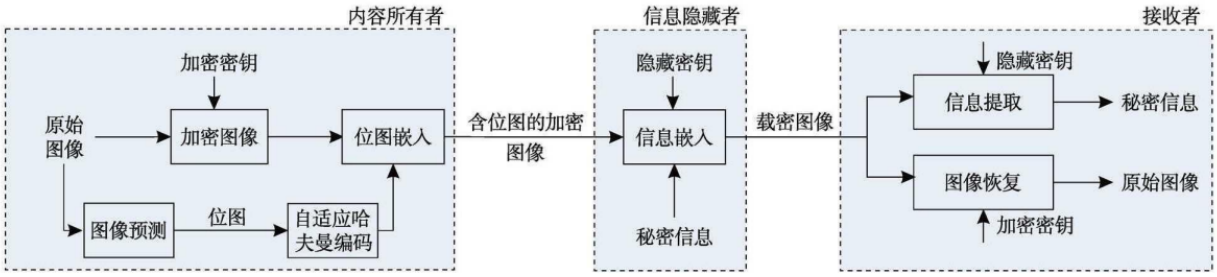
信息隐藏者获得了含有位图的加密图像后，在腾出的空间中，根据隐藏密钥，通过位替换自适应地嵌入秘密信息。

合法接收者拥有数据隐藏密钥，再加上哈夫曼编码的可逆性，他就可以利用密钥对图像进行解密，得到原始秘密信息。如果还拥有图像加密密钥，就可以无损地恢复原始明文图像。

整个流程以及算法框架如下图所示，接下来按照步骤的前后顺序具体说明主要方法。

2.3.1 图像预测

采用中值预测器 MED 对原始明文图像进行像素值预测。图像的第一行和第一列像素作为参考，在像素值预测过程中保持不变，也不参与后续的哈夫曼编码标记。

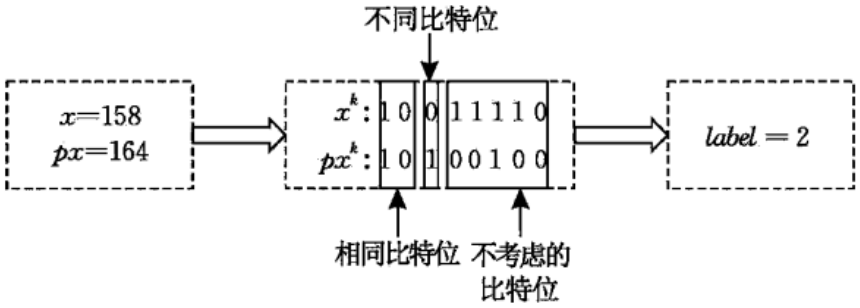


a,b,c 分别为像素 x 的左上、上、左三个相邻像素，预测值 px 计算公式如下:

$$px = \begin{cases} max(b,c), & a \le min(b,c) \\ min(b,c), & a \ge max(b,c) \\ b+c-a, & \text{其他} \end{cases}$$

2.3.2 自适应的哈夫曼编码

得到了像素预测值之后，将原始和预测像素值分别转换为 8 位二进制形式，从最高有效位到最低有效位，逐位比较 x 和 px 的二进制位，直到某位二者不相同，比较停止，后面剩下的都是不需要考虑的比特位。将前面相同的比特位数作为该像素点的标记值。参考像素不参与标记，记为-1 即可。



将每个像素逐一标记后，统计各个标记值的分布概率，出现越多的标记值哈夫曼编码的码长越短，对它们进行自适应哈夫曼码字编码。

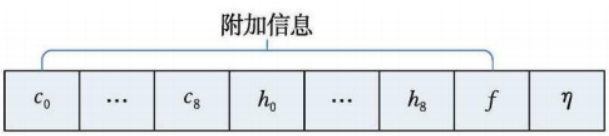
2.3.3 图像加密

利用图像加密密钥 k_e，生成一个和原始图像 I 同样大小的伪随机矩阵 R。将 I 和 R 都转化为二进制形式 $x^k(i,j)$ 和 $r^k(i,j)$。对二者进行按位异或操作，再转换回十进制形式，作为加密结果。

2.3.4 位图嵌入

位图信息的存储结构如下图所示。其中 c_t 为码长的二进制形式，h_t 为哈夫曼编码结果，即码字。Lm 为整个位图哈夫曼编码的总长度。f 为 Lm 的二进制形式；η 为位图的具体信息，共 Lm 位。

$$Lm = \sum_{t=0}^{8}(g_t \times \lambda_t)$$



先将一部分位图信息存储到第一行第一列的参考像素中。然后将另一部分位图信息和暂时挪开的参考像素嵌入到其他非参考像素部分。嵌入公式如下:

$$x_e^{'}(i,j)=\begin{cases}x_e(i,j)mod2^{7-t}+\sum_{s=0}^{t}(b_s\times 2^{7-s}), & 0\le t\le 6\\ \sum_{s=1}^{8}(b_s\times 2^{8-s}), & 7\le t\le 8\end{cases}$$

其中 b_s 为要嵌入的信息，t 为当前像素的位图标记值。嵌入位图信息后，得到新的加密图像 $I_e^{'}$。

2.3.5 信息嵌入

在嵌入秘密信息前，使用数据隐藏密钥 k_d 对其进行加密，仍然按照上面的公式，将秘密信息嵌入到加密图像 $I_e^{'}$ 的预留空间（嵌入位图信息和参考像素后剩余的空间）中，生成最终的载密图像 I_{ew}。

2.3.6 信息提取与图像恢复

首先提取参考像素位置上的位图信息，再按照从左往右、从上到下、先行后列的顺序，提取剩余的位图信息和参考像素。剩下的就是嵌入的秘密信息了。如果接收者拥有数据隐藏密钥，即可解密秘密信息。如果还拥有图像加密密钥，那么可以生成随机矩阵，对图像进行解密。过程和之前加密的过程类似，通过预测值重构像素值的公示如下:

$$x(i,j)=\begin{cases}g_1+g_2+x_{e\omega}^{''}(i,j)mod2^{7-t}, & 0\le t\le 7\\ px(i,j), & t=8\end{cases}$$

2.4 结果分析

以 lena 图像为例，经过整个流程得到了加密图像、嵌入位图信息后的图像、载密图像和恢复后的图像。

2.4.1 算法安全性

整个流程中有两次加密。对大小为 $m\times n$ 的图像进行加密的时候，伪随机矩阵 R 中二进制序列长度为 $m\times n\times 8$，其中每一位都可能是 0 或 1，所以共有 $2^{m\times n\times 8}$ 种可能。在没有加密密钥 k_e 的情况下，解密的概率是极低的，从过程中的三张加密图像中获取原始图像内容是很难的，从而可以保护原始图像。

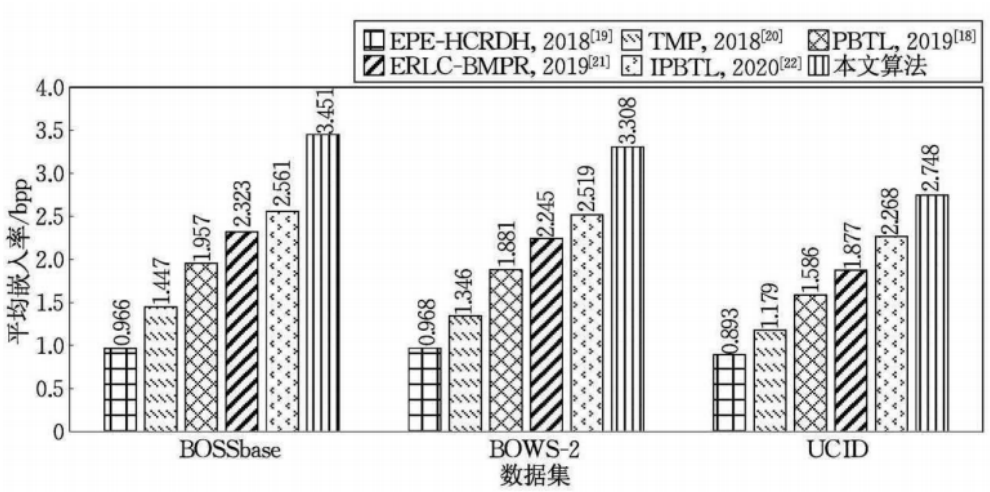
对长度为 num 的秘密信息进行加密，k_d 的密钥空间为 2^{num}，可以确保秘密信息的安全。

2.4.2 性能分析

本文中的算法考虑像素标记值的概率分布，根据标记值出现的多少采用自适应的哈夫曼编码，对不同的图像采用的是不同的码字，而不是固定的，这样不仅可以使平均码长最短，更充分地压缩位图，留下的剩余空间更多，从而提高净嵌入容量和嵌入率；还可以增强位图信息嵌入的安全性。

与 MPHC 算法相比，本文提出的算法无论是在纹理平滑的图像上还是在纹理粗糙的图像上，嵌入率和有效载荷都有所提高。并且两种算法在拥有图像加密密钥的情况下，都可以对原始明文图像进行无损恢复。

与其他同类算法对比，各自都设置最佳参数，本文提出的算法在大部分测试图像上都获得了最高的嵌入率，表现出更好的性能。



2.4.3 运行时间分析

在这三个角色中，内容所有者所完成的图像预测、哈夫曼编码、位图和信息嵌入几个部分的运行时间最直接地影响用户体验。对比参考文献中的各个算法，本文算法和 MPHC 算法的运行时间较长，因为在对图像加密前需要对多位 MSB 进行位图标记，但依然在用户的可接受范围内。

3 算法复现

3.1 图像预测

在图像预测阶段，我们使用中值预测器 MED 对原始明文图像进行像素值预测。算法核心部分通过比较当前像素点及其周围像素的值来预测当前像素点的值，并将预测值存储到 origin_PV_I 中。

算法首先遍历原始图像中除了前 ref_x 行和前 ref_y 列之外的所有像素点。对于每个像素点，算法根据其周围的像素值进行预测，然后将预测值存储到 origin_PV_I 中。预测的方法是根据当前像素点上方、左上方和左方的像素值进行比较，然后根据相应的规则计算出预测值，并将其存储到 origin_PV_I 中。

详细代码如下：

```
function [origin_PV_I] = Predictor_Value(origin_I,ref_x,ref_y)
    % 函数说明：计算origin_I的预测值
    % 输入：origin_I（原始图像）,ref_x,ref_y（参考像素的行列数）
```

```
    % 输出: origin_PV_I (原始图像的预测值)
    [row,col] = size(origin_I); %计算origin_I的行列值
    origin_PV_I = origin_I;  %构建存储origin_I预测值的容器
    for i=ref_x+1:row  %前ref_x行作为参考像素, 不用预测
        for j=ref_y+1:col  %前ref_y列作为参考像素, 不用预测
            a = origin_I(i-1,j);
            b = origin_I(i-1,j-1);
            c = origin_I(i,j-1);
            if b <= min(a,c)
                origin_PV_I(i,j) = max(a,c);
            elseif b >= max(a,c)
                origin_PV_I(i,j) = min(a,c);
            else
                origin_PV_I(i,j) = a + c - b;
            end
        end
    end
end
```

3.2 自适应的哈夫曼编码

我们采用自适应哈夫曼编码对原始图像的每个像素值进行标记，并将像素值的标记类别进行 Huffman 编码标记，下面是哈夫曼编码与解码的过程。

3.2.1 哈夫曼编码

在哈夫曼编码过程中，我们首先需要根据像素值标记类别的统计情况，生成变长编码（即多位 0/1 编码）来表示像素值的标记类别。变长编码根据标记类别中像素的个数来确定编码的长度，使得像素值出现次数越多的类别，其编码长度越短。我们根据像素值标记类别的统计情况确定了映射关系 Code，然后根据该映射关系将像素值标记类别映射为二进制数组 Code_Bin。具体来说，根据编码规则 00,01,100,101,1100,1101,1110,11110,11111 来表示标记类别，然后将这些编码用二进制数组表示。

```
function [Code,Code_Bin] = Huffman_Code(num_Map_origin_I)
    % 函数说明: 用变长编码(多位0/1编码)表示像素值的标记类别
    % 输入: num_Map_origin_I (像素值标记类别的统计情况)
    % 输出: Code (映射关系) ,Code_Bin (Code的二进制表示)
    % 备注: 用{00,01,100,101,1100,1101,1110,11110,11111}来表示9种标记类别
    % 规则: 标记类别中像素的个数越多, 则用来表示其类别的编码长度越短
    % {00,01,100,101,1100,1101,1110,11110,11111}→{0,1,4,5,12,13,14,30,31}
    % 求其映射编码关系
    % Code = [-1,0;-1,1;-1,4;-1,5;-1,12;-1,13;-1,14;-1,30;-1,31];
    % 此处的-1仅有一个标记作用, 也可以修改为其他第二列未出现的值
    % 最终都会被替换成映射关系
```

```
    Code = [-2,0;-2,1;-2,4;-2,5;-2,12;-2,13;-2,14;-2,30;-2,31];
    for i=1:9
        drder=1;
        for j=1:9
            if num_Map_origin_I(i,2) < num_Map_origin_I(j,2)
                drder = drder + 1; %排序寻找最小值
            end
        end
        while Code(drder) ~= -2 %防止两种标记类别中像素的个数相等
            drder = drder + 1;
        end
        Code(drder,1) = num_Map_origin_I(i,1);
           %第一列从小到大排列了出现的标记顺序
    end
    % 将Map映射关系用二进制比特流表示
    Code_Bin = zeros();
    t = 0; %计数
    for i=0:8
        for j=1:9
            if Code(j,1) == i
                value = Code(j,2);
            end
        end
        if value == 0
            Code_Bin(t+1) = 0;
            Code_Bin(t+2) = 0;
            t = t+2;
        elseif value == 1
            Code_Bin(t+1) = 0;
            Code_Bin(t+2) = 1;
            t = t+2;
        else
            add = ceil(log2(value+1)); %表示标记编码的长度
            Code_Bin(t+1:t+add) = dec2bin(value)-'0';
               %将value转换成二进制数组
            t = t + add;
        end
    end
end
```

3.2.2 哈夫曼解码

在哈夫曼解码时，需要从二进制比特流中解码出下一个 Huffman 编码转换成的整数值。并根据输入的二进制序列 Binary 和上一个映射结束的位置 last_end，逐位判断并解析出下一个 Huffman 编码转换成的整数值，最后返回该值和本次解析结束的位置。

```
function [value,this_end] = Huffman_DeCode(Binary,last_end)
    % 求二进制比特流Binary中下一个Huffman编码转换成的整数值
    % 输入: Binary (二进制映射序列) ,last_end (上一个映射结束的位置)
    %
        输出: value (十进制整数值) →{0,1,4,5,12,13,14,30,31},end (本次结束的位置)
    len = length(Binary);
    i = last_end+1;
    t = 0; %计数
    if i <= len
        if i+1<=len && Binary(i)==0 %→0
            t = t + 1;
            if Binary(i+1) == 0 %→00表示0
                t = t + 1;
                value = 0;
            elseif Binary(i+1) == 1 %→01表示1
                t = t + 1;
                value = 1;
            end
        else  % Binary(i)==1
            if i+2<=len && Binary(i+1)==0 %→10
                t = t + 2;
                if Binary(i+2) == 0  %→100表示4
                    t = t + 1;
                    value = 4;
                elseif Binary(i+2) == 1 %→101表示5
                    t = t + 1;
                    value = 5;
                end
            else % Binary(i+1)==1
                if i+3<=len && Binary(i+2)==0  %→110
                    t = t + 3;
                    if Binary(i+3) == 0  %→1100表示12
                        t = t + 1;
                        value = 12;
                    elseif Binary(i+3) == 1  %→1101表示13
                        t = t + 1;
                        value = 13;
```

```
                    end
                else % Binary(i+2)==1
                    if i+3 <= len
                        t = t + 3;
                        if Binary(i+3) == 0  %→1110表示14
                            t = t + 1;
                            value = 14;
                        elseif i+4<=len && Binary(i+3)==1  %→1111
                            t = t + 1;
                            if Binary(i+4) == 0  %→11110表示30
                                t = t + 1;
                                value = 30;
                            elseif Binary(i+4) == 1  %→11111表示31
                                t = t + 1;
                                value = 31;
                            end
                        else
                            t = 0;
                            value = -1;
                               %表示辅助信息长度不够，无法恢复下一个Huffman编码
                        end
                    else
                        t = 0;
                        value = -1;
                           %表示辅助信息长度不够，无法恢复下一个Huffman编码
                    end
                end
            end
        end
    else
        t = 0;
        value = -1; %表示辅助信息长度不够，无法恢复下一个Huffman编码
    end
    this_end = last_end + t;
end
```

3.3 图像加密

在对原始明文图像进行像素值预测和自适应哈夫曼编码位图后，采用流密码对原始明文图像 I 进行加密。

我们实现了一个对图像进行异或加密的函数。该函数接受原始图像 origin_I 和图像加密密钥 Image_key 作为输入，并输出加密后的图像 encrypt_I。在加密过程中，首先获取原始图像的大小，并初

始化一个与原始图像大小相同的容器来存储加密后的图像。然后，根据密钥生成随机数种子，并使用该种子生成与原始图像大小相同的随机矩阵。接下来，通过遍历原始图像中的每个像素，将其与随机矩阵中对应位置的像素进行 bit 级异或操作，得到加密后的图像。

```
function [encrypt_I] = Encrypt_Image(origin_I,Image_key)
    % 函数说明：对图像origin_I进行bit级异或加密
    % 输入：origin_I（原始图像）,Image_key（图像加密密钥）
    % 输出：encrypt_I（加密图像）
    [row,col] = size(origin_I); %计算origin_I的行列值
    encrypt_I = origin_I;  %构建存储加密图像的容器
    % 根据密钥生成与origin_I大小相同的随机矩阵
    rand('seed',Image_key); %设置种子
    E = round(rand(row,col)*255); %随机生成row*col矩阵
    % 根据E对图像origin_I进行bit级加密
    for i=1:row
        for j=1:col
            encrypt_I(i,j) = bitxor(origin_I(i,j),E(i,j));
        end
    end
end
```

除了对图像进行加密以外，我们还实现了一个对原始秘密信息进行异或加密的函数。该函数接受两个输入参数：原始秘密信息 D 和数据加密密钥 Data_key，并返回加密后的秘密信息 Encrypt_D。在实现过程中，首先通过 length(D) 计算了原始秘密信息的长度，并将其存储在变量 num_D 中以确定循环次数。接着，将原始秘密信息复制到 Encrypt_D 中，以防止修改原始数据。然后，使用给定的密钥 Data_key 设置了随机数生成器的种子，以确保生成的随机序列是可重复的。随机生成了一个与原始秘密信息长度相同的随机 0/1 序列 E，并通过循环遍历原始秘密信息的每个元素，并与随机序列 E 中对应位置的元素进行 bit 级异或操作，得到加密后的秘密信息 Encrypt_D。

```
function [Encrypt_D] = Encrypt_Data(D,Data_key)
    % 函数说明：对原始秘密信息D进行bit级异或加密
    % 输入：D（原始秘密信息）,Data_key（数据加密密钥）
    % 输出：Encrypt_D（加密的秘密信息）
    num_D = length(D); %求嵌入数据D的长度
    Encrypt_D = D;  %构建存储加密秘密信息的容器
    % 根据密钥生成与D长度相同的随机0/1序列
    rand('seed',Data_key); %设置种子
    E = round(rand(1,num_D)*1); %随机生成长度为num_D的0/1序列
    % 根据E对原始秘密信息D进行异或加密
    for i=1:num_D
        Encrypt_D(i) = bitxor(D(i),E(i));
    end
end
```

3.4 位图嵌入

下面是对原图像的加密和数据嵌入的全过程，其中包含了位图嵌入的部分：

```
function [encrypt_I,stego_I,emD] =
    Encrypt_Embed(origin_I,D,Image_key,Data_key,ref_x,ref_y)
    % 函数说明：将原始图像origin_I加密并嵌入数据
    % 输入：origin_I（原始图像），D（要嵌入的数据），
    % Image_key,Data_key（密钥），ref_x,ref_y（参考像素的行列数）
    %
        输出：encrypt_I（加密图像），stego_I（加密标记图像），emD（嵌入的数据）

    % 计算origin_I的预测值
    [origin_PV_I] = Predictor_Value(origin_I,ref_x,ref_y);
    % 对每个像素值进行标记（即原始图像的位置图）
    [Map_origin_I] = Category_Mark(origin_PV_I,origin_I,ref_x,ref_y);
    % 将像素值的标记类别进行Huffman编码标记
    hist_Map_origin_I = tabulate(Map_origin_I(:));
       %统计每个标记类别的像素值个数
    num_Map_origin_I = zeros(9,2);
    for i=1:9  % 9种类别的标记
        num_Map_origin_I(i,1) = i-1;   % num_Map_origin_I=[0 0;1 0;2 0;3
           0;4 0;5 0;6 0;7 0;8 0]
    end
    [m,~] = size(hist_Map_origin_I);
    for i=1:9
        for j=2:m %hist_Map_origin_I第一行统计的是参考像素的个数
            if num_Map_origin_I(i,1) == hist_Map_origin_I(j,1)
                num_Map_origin_I(i,2) = hist_Map_origin_I(j,2);
                   %去掉参考像素信息，只统计标记类别信息
            end
        end
    end
    [Code,Code_Bin] = Huffman_Code(num_Map_origin_I);
       %计算标记的映射关系及其二进制序列表示
    % 将位置图Map_origin_I转换成二进制数组
    [Map_Bin] = Map_Binary(Map_origin_I,Code);
    % 计算存储Map_Binary长度需要的信息长度
    [row,col]=size(origin_I);
    max = ceil(log2(row)) + ceil(log2(col)) + 2;
       %用这么长的二进制表示Map_Binary的长度
       ceil()与floor相对，表示向上取整
    length_Map = length(Map_Bin);
```

```
    len_Bin = dec2bin(length_Map)-'0'; %将length_Map转换成二进制数组
    if length(len_Bin) < max
        len = length(len_Bin);
        B = len_Bin;
        len_Bin = zeros(1,max);
        for i=1:len
            len_Bin(max-len+i) = B(i); %存储Map_Bin的长度信息
        end
    end
    % 统计恢复时需要的辅助信息 (Code_Bin, len_Bin, Map_Bin)
    Side_Information = [Code_Bin,len_Bin,Map_Bin];
    % 对原始图像origin_I进行加密
    [encrypt_I] = Encrypt_Image(origin_I,Image_key);
    % 在Encrypt_I中嵌入信息
    [stego_I,emD] =
       Embed_Data(encrypt_I,Map_origin_I,Side_Information,D,Data_key,ref_x,ref_y);
end
```

3.5 信息嵌入

接下来，需要将秘密信息和辅助信息嵌入到加密图像中，其中嵌入的位置由位置图 Map_origin_I 决定。秘密信息会在位置图允许的情况下优先嵌入，然后是辅助信息和参考像素。

首先对原始秘密信息 D 进行加密，得到 Encrypt_D。然后需要在前 ref_y 列、前 ref_x 行的参考像素中存储辅助信息。接下来，根据位置图，将秘密信息和辅助信息嵌入到图像中。具体嵌入过程如下：

1. 如果像素可以嵌入 1 到 8 位的信息，首先检查秘密信息是否已经全部嵌入完成，然后检查辅助信息和参考像素是否还有剩余，根据情况进行嵌入。
2. 根据像素可以嵌入的位数，选择对应数量的位数进行嵌入。例如，如果可以嵌入 1 位信息，直接替换最高有效位（MSB）；如果可以嵌入 2 位信息，则替换最高的两位。
3. 如果辅助信息或参考像素的位数不够，则从嵌入的秘密信息中获取。若秘密信息位数不够，则将其余部分作为参考像素信息嵌入。
4. 重复上述过程，直到所有信息都嵌入完成。

详细代码如下：

```
function [stego_I,emD] =
   Embed_Data(encrypt_I,Map_origin_I,Side_Information,D,Data_key,ref_x,ref_y)
    % 函数说明: 根据位置图将辅助信息和秘密信息嵌入到加密图像中
    %
        输入: encrypt_I (加密图像) ,Map_origin_I (位置图) ,Side_Information (辅助信息
    % 输出: stego_I (加密标记图像) ,emD (嵌入的数据)
```

```
    stego_I = encrypt_I;
    [row,col] = size(encrypt_I); %统计encrypt_I的行列数
    % 对原始秘密信息D进行加密
    [Encrypt_D] = Encrypt_Data(D,Data_key);
    %
        将前ref_y列、前ref_x行的参考像素记录下来，放在秘密信息之前嵌入图像中
    Refer_Value = zeros(); %记录参考像素的数组
    t = 0; %计数
    for i=1:row
        for j=1:ref_y  %前ref_y列
            value = encrypt_I(i,j);
            [bin2_8] = Decimalism_Binary(value);
               %将十进制整数转换成8位二进制数组
            Refer_Value(t+1:t+8) = bin2_8; %因为t=0，所以从t+1开始
            t = t + 8;
        end
    end
    for i=1:ref_x  %前ref_x行
        for j=ref_y+1:col
            value = encrypt_I(i,j);
            [bin2_8] = Decimalism_Binary(value);
               %将十进制整数转换成8位二进制数组
            Refer_Value(t+1:t+8) = bin2_8;
            t = t + 8;
        end
    end
    %% 辅助量
    num_D = length(D); %求秘密信息D的长度
    num_emD = 0; %计数，统计嵌入秘密信息的个数
    num_S = length(Side_Information); %求辅助信息Side_Information的长度
    num_side = 0;%计数，统计嵌入辅助信息的个数
    num_RV = length(Refer_Value); %参考像素二进制序列信息的长度
    num_re = 0; %计数，统计嵌入参考像素二进制序列信息的长度
    %% 先在前ref_y列、前ref_x行的参考像素中存储辅助信息
    for i=1:row
        for j=1:ref_y  %前ref_y列
            bin2_8 = Side_Information(num_side+1:num_side+8);
            [value] = Binary_Decimalism(bin2_8);
               %将8位二进制数组转换成十进制整数
            stego_I(i,j) = value;
            num_side = num_side + 8;
        end
```

```
    end
    for i=1:ref_x  %前ref_x行
        for j=ref_y+1:col
            bin2_8 = Side_Information(num_side+1:num_side+8);
            [value] = Binary_Decimalism(bin2_8);
               %将8位二进制数组转换成十进制整数
            stego_I(i,j) = value;
            num_side = num_side + 8;
        end
    end
    %% 再在其余位置嵌入辅助信息、参考像素和秘密数据
    for i=ref_x+1:row
        for j=ref_y+1:col
            if num_emD >= num_D %秘密数据已嵌完
                break;
            end
            %------表示这个像素点可以嵌入 1 bit信息------%
            if Map_origin_I(i,j) == 0
               %Map=0表示原始像素值的第1MSB与其预测值相反
                if num_side < num_S %辅助信息没有嵌完
                    num_side = num_side + 1;
                    stego_I(i,j) = mod(stego_I(i,j),2^7) +
                       Side_Information(num_side)*(2^7); %替换1位MSB
                else
                    if num_re < num_RV %参考像素二进制序列信息没有嵌完
                        num_re = num_re + 1;
                        stego_I(i,j) = mod(stego_I(i,j),2^7) +
                           Refer_Value(num_re)*(2^7); %替换1位MSB
                    else %最后嵌入秘密信息
                        num_emD = num_emD + 1;
                        stego_I(i,j) = mod(stego_I(i,j),2^7) +
                           Encrypt_D(num_emD)*(2^7); %替换1位MSB
                    end
                end
            %------表示这个像素点可以嵌入 2 bit信息------%
            elseif Map_origin_I(i,j) == 1
               %Map=1表示原始像素值的第2MSB与其预测值相反
                if num_side < num_S %辅助信息没有嵌完
                    if num_side+2 <= num_S %2位MSB都用来嵌入辅助信息
                        num_side = num_side + 2;
                        stego_I(i,j) = mod(stego_I(i,j),2^6) +
                           Side_Information(num_side-1)*(2^7) +
```

```
                    Side_Information(num_side)*(2^6); %替换2位MSB
78                elseif
79                    num_side = num_side + 1; %1bit辅助信息
80                    num_re = num_re + 1; %1bit参考像素二进制序列信息
81                    stego_I(i,j) = mod(stego_I(i,j),2^6) +
                          Side_Information(num_side)*(2^7) +
                          Refer_Value(num_re)*(2^6); %替换2位MSB
82                end
83            else
84                if num_re < num_RV %参考像素二进制序列信息没有嵌完
85                    if num_re+2 <= num_RV
                          %2位MSB都用来嵌入参考像素二进制序列信息
86                        num_re = num_re + 2;
87                        stego_I(i,j) = mod(stego_I(i,j),2^6) +
                              Refer_Value(num_re-1)*(2^7) +
                              Refer_Value(num_re)*(2^6); %替换2位MSB
88                    else
89                        num_re = num_re + 1;
                              %1bit参考像素二进制序列信息
90                        num_emD = num_emD + 1; %1bit秘密信息
91                        stego_I(i,j) = mod(stego_I(i,j),2^6) +
                              Refer_Value(num_re)*(2^7) +
                              Encrypt_D(num_emD)*(2^6); %替换2位MSB
92                    end
93                elseif
94                    if num_emD+2 <= num_D
95                        num_emD = num_emD + 2; %2bit秘密信息
96                        stego_I(i,j) = mod(stego_I(i,j),2^6) +
                              Encrypt_D(num_emD-1)*(2^7) +
                              Encrypt_D(num_emD)*(2^6); %替换2位MSB
97                    else
98                        num_emD = num_emD + 1; %1bit秘密信息
99                        stego_I(i,j) = mod(stego_I(i,j),2^7) +
                              Encrypt_D(num_emD)*(2^7); %替换1位MSB
100                   end
101               end
102           end
103       %------表示这个像素点可以嵌入 3 bit信息------%
104       elseif Map_origin_I(i,j) == 2
              %Map=2表示原始像素值的第3MSB与其预测值相反
105           bin2_8 = zeros(1,8);
                  %用来记录要嵌入的信息，少于8位的低位(LSB)默认为0
```

```
106                 if num_side < num_S %辅助信息没有嵌完
107                     if num_side+3 <= num_S %3位MSB都用来嵌入辅助信息
108                         bin2_8(1:3) =
                               Side_Information(num_side+1:num_side+3);
109                         num_side = num_side + 3;
110                         [value] = Binary_Decimalism(bin2_8);
111                         stego_I(i,j) = mod(stego_I(i,j),2^5) + value;
                               %替换3位MSB
112                     else
113                         t = num_S - num_side; %剩余辅助信息个数
114                         bin2_8(1:t) =
                               Side_Information(num_side+1:num_S);
                               %tbit辅助信息
115                         num_side = num_side + t;
116                         bin2_8(t+1:3) =
                               Refer_Value(num_re+1:num_re+3-t);
                               %(3-t)bit参考像素二进制序列信息
117                         num_re = num_re + 3-t;
118                         [value] = Binary_Decimalism(bin2_8);
119                         stego_I(i,j) = mod(stego_I(i,j),2^5) + value;
                               %替换3位MSB
120                     end
121                 else
122                     if num_re < num_RV  %参考像素二进制序列信息没有嵌完
123                         if num_re+3 <= num_RV
                               %3位MSB都用来嵌入参考像素二进制序列信息
124                             bin2_8(1:3) = Refer_Value(num_re+1:num_re+3);
125                             num_re = num_re + 3;
126                             [value] = Binary_Decimalism(bin2_8);
127                             stego_I(i,j) = mod(stego_I(i,j),2^5) +
                                   value; %替换3位MSB
128                         else
129                             t = num_RV - num_re;
                                   %剩余参考像素二进制序列信息个数
130                             bin2_8(1:t) = Refer_Value(num_re+1:num_RV);
                                   %tbit参考像素二进制序列信息
131                             num_re = num_re + t;
132                             bin2_8(t+1:3) =
                                   Encrypt_D(num_emD+1:num_emD+3-t);
                                   %(3-t)bit秘密信息
133                             num_emD = num_emD + 3-t;
134                             [value] = Binary_Decimalism(bin2_8);
```

```
135                         stego_I(i,j) = mod(stego_I(i,j),2^5) +
                                value; %替换3位MSB
136                     end
137                 else
138                     if num_emD+3 <= num_D
139                         bin2_8(1:3) =
                                Encrypt_D(num_emD+1:num_emD+3);
                                %3bit秘密信息
140                         num_emD = num_emD + 3;
141                         [value] = Binary_Decimalism(bin2_8);
142                         stego_I(i,j) = mod(stego_I(i,j),2^5) +
                                value; %替换3位MSB
143                     else
144                         t = num_D - num_emD; %剩余秘密信息个数
145                         bin2_8(1:t) =
                                Encrypt_D(num_emD+1:num_emD+t);
                                %tbit秘密信息
146                         num_emD = num_emD + t;
147                         [value] = Binary_Decimalism(bin2_8);
148                         stego_I(i,j) = mod(stego_I(i,j),2^(8-t)) +
                                value; %替换t位MSB
149                     end
150                 end
151             end
152         %------表示这个像素点可以嵌入 4 bit信息------%
153         elseif Map_origin_I(i,j) == 3
               %Map=3表示原始像素值的第4MSB与其预测值相反
154             bin2_8 = zeros(1,8);
                   %用来记录要嵌入的信息，少于8位的低位(LSB)默认为0
155             if num_side < num_S %辅助信息没有嵌完
156                 if num_side+4 <= num_S %4位MSB都用来嵌入辅助信息
157                     bin2_8(1:4) =
                            Side_Information(num_side+1:num_side+4);
158                     num_side = num_side + 4;
159                     [value] = Binary_Decimalism(bin2_8);
160                     stego_I(i,j) = mod(stego_I(i,j),2^4) + value;
                            %替换4位MSB
161                 else
162                     t = num_S - num_side; %剩余辅助信息个数
163                     bin2_8(1:t) =
                            Side_Information(num_side+1:num_S);
                            %tbit辅助信息
```

```
                    num_side = num_side + t;
                    bin2_8(t+1:4) =
                        Refer_Value(num_re+1:num_re+4-t);
                        %(4-t)bit参考像素二进制序列信息
                    num_re = num_re + 4-t;
                    [value] = Binary_Decimalism(bin2_8);
                    stego_I(i,j) = mod(stego_I(i,j),2^4) + value;
                        %替换4位MSB
                end
            else
                if num_re < num_RV  %参考像素二进制序列信息没有嵌完
                    if num_re+4 <= num_RV
                        %4位MSB都用来嵌入参考像素二进制序列信息
                        bin2_8(1:4) = Refer_Value(num_re+1:num_re+4);
                        num_re = num_re + 4;
                        [value] = Binary_Decimalism(bin2_8);
                        stego_I(i,j) = mod(stego_I(i,j),2^4) +
                            value; %替换4位MSB
                    else
                        t = num_RV - num_re;
                            %剩余参考像素二进制序列信息个数
                        bin2_8(1:t) = Refer_Value(num_re+1:num_RV);
                            %tbit参考像素二进制序列信息
                        num_re = num_re + t;
                        bin2_8(t+1:4) =
                            Encrypt_D(num_emD+1:num_emD+4-t);
                            %(4-t)bit秘密信息
                        num_emD = num_emD + 4-t;
                        [value] = Binary_Decimalism(bin2_8);
                        stego_I(i,j) = mod(stego_I(i,j),2^4) +
                            value; %替换4位MSB
                    end
                else
                    if num_emD+4 <= num_D
                        bin2_8(1:4) =
                            Encrypt_D(num_emD+1:num_emD+4);
                            %4bit秘密信息
                        num_emD = num_emD + 4;
                        [value] = Binary_Decimalism(bin2_8);
                        stego_I(i,j) = mod(stego_I(i,j),2^4) +
                            value; %替换4位MSB
                    else
```

```
193                         t = num_D - num_emD; %剩余秘密信息个数
194                         bin2_8(1:t) =
                                Encrypt_D(num_emD+1:num_emD+t);
                                %tbit秘密信息
195                         num_emD = num_emD + t;
196                         [value] = Binary_Decimalism(bin2_8);
197                         stego_I(i,j) = mod(stego_I(i,j),2^(8-t)) +
                                value; %替换t位MSB
198                     end
199                 end
200             end
201         %------表示这个像素点可以嵌入 5 bit信息------%
202         elseif Map_origin_I(i,j) == 4
               %Map=4表示原始像素值的第5MSB与其预测值相反
203             bin2_8 = zeros(1,8);
                   %用来记录要嵌入的信息，少于8位的低位(LSB)默认为0
204             if num_side < num_S %辅助信息没有嵌完
205                 if num_side+5 <= num_S %5位MSB都用来嵌入辅助信息
206                     bin2_8(1:5) =
                            Side_Information(num_side+1:num_side+5);
207                     num_side = num_side + 5;
208                     [value] = Binary_Decimalism(bin2_8);
209                     stego_I(i,j) = mod(stego_I(i,j),2^3) + value;
                            %替换5位MSB
210                 else
211                     t = num_S - num_side; %剩余辅助信息个数
212                     bin2_8(1:t) =
                            Side_Information(num_side+1:num_S);
                            %tbit辅助信息
213                     num_side = num_side + t;
214                     bin2_8(t+1:5) =
                            Refer_Value(num_re+1:num_re+5-t);
                            %(5-t)bit参考像素二进制序列信息
215                     num_re = num_re + 5-t;
216                     [value] = Binary_Decimalism(bin2_8);
217                     stego_I(i,j) = mod(stego_I(i,j),2^3) + value;
                            %替换5位MSB
218                 end
219             else
220                 if num_re < num_RV  %参考像素二进制序列信息没有嵌完
221                     if num_re+5 <= num_RV
                            %5位MSB都用来嵌入参考像素二进制序列信息
```

```
222                             bin2_8(1:5) = Refer_Value(num_re+1:num_re+5);
223                             num_re = num_re + 5;
224                             [value] = Binary_Decimalism(bin2_8);
225                             stego_I(i,j) = mod(stego_I(i,j),2^3) +
                                    value; %替换5位MSB
226                         else
227                             t = num_RV - num_re;
                                    %剩余参考像素二进制序列信息个数
228                             bin2_8(1:t) = Refer_Value(num_re+1:num_RV);
                                    %tbit参考像素二进制序列信息
229                             num_re = num_re + t;
230                             bin2_8(t+1:5) =
                                    Encrypt_D(num_emD+1:num_emD+5-t);
                                    %(5-t)bit秘密信息
231                             num_emD = num_emD + 5-t;
232                             [value] = Binary_Decimalism(bin2_8);
233                             stego_I(i,j) = mod(stego_I(i,j),2^3) +
                                    value; %替换5位MSB
234                         end
235                     else
236                         if num_emD+5 <= num_D
237                             bin2_8(1:5) =
                                    Encrypt_D(num_emD+1:num_emD+5);
                                    %5bit秘密信息
238                             num_emD = num_emD + 5;
239                             [value] = Binary_Decimalism(bin2_8);
240                             stego_I(i,j) = mod(stego_I(i,j),2^3) +
                                    value; %替换5位MSB
241                         else
242                             t = num_D - num_emD; %剩余秘密信息个数
243                             bin2_8(1:t) =
                                    Encrypt_D(num_emD+1:num_emD+t);
                                    %tbit秘密信息
244                             num_emD = num_emD + t;
245                             [value] = Binary_Decimalism(bin2_8);
246                             stego_I(i,j) = mod(stego_I(i,j),2^(8-t)) +
                                    value; %替换t位MSB
247                         end
248                     end
249                 end
250             %------表示这个像素点可以嵌入 6 bit信息------%
251             elseif Map_origin_I(i,j) == 5
```

```
                %Map=5表示原始像素值的第6MSB与其预测值相反
252             bin2_8 = zeros(1,8);
                   %用来记录要嵌入的信息，少于8位的低位(LSB)默认为0
253             if num_side < num_S %辅助信息没有嵌完
254                 if num_side+6 <= num_S %6位MSB都用来嵌入辅助信息
255                     bin2_8(1:6) =
                           Side_Information(num_side+1:num_side+6);
256                     num_side = num_side + 6;
257                     [value] = Binary_Decimalism(bin2_8);
258                     stego_I(i,j) = mod(stego_I(i,j),2^2) + value;
                           %替换6位MSB
259                 else
260                     t = num_S - num_side; %剩余辅助信息个数
261                     bin2_8(1:t) =
                           Side_Information(num_side+1:num_S);
                           %tbit辅助信息
262                     num_side = num_side + t;
263                     bin2_8(t+1:6) =
                           Refer_Value(num_re+1:num_re+6-t);
                           %(6-t)bit参考像素二进制序列信息
264                     num_re = num_re + 6-t;
265                     [value] = Binary_Decimalism(bin2_8);
266                     stego_I(i,j) = mod(stego_I(i,j),2^2) + value;
                           %替换6位MSB
267                 end
268             else
269                 if num_re < num_RV  %参考像素二进制序列信息没有嵌完
270                     if num_re+6 <= num_RV
                           %3位MSB都用来嵌入参考像素二进制序列信息
271                         bin2_8(1:6) = Refer_Value(num_re+1:num_re+6);
272                         num_re = num_re + 6;
273                         [value] = Binary_Decimalism(bin2_8);
274                         stego_I(i,j) = mod(stego_I(i,j),2^2) +
                               value; %替换6位MSB
275                     else
276                         t = num_RV - num_re;
                               %剩余参考像素二进制序列信息个数
277                         bin2_8(1:t) = Refer_Value(num_re+1:num_RV);
                               %tbit参考像素二进制序列信息
278                         num_re = num_re + t;
279                         bin2_8(t+1:6) =
                               Encrypt_D(num_emD+1:num_emD+6-t);
```

```
                        %(6-t)bit秘密信息
280                     num_emD = num_emD + 6-t;
281                     [value] = Binary_Decimalism(bin2_8);
282                     stego_I(i,j) = mod(stego_I(i,j),2^2) +
                            value; %替换6位MSB
283                 end
284             else
285                 if num_emD+6 <= num_D
286                     bin2_8(1:6) =
                            Encrypt_D(num_emD+1:num_emD+6);
                            %6bit秘密信息
287                     num_emD = num_emD + 6;
288                     [value] = Binary_Decimalism(bin2_8);
289                     stego_I(i,j) = mod(stego_I(i,j),2^2) +
                            value; %替换6位MSB
290                 else
291                     t = num_D - num_emD; %剩余秘密信息个数
292                     bin2_8(1:t) =
                            Encrypt_D(num_emD+1:num_emD+t);
                            %tbit秘密信息
293                     num_emD = num_emD + t;
294                     [value] = Binary_Decimalism(bin2_8);
295                     stego_I(i,j) = mod(stego_I(i,j),2^(8-t)) +
                            value; %替换t位MSB
296                 end
297             end
298         end
299     %------表示这个像素点可以嵌入 7 bit信息------%
300     elseif Map_origin_I(i,j) == 6
            %Map=6表示原始像素值的第7MSB与其预测值相反
301         bin2_8 = zeros(1,8);
                %用来记录要嵌入的信息，少于8位的低位(LSB)默认为0
302         if num_side < num_S %辅助信息没有嵌完
303             if num_side+7 <= num_S %7位MSB都用来嵌入辅助信息
304                 bin2_8(1:7) =
                        Side_Information(num_side+1:num_side+7);
305                 num_side = num_side + 7;
306                 [value] = Binary_Decimalism(bin2_8);
307                 stego_I(i,j) = mod(stego_I(i,j),2^1) + value;
                        %替换7位MSB
308             else
309                 t = num_S - num_side; %剩余辅助信息个数
```

```
310                         bin2_8(1:t) =
                                Side_Information(num_side+1:num_S);
                                %tbit辅助信息
311                         num_side = num_side + t;
312                         bin2_8(t+1:7) =
                                Refer_Value(num_re+1:num_re+7-t);
                                %(7-t)bit参考像素二进制序列信息
313                         num_re = num_re + 7-t;
314                         [value] = Binary_Decimalism(bin2_8);
315                         stego_I(i,j) = mod(stego_I(i,j),2^1) + value;
                                %替换7位MSB
316                     end
317                 else
318                     if num_re < num_RV  %参考像素二进制序列信息没有嵌完
319                         if num_re+7 <= num_RV
                                %7位MSB都用来嵌入参考像素二进制序列信息
320                             bin2_8(1:7) = Refer_Value(num_re+1:num_re+7);
321                             num_re = num_re + 7;
322                             [value] = Binary_Decimalism(bin2_8);
323                             stego_I(i,j) = mod(stego_I(i,j),2^1) +
                                    value; %替换7位MSB
324                         else
325                             t = num_RV - num_re;
                                    %剩余参考像素二进制序列信息个数
326                             bin2_8(1:t) = Refer_Value(num_re+1:num_RV);
                                    %tbit参考像素二进制序列信息
327                             num_re = num_re + t;
328                             bin2_8(t+1:7) =
                                    Encrypt_D(num_emD+1:num_emD+7-t);
                                    %(7-t)bit秘密信息
329                             num_emD = num_emD + 7-t;
330                             [value] = Binary_Decimalism(bin2_8);
331                             stego_I(i,j) = mod(stego_I(i,j),2^1) +
                                    value; %替换7位MSB
332                         end
333                     else
334                         if num_emD+7 <= num_D
335                             bin2_8(1:7) =
                                    Encrypt_D(num_emD+1:num_emD+7);
                                    %7bit秘密信息
336                             num_emD = num_emD + 7;
337                             [value] = Binary_Decimalism(bin2_8);
```

```
338                         stego_I(i,j) = mod(stego_I(i,j),2^1) +
                                value; %替换7位MSB
339                     else
340                         t = num_D - num_emD; %剩余秘密信息个数
341                         bin2_8(1:t) =
                                Encrypt_D(num_emD+1:num_emD+t);
                                %tbit秘密信息
342                         num_emD = num_emD + t;
343                         [value] = Binary_Decimalism(bin2_8);
344                         stego_I(i,j) = mod(stego_I(i,j),2^(8-t)) +
                                value; %替换t位MSB
345                     end
346                 end
347             end
348         %------表示这个像素点可以嵌入 8 bit信息------%
349         elseif Map_origin_I(i,j) == 7 || Map_origin_I(i,j) == 8
               %Map=7表示原始像素值的第8MSB(LSB)与其预测值相反
350             bin2_8 = zeros(1,8);
                   %用来记录要嵌入的信息，少于8位的低位(LSB)默认为0
351             if num_side < num_S %辅助信息没有嵌完
352                 if num_side+8 <= num_S %8位MSB都用来嵌入辅助信息
353                     bin2_8(1:8) =
                            Side_Information(num_side+1:num_side+8);
354                     num_side = num_side + 8;
355                     [value] = Binary_Decimalism(bin2_8);
356                     stego_I(i,j) = value; %替换8位MSB
357                 else
358                     t = num_S - num_side; %剩余辅助信息个数
359                     bin2_8(1:t) =
                            Side_Information(num_side+1:num_S);
                            %tbit辅助信息
360                     num_side = num_side + t;
361                     bin2_8(t+1:8) =
                            Refer_Value(num_re+1:num_re+8-t);
                            %(8-t)bit参考像素二进制序列信息
362                     num_re = num_re + 8-t;
363                     [value] = Binary_Decimalism(bin2_8);
364                     stego_I(i,j) = value; %替换8位MSB
365                 end
366             else
367                 if num_re < num_RV  %参考像素二进制序列信息没有嵌完
368                     if num_re+8 <= num_RV
```

```
                                %8位MSB都用来嵌入参考像素二进制序列信息
369                             bin2_8(1:8) = Refer_Value(num_re+1:num_re+8);
370                             num_re = num_re + 8;
371                             [value] = Binary_Decimalism(bin2_8);
372                             stego_I(i,j) = value; %替换8位MSB
373                         else
374                             t = num_RV - num_re;
                                    %剩余参考像素二进制序列信息个数
375                             bin2_8(1:t) = Refer_Value(num_re+1:num_RV);
                                    %tbit参考像素二进制序列信息
376                             num_re = num_re + t;
377                             bin2_8(t+1:8) =
                                    Encrypt_D(num_emD+1:num_emD+8-t);
                                    %(8-t)bit秘密信息
378                             num_emD = num_emD + 8-t;
379                             [value] = Binary_Decimalism(bin2_8);
380                             stego_I(i,j) = value; %替换8位MSB
381                         end
382                     else
383                         if num_emD+8 <= num_D
384                             bin2_8(1:8) =
                                    Encrypt_D(num_emD+1:num_emD+8);
                                    %8bit秘密信息
385                             num_emD = num_emD + 8;
386                             [value] = Binary_Decimalism(bin2_8);
387                             stego_I(i,j) = value; %替换8位MSB
388                         else
389                             t = num_D - num_emD; %剩余秘密信息个数
390                             bin2_8(1:t) =
                                    Encrypt_D(num_emD+1:num_emD+t);
                                    %tbit秘密信息
391                             num_emD = num_emD + t;
392                             [value] = Binary_Decimalism(bin2_8);
393                             stego_I(i,j) = mod(stego_I(i,j),2^(8-t)) +
                                    value; %替换t位MSB
394                         end
395                     end
396                 end
397             end
398         end
399     end
400     % 统计嵌入的秘密数据
```

```
401     emD = D(1:num_emD);
402 end
```

3.6 信息提取

在接受者提取秘密信息之前，需要先从载密图像中提取位图信息，并用3.5节**信息嵌入**同样的方法提取全部位图信息、参考像素和加密的秘密信息。

首先，创建一个与加密图像相同大小的位置图，并将前面的参考像素位置标记为-1。接下来提取辅助信息，包括映射规则、位置图二进制序列的长度信息以及参考像素信息。随后根据位置图和辅助信息，逐个像素提取隐藏的秘密信息。在提取秘密信息时，根据位置图的不同类别，决定每个像素可以提取多少位的信息，并将提取出的信息放置在相应的数据容器中。最后，返回提取得到的所有信息。

详细代码如下：

```
1  function [Side_Information,Refer_Value,Encrypt_exD,Map_I,sign] =
       Extract_Data(stego_I,num,ref_x,ref_y)
2      % 函数说明：在加密标记图像中提取信息
3      % 输入：stego_I（加密标记图像），num（秘密信息的长度），
4      % ref_x,ref_y（参考像素的行列数）
5      % 输出：Side_Information（辅助信息），Refer_Value（参考像素信息），
6      % Encrypt_exD（加密的秘密信息），Map_I（位置图），sign（判断标记）
7      [row,col]=size(stego_I); %统计stego_I的行列数
8      %% 构建存储位置图的矩阵
9      Map_I = zeros(row,col); %构建存储位置图的矩阵
10     for i=1:row
11         for j=1:ref_y
12             Map_I(i,j) = -1; %前面ref_y列为参考像素，不进行标记
13         end
14     end
15     for i=1:ref_x
16         for j=ref_y+1:col
17             Map_I(i,j) = -1; %前面ref_x行为参考像素，不进行标记
18         end
19     end
20     %% 先提取前ref_y列、前ref_x行中的辅助信息
21     Side_Information = zeros();
22     num_side = 0;%计数，统计提取辅助信息的个数
23     for i=1:row
24         for j=1:ref_y
25             value = stego_I(i,j);
26             [bin2_8] = Decimalism_Binary(value);
                   %将十进制整数转换成8位二进制数组
27             Side_Information(num_side+1:num_side+8) = bin2_8;
```

```
            num_side = num_side + 8;
        end
    end
    for i=1:ref_x
        for j=ref_y+1:col
            value = stego_I(i,j);
            [bin2_8] = Decimalism_Binary(value);
               %将十进制整数转换成8位二进制数组
            Side_Information(num_side+1:num_side+8) = bin2_8;
            num_side = num_side + 8;
        end
    end
    %% 提取代表映射规则的辅助信息
    Code_Bin = Side_Information(1:32); %前32位是映射规则信息
    Code = [0,-1;1,-1;2,-1;3,-1;4,-1;5,-1;6,-1;7,-1;8,-1];
    this_end = 0;
    for i=1:9 %将二进制序列映射转换成整数映射
        last_end = this_end;
        [code_value,this_end] = Huffman_DeCode(Code_Bin,last_end);
        Code(i,2) = code_value;
    end
    %% 提取位置图二进制序列的长度信息
    max = ceil(log2(row)) + ceil(log2(col)) + 2;
       %用这么长的二进制表示Map_I转化成二进制数列的长度
    len_Bin = Side_Information(33:32+max);
       %前33到32+max位是位置图二进制序列的长度信息
    num_Map = 0; %将二进制序列len_Bin转换成十进制数
    for i=1:max
        num_Map = num_Map + len_Bin(i)*(2^(max-i));
    end
    %% 辅助量
    num_S = 32 + max + num_Map; %辅助信息长度
    Refer_Value = zeros();
    num_RV = (ref_x*row+ref_y*col-ref_x*ref_y)*8;
       %参考像素二进制序列信息的长度
    num_re = 0; %计数，统计提取参考像素二进制序列信息的长度
    Encrypt_exD = zeros();
    num_D = num; %二进制秘密信息的长度
    num_exD = 0; %计数，统计嵌入秘密信息的个数
    %% 在前多行多列之外的位置提取信息
    this_end = 32 + max; %前面的辅助信息不是位置图
    sign = 1; %表示可以完全提取数据恢复图像
```

```
    for i=ref_x+1:row
        if sign == 0 %表示不能完全提取数据恢复图像
            break;
        end
        for j=ref_y+1:col
            if num_exD >= num_D %秘密数据已提取完毕
                break;
            end
            %------将当前十进制像素值转换成8位二进制数组------%
            value = stego_I(i,j);
            [bin2_8] = Decimalism_Binary(value);
            %--通过辅助信息计算当前像素点能提取多少bit的信息--%
            last_end = this_end;
            [map_value,this_end] =
               Huffman_DeCode(Side_Information,last_end);
            if map_value == -1
               %表示辅助信息长度不够，无法恢复下一个Huffman编码
                sign = 0;
                break;
            end
            for k=1:9
                if map_value == Code(k,2)
                    Map_I(i,j) = Code(k,1); %当前像素的位置图信息
                    break;
                end
            end
            %--------表示这个像素点可以提取 1 bit信息--------%
            if Map_I(i,j) == 0
               %Map=0表示原始像素值的第1MSB与其预测值相反
                if num_side < num_S %辅助信息没有提取完毕
                    num_side = num_side + 1;
                    Side_Information(num_side) = bin2_8(1);
                else
                    if num_re < num_RV
                       %参考像素二进制序列信息没有提取完毕
                        num_re = num_re + 1;
                        Refer_Value(num_re) = bin2_8(1);
                    else %最后提取秘密信息
                        num_exD = num_exD + 1;
                        Encrypt_exD(num_exD) = bin2_8(1);
                    end
                end
```

```
104             %--------表示这个像素点可以提取 2 bit信息--------%
105             elseif Map_I(i,j) == 1
                    %Map=1表示原始像素值的第2MSB与其预测值相反
106                 if num_side < num_S %辅助信息没有提取完毕
107                     if num_side+2 <= num_S %2位MSB都是辅助信息
108                         Side_Information(num_side+1:num_side+2) =
                                bin2_8(1:2);
109                         num_side = num_side + 2;
110                     else
111                         num_side = num_side + 1; %1bit辅助信息
112                         Side_Information(num_side) = bin2_8(1);
113                         num_re = num_re + 1; %1bit参考像素二进制序列信息
114                         Refer_Value(num_re) = bin2_8(2);
115                     end
116                 else
117                     if num_re < num_RV
                            %参考像素二进制序列信息没有提取完毕
118                         if num_re+2 <= num_RV
                                %2位MSB都是参考像素二进制序列信息
119                             Refer_Value(num_re+1:num_re+2) = bin2_8(1:2);
120                             num_re = num_re + 2;
121                         else
122                             num_re = num_re + 1;
                                    %1bit参考像素二进制序列信息
123                             Refer_Value(num_re) = bin2_8(1);
124                             num_exD = num_exD + 1; %1bit秘密信息
125                             Encrypt_exD(num_exD) = bin2_8(2);
126                         end
127                     else
128                         if num_exD+2 <= num_D
129                             Encrypt_exD(num_exD+1:num_exD+2) =
                                    bin2_8(1:2); %2bit秘密信息
130                             num_exD = num_exD + 2;
131                         else
132                             num_exD = num_exD + 1; %1bit秘密信息
133                             Encrypt_exD(num_exD) = bin2_8(1);
134                         end
135                     end
136                 end
137             %--------表示这个像素点可以提取 3 bit信息--------%
138             elseif Map_I(i,j) == 2
                    %Map=2表示原始像素值的第3MSB与其预测值相反
```

```
                if num_side < num_S %辅助信息没有提取完毕
                    if num_side+3 <= num_S %3位MSB都是辅助信息
                        Side_Information(num_side+1:num_side+3) =
                            bin2_8(1:3);
                        num_side = num_side + 3;
                    else
                        t = num_S - num_side; %剩余辅助信息个数
                        Side_Information(num_side+1:num_side+t) =
                            bin2_8(1:t); %tbit辅助信息
                        num_side = num_side + t;
                        Refer_Value(num_re+1:num_re+3-t) =
                            bin2_8(t+1:3); %(3-t)bit参考像素二进制序列信息
                        num_re = num_re + 3-t;
                    end
                else
                    if num_re < num_RV
                       %参考像素二进制序列信息没有提取完毕
                        if num_re+3 <= num_RV
                           %3位MSB都是参考像素二进制序列信息
                            Refer_Value(num_re+1:num_re+3) = bin2_8(1:3);
                            num_re = num_re + 3;
                        else
                            t = num_RV - num_re;
                               %剩余参考像素二进制序列信息个数
                            Refer_Value(num_re+1:num_re+t) =
                               bin2_8(1:t); %tbit参考像素二进制序列信息
                            num_re = num_re + t;
                            Encrypt_exD(num_exD+1:num_exD+3-t) =
                               bin2_8(t+1:3); %(3-t)bit秘密信息
                            num_exD = num_exD + 3-t;
                        end
                    else
                        if num_exD+3 <= num_D
                            Encrypt_exD(num_exD+1:num_exD+3) =
                               bin2_8(1:3); %3bit秘密信息
                            num_exD = num_exD + 3;
                        else
                            t = num_D - num_exD;
                            Encrypt_exD(num_exD+1:num_exD+t) =
                               bin2_8(1:t); %tbit秘密信息
                            num_exD = num_exD + t;
                        end
```

```
                end
            end
        %--------表示这个像素点可以提取 4 bit信息--------%
        elseif Map_I(i,j) == 3
            %Map=3表示原始像素值的第4MSB与其预测值相反
            if num_side < num_S %辅助信息没有提取完毕
                if num_side+4 <= num_S %4位MSB都是辅助信息
                    Side_Information(num_side+1:num_side+4) =
                        bin2_8(1:4);
                    num_side = num_side + 4;
                else
                    t = num_S - num_side; %剩余辅助信息个数
                    Side_Information(num_side+1:num_side+t) =
                        bin2_8(1:t); %tbit辅助信息
                    num_side = num_side + t;
                    Refer_Value(num_re+1:num_re+4-t) =
                        bin2_8(t+1:4); %(4-t)bit参考像素二进制序列信息
                    num_re = num_re + 4-t;
                end
            else
                if num_re < num_RV
                    %参考像素二进制序列信息没有提取完毕
                    if num_re+4 <= num_RV
                        %4位MSB都是参考像素二进制序列信息
                        Refer_Value(num_re+1:num_re+4) = bin2_8(1:4);
                        num_re = num_re + 4;
                    else
                        t = num_RV - num_re;
                            %剩余参考像素二进制序列信息个数
                        Refer_Value(num_re+1:num_re+t) =
                            bin2_8(1:t); %tbit参考像素二进制序列信息
                        num_re = num_re + t;
                        Encrypt_exD(num_exD+1:num_exD+4-t) =
                            bin2_8(t+1:4); %(4-t)bit秘密信息
                        num_exD = num_exD + 4-t;
                    end
                else
                    if num_exD+4 <= num_D
                        Encrypt_exD(num_exD+1:num_exD+4) =
                            bin2_8(1:4); %4bit秘密信息
                        num_exD = num_exD + 4;
                    else
```

```
                        t = num_D - num_exD;
                        Encrypt_exD(num_exD+1:num_exD+t) =
                            bin2_8(1:t); %tbit秘密信息
                        num_exD = num_exD + t;
                    end
                end
            end
        %--------表示这个像素点可以提取 5 bit信息--------%
        elseif Map_I(i,j) == 4
            %Map=4表示原始像素值的第5MSB与其预测值相反
            if num_side < num_S %辅助信息没有提取完毕
                if num_side+5 <= num_S %5位MSB都是辅助信息
                    Side_Information(num_side+1:num_side+5) =
                        bin2_8(1:5);
                    num_side = num_side + 5;
                else
                    t = num_S - num_side; %剩余辅助信息个数
                    Side_Information(num_side+1:num_side+t) =
                        bin2_8(1:t); %tbit辅助信息
                    num_side = num_side + t;
                    Refer_Value(num_re+1:num_re+5-t) =
                        bin2_8(t+1:5); %(5-t)bit参考像素二进制序列信息
                    num_re = num_re + 5-t;
                end
            else
                if num_re < num_RV
                    %参考像素二进制序列信息没有提取完毕
                    if num_re+5 <= num_RV
                        %5位MSB都是参考像素二进制序列信息
                        Refer_Value(num_re+1:num_re+5) = bin2_8(1:5);
                        num_re = num_re + 5;
                    else
                        t = num_RV - num_re;
                            %剩余参考像素二进制序列信息个数
                        Refer_Value(num_re+1:num_re+t) =
                            bin2_8(1:t); %tbit参考像素二进制序列信息
                        num_re = num_re + t;
                        Encrypt_exD(num_exD+1:num_exD+5-t) =
                            bin2_8(t+1:5); %(5-t)bit秘密信息
                        num_exD = num_exD + 5-t;
                    end
                else
```

```
235                         if num_exD+5 <= num_D
236                             Encrypt_exD(num_exD+1:num_exD+5) =
                                    bin2_8(1:5); %5bit秘密信息
237                             num_exD = num_exD + 5;
238                         else
239                             t = num_D - num_exD;
240                             Encrypt_exD(num_exD+1:num_exD+t) =
                                    bin2_8(1:t); %tbit秘密信息
241                             num_exD = num_exD + t;
242                         end
243                     end
244                 end
245                 %--------表示这个像素点可以提取 6 bit信息--------%
246             elseif Map_I(i,j) == 5
                    %Map=5表示原始像素值的第6MSB与其预测值相反
247                 if num_side < num_S %辅助信息没有提取完毕
248                     if num_side+6 <= num_S %6位MSB都是辅助信息
249                         Side_Information(num_side+1:num_side+6) =
                                bin2_8(1:6);
250                         num_side = num_side + 6;
251                     else
252                         t = num_S - num_side; %剩余辅助信息个数
253                         Side_Information(num_side+1:num_side+t) =
                                bin2_8(1:t); %tbit辅助信息
254                         num_side = num_side + t;
255                         Refer_Value(num_re+1:num_re+6-t) =
                                bin2_8(t+1:6); %(6-t)bit参考像素二进制序列信息
256                         num_re = num_re + 6-t;
257                     end
258                 else
259                     if num_re < num_RV
                            %参考像素二进制序列信息没有提取完毕
260                         if num_re+6 <= num_RV
                                %6位MSB都是参考像素二进制序列信息
261                             Refer_Value(num_re+1:num_re+6) = bin2_8(1:6);
262                             num_re = num_re + 6;
263                         else
264                             t = num_RV - num_re;
                                    %剩余参考像素二进制序列信息个数
265                             Refer_Value(num_re+1:num_re+t) =
                                    bin2_8(1:t); %tbit参考像素二进制序列信息
266                             num_re = num_re + t;
```

```
267                         Encrypt_exD(num_exD+1:num_exD+6-t) =
                                bin2_8(t+1:6); %(6-t)bit秘密信息
268                         num_exD = num_exD + 6-t;
269                     end
270                 else
271                     if num_exD+6 <= num_D
272                         Encrypt_exD(num_exD+1:num_exD+6) =
                                bin2_8(1:6); %6bit秘密信息
273                         num_exD = num_exD + 6;
274                     else
275                         t = num_D - num_exD;
276                         Encrypt_exD(num_exD+1:num_exD+t) =
                                bin2_8(1:t); %tbit秘密信息
277                         num_exD = num_exD + t;
278                     end
279                 end
280             end
281             %--------表示这个像素点可以提取 7 bit信息--------%
282         elseif Map_I(i,j) == 6
            %Map=6表示原始像素值的第7MSB与其预测值相反
283             if num_side < num_S %辅助信息没有提取完毕
284                 if num_side+7 <= num_S %7位MSB都是辅助信息
285                     Side_Information(num_side+1:num_side+7) =
                            bin2_8(1:7);
286                     num_side = num_side + 7;
287                 else
288                     t = num_S - num_side; %剩余辅助信息个数
289                     Side_Information(num_side+1:num_side+t) =
                            bin2_8(1:t); %tbit辅助信息
290                     num_side = num_side + t;
291                     Refer_Value(num_re+1:num_re+7-t) =
                            bin2_8(t+1:7); %(7-t)bit参考像素二进制序列信息
292                     num_re = num_re + 7-t;
293                 end
294             else
295                 if num_re < num_RV
                    %参考像素二进制序列信息没有提取完毕
296                     if num_re+7 <= num_RV
                        %7位MSB都是参考像素二进制序列信息
297                         Refer_Value(num_re+1:num_re+7) = bin2_8(1:7);
298                         num_re = num_re + 7;
299                     else
```

```
300                         t = num_RV - num_re;
                               %剩余参考像素二进制序列信息个数
301                         Refer_Value(num_re+1:num_re+t) =
                               bin2_8(1:t); %tbit参考像素二进制序列信息
302                         num_re = num_re + t;
303                         Encrypt_exD(num_exD+1:num_exD+7-t) =
                               bin2_8(t+1:7); %(7-t)bit秘密信息
304                         num_exD = num_exD + 7-t;
305                     end
306                 else
307                     if num_exD+7 <= num_D
308                         Encrypt_exD(num_exD+1:num_exD+7) =
                               bin2_8(1:7); %7bit秘密信息
309                         num_exD = num_exD + 7;
310                     else
311                         t = num_D - num_exD;
312                         Encrypt_exD(num_exD+1:num_exD+t) =
                               bin2_8(1:t); %tbit秘密信息
313                         num_exD = num_exD + t;
314                     end
315                 end
316             end
317             %--------表示这个像素点可以提取 8 bit信息--------%
318         elseif Map_I(i,j) == 7 || Map_I(i,j) == 8
               %Map=7表示原始像素值的第8MSB(LSB)与其预测值相反
319             if num_side < num_S %辅助信息没有提取完毕
320                 if num_side+8 <= num_S %8位MSB都是辅助信息
321                     Side_Information(num_side+1:num_side+8) =
                           bin2_8(1:8);
322                     num_side = num_side + 8;
323                 else
324                     t = num_S - num_side; %剩余辅助信息个数
325                     Side_Information(num_side+1:num_side+t) =
                           bin2_8(1:t); %tbit辅助信息
326                     num_side = num_side + t;
327                     Refer_Value(num_re+1:num_re+8-t) =
                           bin2_8(t+1:8); %(8-t)bit参考像素二进制序列信息
328                     num_re = num_re + 8-t;
329                 end
330             else
331                 if num_re < num_RV
                       %参考像素二进制序列信息没有提取完毕
```

```
                        if num_re+8 <= num_RV
                            %8位MSB都是参考像素二进制序列信息
                            Refer_Value(num_re+1:num_re+8) = bin2_8(1:8);
                            num_re = num_re + 8;
                        else
                            t = num_RV - num_re;
                                %剩余参考像素二进制序列信息个数
                            Refer_Value(num_re+1:num_re+t) =
                                bin2_8(1:t); %tbit参考像素二进制序列信息
                            num_re = num_re + t;
                            Encrypt_exD(num_exD+1:num_exD+8-t) =
                                bin2_8(t+1:8); %(8-t)bit秘密信息
                            num_exD = num_exD + 8-t;
                        end
                    else
                        if num_exD+8 <= num_D
                            Encrypt_exD(num_exD+1:num_exD+8) =
                                bin2_8(1:8); %8bit秘密信息
                            num_exD = num_exD + 8;
                        else
                            t = num_D - num_exD;
                            Encrypt_exD(num_exD+1:num_exD+t) =
                                bin2_8(1:t); %tbit秘密信息
                            num_exD = num_exD + t;
                        end
                    end
                end
            end
        end
    end
end
```

3.7 图像恢复

在提取完全部辅助信息后，我们编写了一个根据辅助信息恢复原图像的函数。其输入包括载密图像 stego_I、图像加密密钥 Image_key、辅助信息 Side_Information、参考像素信息 Refer_Value、位置图 Map_I、秘密信息的长度 num，以及参考像素的行列数 ref_x 和 ref_y。

首先使用 Refer_Value 恢复参考像素，然后使用提供的加密密钥 Image_key 对图像进行解密。最后，根据 Map_I 和 Side_Information 确定需要恢复的像素的比特位，并进行相应的修改，直到恢复了所有指定的比特位。

详细代码如下：

```
function [recover_I] =
    Recover_Image(stego_I,Image_key,Side_Information,Refer_Value,Map_I,num,ref_x,ref_
    % 函数说明：根据提取的辅助信息恢复图像
    %
        输入：stego_I（载密图像）,Image_key（图像加密密钥）,Side_Information（辅助信息
    % 输出：recover_I（恢复图像）
    [row,col] = size(stego_I); %统计stego_I的行列数
    %% 根据Refer_Value恢复前ref_y列、前ref_x行的参考像素
    refer_I = stego_I;
    t = 0; %计数
    for i=1:row
        for j=1:ref_y
            bin2_8 = Refer_Value(t+1:t+8);
            [value] = Binary_Decimalism(bin2_8);
               %将8位二进制数组转换成十进制整数
            refer_I(i,j) = value;
            t = t + 8;
        end
    end
    for i=1:ref_x
        for j=ref_y+1:col
            bin2_8 = Refer_Value(t+1:t+8);
            [value] = Binary_Decimalism(bin2_8);
               %将8位二进制数组转换成十进制整数
            refer_I(i,j) = value;
            t = t + 8;
        end
    end
    %% 将图像refer_I根据图像加密密钥解密
    [decrypt_I] = Encrypt_Image(refer_I,Image_key);
    %% 根据Side_Information、Map_I和num恢复其他位置的像素
    recover_I = decrypt_I;
    num_S = length(Side_Information);
    num_D = num_S + num; %嵌入信息的总数
    re = 0; %计数
    for i=ref_x+1:row
        for j=ref_y+1:col
            if re >= num_D %嵌入信息的比特位全部恢复完毕
                break;
            end
            %---------求当前像素点的预测值---------%
            a = recover_I(i-1,j);
```

```
39              b = recover_I(i-1,j-1);
40              c = recover_I(i,j-1);
41              if b <= min(a,c)
42                  pv = max(a,c);
43              elseif b >= max(a,c)
44                  pv = min(a,c);
45              else
46                  pv = a + c - b;
47              end
48              %--将原始值和预测值转换成8位二进制数组--%
49              x = recover_I(i,j);
50              [bin2_x] = Decimalism_Binary(x);
51              [bin2_pv] = Decimalism_Binary(pv);
52              %--------表示这个像素点需要恢复 1 bit MSB--------%
53              if Map_I(i,j) == 0
                    %Map=0表示原始像素值的第1MSB与其预测值相反
54                  if bin2_pv(1) == 0
55                      bin2_x(1) = 1;
56                  else
57                      bin2_x(1) = 0;
58                  end
59                  [value] = Binary_Decimalism(bin2_x);
                        %将8位二进制数组转换成十进制整数
60                  recover_I(i,j) = value;
61                  re = re + 1; %恢复1bit
62              %--------表示这个像素点需要恢复 2 bit MSB--------%
63              elseif Map_I(i,j) == 1
                    %Map=1表示原始像素值的第2MSB与其预测值相反
64                  if re+2 <= num_D
65                      if bin2_pv(2) == 0
66                          bin2_x(2) = 1;
67                      else
68                          bin2_x(2) = 0;
69                      end
70                      bin2_x(1) = bin2_pv(1);
71                      [value] = Binary_Decimalism(bin2_x);
                            %将8位二进制数组转换成十进制整数
72                      recover_I(i,j) = value;
73                      re = re + 2; %恢复2bit
74                  else
75                      t = num_D - re; %剩余恢复的bit数
76                      bin2_x(1:t) = bin2_pv(1:t);
```

```
77                  [value] = Binary_Decimalism(bin2_x);
                        %将8位二进制数组转换成十进制整数
78                  recover_I(i,j) = value;
79                  re = re + t; %恢复tbit
80              end
81          %--------表示这个像素点需要恢复 3 bit MSB--------%
82          elseif Map_I(i,j) == 2
                %Map=2表示原始像素值的第3MSB与其预测值相反
83              if re+2 <= num_D
84                  if bin2_pv(3) == 0
85                      bin2_x(3) = 1;
86                  else
87                      bin2_x(3) = 0;
88                  end
89                  bin2_x(1:2) = bin2_pv(1:2);
90                  [value] = Binary_Decimalism(bin2_x);
                        %将8位二进制数组转换成十进制整数
91                  recover_I(i,j) = value;
92                  re = re + 3; %恢复3bit
93              else
94                  t = num_D - re; %剩余恢复的bit数
95                  bin2_x(1:t) = bin2_pv(1:t);
96                  [value] = Binary_Decimalism(bin2_x);
                        %将8位二进制数组转换成十进制整数
97                  recover_I(i,j) = value;
98                  re = re + t; %恢复tbit
99              end
100         %--------表示这个像素点需要恢复 4 bit MSB--------%
101         elseif Map_I(i,j) == 3
                %Map=3表示原始像素值的第4MSB与其预测值相反
102             if re+3 <= num_D
103                 if bin2_pv(4) == 0
104                     bin2_x(4) = 1;
105                 else
106                     bin2_x(4) = 0;
107                 end
108                 bin2_x(1:3) = bin2_pv(1:3);
109                 [value] = Binary_Decimalism(bin2_x);
                        %将8位二进制数组转换成十进制整数
110                 recover_I(i,j) = value;
111                 re = re + 4; %恢复4bit
112             else
```

```
                t = num_D - re; %剩余恢复的bit数
                bin2_x(1:t) = bin2_pv(1:t);
                [value] = Binary_Decimalism(bin2_x);
                   %将8位二进制数组转换成十进制整数
                recover_I(i,j) = value;
                re = re + t; %恢复tbit
            end
        %--------表示这个像素点需要恢复 5 bit MSB--------%
        elseif Map_I(i,j) == 4
           %Map=4表示原始像素值的第5MSB与其预测值相反
            if re+4 <= num_D
                if bin2_pv(5) == 0
                    bin2_x(5) = 1;
                else
                    bin2_x(5) = 0;
                end
                bin2_x(1:4) = bin2_pv(1:4);
                [value] = Binary_Decimalism(bin2_x);
                   %将8位二进制数组转换成十进制整数
                recover_I(i,j) = value;
                re = re + 5; %恢复5bit
            else
                t = num_D - re; %剩余恢复的bit数
                bin2_x(1:t) = bin2_pv(1:t);
                [value] = Binary_Decimalism(bin2_x);
                   %将8位二进制数组转换成十进制整数
                recover_I(i,j) = value;
                re = re + t; %恢复tbit
            end
        %--------表示这个像素点需要恢复 6 bit MSB--------%
        elseif Map_I(i,j) == 5
           %Map=5表示原始像素值的第6MSB与其预测值相反
            if re+5 <= num_D
                if bin2_pv(6) == 0
                    bin2_x(6) = 1;
                else
                    bin2_x(6) = 0;
                end
                bin2_x(1:5) = bin2_pv(1:5);
                [value] = Binary_Decimalism(bin2_x);
                   %将8位二进制数组转换成十进制整数
                recover_I(i,j) = value;
```

```
                re = re + 6; %恢复6bit
            else
                t = num_D - re; %剩余恢复的bit数
                bin2_x(1:t) = bin2_pv(1:t);
                [value] = Binary_Decimalism(bin2_x);
                   %将8位二进制数组转换成十进制整数
                recover_I(i,j) = value;
                re = re + t; %恢复tbit
            end
        %--------表示这个像素点需要恢复 7 bit MSB--------%
        elseif Map_I(i,j) == 6
           %Map=6表示原始像素值的第7MSB与其预测值相反
            if re+6 <= num_D
                if bin2_pv(7) == 0
                    bin2_x(7) = 1;
                else
                    bin2_x(7) = 0;
                end
                bin2_x(1:6) = bin2_pv(1:6);
                [value] = Binary_Decimalism(bin2_x);
                   %将8位二进制数组转换成十进制整数
                recover_I(i,j) = value;
                re = re + 7; %恢复7bit
            else
                t = num_D - re; %剩余恢复的bit数
                bin2_x(1:t) = bin2_pv(1:t);
                [value] = Binary_Decimalism(bin2_x);
                   %将8位二进制数组转换成十进制整数
                recover_I(i,j) = value;
                re = re + t; %恢复tbit
            end
        %--------表示这个像素点需要恢复 8 bit MSB--------%
        elseif Map_I(i,j) == 7
           %Map=7表示原始像素值的第8MSB与其预测值相反
            if re+7 <= num_D
                if bin2_pv(8) == 0
                    bin2_x(8) = 1;
                else
                    bin2_x(8) = 0;
                end
                bin2_x(1:7) = bin2_pv(1:7);
                [value] = Binary_Decimalism(bin2_x);
```

```
                    %将8位二进制数组转换成十进制整数
186                 recover_I(i,j) = value;
187                 re = re + 8; %恢复8bit
188             else
189                 t = num_D - re; %剩余恢复的bit数
190                 bin2_x(1:t) = bin2_pv(1:t);
191                 [value] = Binary_Decimalism(bin2_x);
                    %将8位二进制数组转换成十进制整数
192                 recover_I(i,j) = value;
193                 re = re + t; %恢复tbit
194             end
195         %--------表示这个像素点需要恢复 8 bit MSB--------%
196         elseif Map_I(i,j) == 8  %Map=8表示原始像素值等于其预测值
197             if re+8 <= num_D
198                 bin2_x(1:8) = bin2_pv(1:8);
199                 [value] = Binary_Decimalism(bin2_x);
                    %将8位二进制数组转换成十进制整数
200                 recover_I(i,j) = value;
201                 re = re + 8; %恢复8bit
202             else
203                 t = num_D - re; %剩余恢复的bit数
204                 bin2_x(1:t) = bin2_pv(1:t);
205                 [value] = Binary_Decimalism(bin2_x);
                    %将8位二进制数组转换成十进制整数
206                 recover_I(i,j) = value;
207                 re = re + t; %恢复tbit
208             end
209         end
210     end
211 end
212 end
```

4 实验结果

以 Lena 图像为例，该代码实现结果如下：

图上的四张图像分别为原始图像、利用加密密钥加密后的图像、载密图像以及最终恢复的图像，可以观察到恢复图像与原始图像一致，符合实验预期。

5 方法改进

针对该算法，我们提出以下几点改进思路：

1. 预测优化：引入更先进的预测模型，如深度学习方法，以提高预测精度，减少预测误差，从而更高效腾出空间。
2. 动态编码策略：动态调整哈夫曼编码参数，依据图像特征实时变化，比如纹理复杂度，动态选择最适配的码字典，提高编码效率。
3. 加密增强：结合更加安全的加密方案，如同态加密技术，以应对潜在的侧信道攻击，增强整体系统的安全性。
4. 位图优化: 位图标记与压缩技术结合熵编码，考虑位图概率分布特性，进一步优化压缩率失真，提升嵌入率。

预期通过上述改进，提升算法的嵌入率、增强图像质量，使其在复杂图像和纹理图像上表现更为理想。同时，增强的安全机制将确保算法在云环境下更稳健，满足实际应用需求。