南開大學

信息隐藏技术 课程实验报告

基于秘密共享的矩阵编码信息隐藏算法

小组成员

2112492 刘修铭
2112515 武桐西
2111698 于泽林
2113997 齐明杰
2111408 周钰宸

2024 年 5 月 23 日

摘　　要

本次实验中我们研读一篇信息隐藏领域的文献，学习了文献中基于秘密共享的矩阵编码信息隐藏算法。算法结合了矩阵编码技术和秘密共享技术，加密得到两幅共享图像，接收方从公共信道中接收到上述两幅图像后可以恢复出秘密信息和原始的负载图像，并且实验结果表明：该方法与现有的其他同类方法相比，在相同嵌入量的情况下得到的共享图像质量更优，并且该方法完全可逆。

关键词： 汉明码；矩阵编码；秘密共享；信息隐藏

目录

图表

1 实验要求

在网上搜一篇信息隐藏或数字水印的学术论文，复现其中的算法并对其进行改进和实现。

2 实验原理

信息隐藏技术是一种在数字媒体（如图像、音频和视频）中嵌入秘密信息的技术，通过这种技术可以实现数据的保密传输、版权保护等功能。本文实验采用了基于(7,4)汉明码的矩阵编码信息隐藏算法，该算法结合了矩阵编码技术和秘密共享技术，在保证信息隐藏的同时能够有效地恢复原始图像。

汉明码是一种二进制线性编码，可以用来检测和纠正数据传输中的单个比特错误。一个(7,4)汉明码的含义是：编码后的码字长度为 7，其中包含 4 个数据比特和 3 个校验比特。校验矩阵 H 用于生成和检测码字。[1]

基于(7,4) 汉明码，只需在原始图像中连续选择 7 个像素，通过修改其中一个的像素值就可以隐藏 3 比特的秘密信息。从原始图像中提取 7 个像素的最末位组成一个 7 比特长的二进制序列 x，在其中隐藏一个占 3 比特的八进制秘密信息 s，采用如下公式计算标志位 z

$$z = \left(H \times x^T\right)^T \oplus s \tag{2.1}$$

其中 H 是$(7,4)$ 汉明码校验矩阵. 若 z = 0，原始序列 x 保持不变，若$z \neq 0$，则翻转 x 的第 z 个二进制位，这样就得到了新的二进制序列 y 接收方只需按照如下公式计算就可以恢复出秘密信息 s:

$$s = H \times y^T \tag{2.2}$$

对原始图像来说，每嵌入 3 比特数据，最多改变 1 比特，因此图像失真极小，得到的图像品质非常高。

3 算法步骤

3.1 加密

加密算法的输入包括：秘密信息、负载图片和校验矩阵，输出包括伪装图像I_1和I_2，具体过程如下：

第一步

按照光栅扫描顺序把原始图像划分为一个个不重叠的 1×7 大小的块

$$\left\{C_i |\ i = 1, 2, ..., \frac{H \times W}{7}\right\} \tag{3.3}$$

第二步

依次从每个块中取出每个像素的最低有效位组成一个二进制一维数组即码字

$$\left\{ \mathrm{CW}_i |\ i = 1, 2, ..., \frac{H \times W}{7} \right\} \tag{3.4}$$

第三步

将这些码字CW_i和八进制秘密信息S_i，通过矩阵编码运算，得到标记码字$\mathrm{MCM}_{1\mathrm{i}}$和标志位$z_{1\mathrm{i}}$

第四步

用$\mathrm{MCM}_{1\mathrm{i}}$替换原始图像对应块$C_i$的最低有效位

第五步

重复第三、四步的操作得到负载图像I_1

第六步

与第三步类似，将码字CW_i和上面得到的标志位$z_{1\mathrm{i}}$进行矩阵编码运算，得到另一组标记码字$\mathrm{MCM}_{2\mathrm{i}}$和标志位$z_{2\mathrm{i}}$

第七步

用$\mathrm{MCM}_{2\mathrm{i}}$替换原始图像对应块$C_i$的最低有效位

第八步

重复第六、七步的操作得到负载图像I_2

上述过程图示如下：

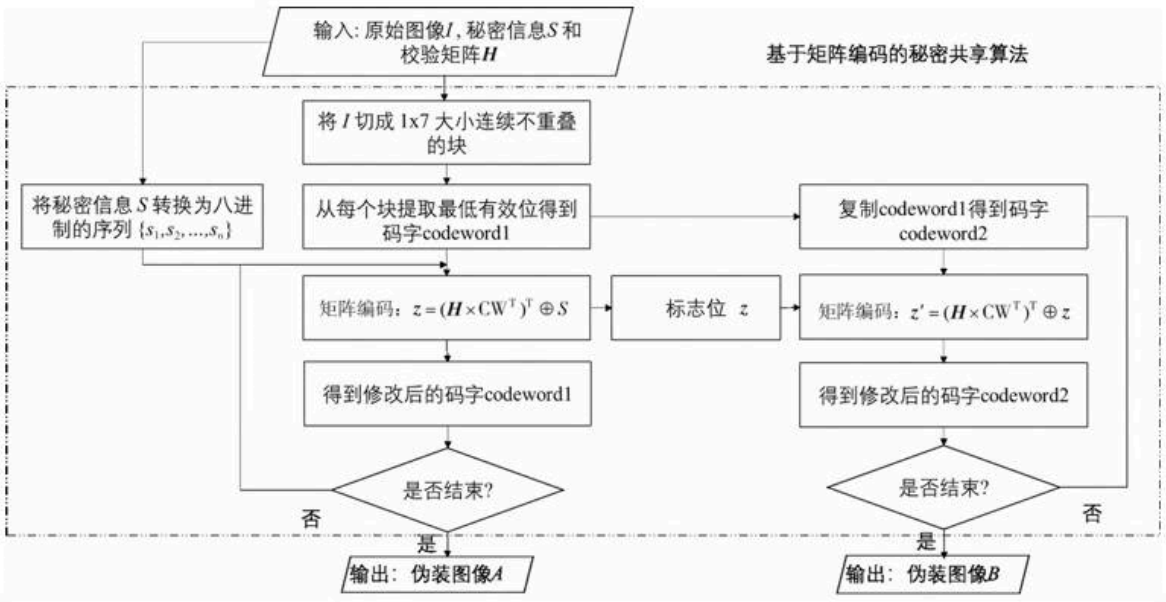

图 3.1.1: 加密流程图

3.2 解密

解密算法的输入包括：负载图像I_1和I_2以及校验矩阵 H，输出包括秘密信息和原始图像 I，具体过程如下：

第一步

按照光栅扫描顺序把图像I_1和I_2划分为一个个不重叠的 1×7 大小的块

$$\left\{C_{1i}, C_{2i} |\ i = 1, 2, ..., \frac{H \times W}{7}\right\} \tag{3.5}$$

第二步

依次从每个块中取出每个像素的最低有效位组成一个二进制一维数组即码字

$$\left\{\mathrm{CW}_{1i}, \mathrm{CW}_{2i} |\ i = 1, 2, ..., \frac{H \times W}{7}\right\} \tag{3.6}$$

第三步

将CW_{1i}和校验矩阵 H 通过矩阵解码运算，得到秘密信息s_i

第四步

重复第三步,将所有得到的s_i连接后得到秘密信息 S

第五步

将CW_{2i}和校验矩阵 H 通过矩阵解码运算，得到标志位z_{1i}

第六步

若$z_{1i} = 0$，则CW_{1i}保持不变，若$z_{1i} \neq 0$，则将CW_{1i}对应位置翻转，得到更新后的CW_{1i}

第七步

用更新后的CW_{1i}替换C_{1i}的最低有效位

第八步

重复第六步和第七步，最终恢复出原始图像 I 上述过程图示如下：

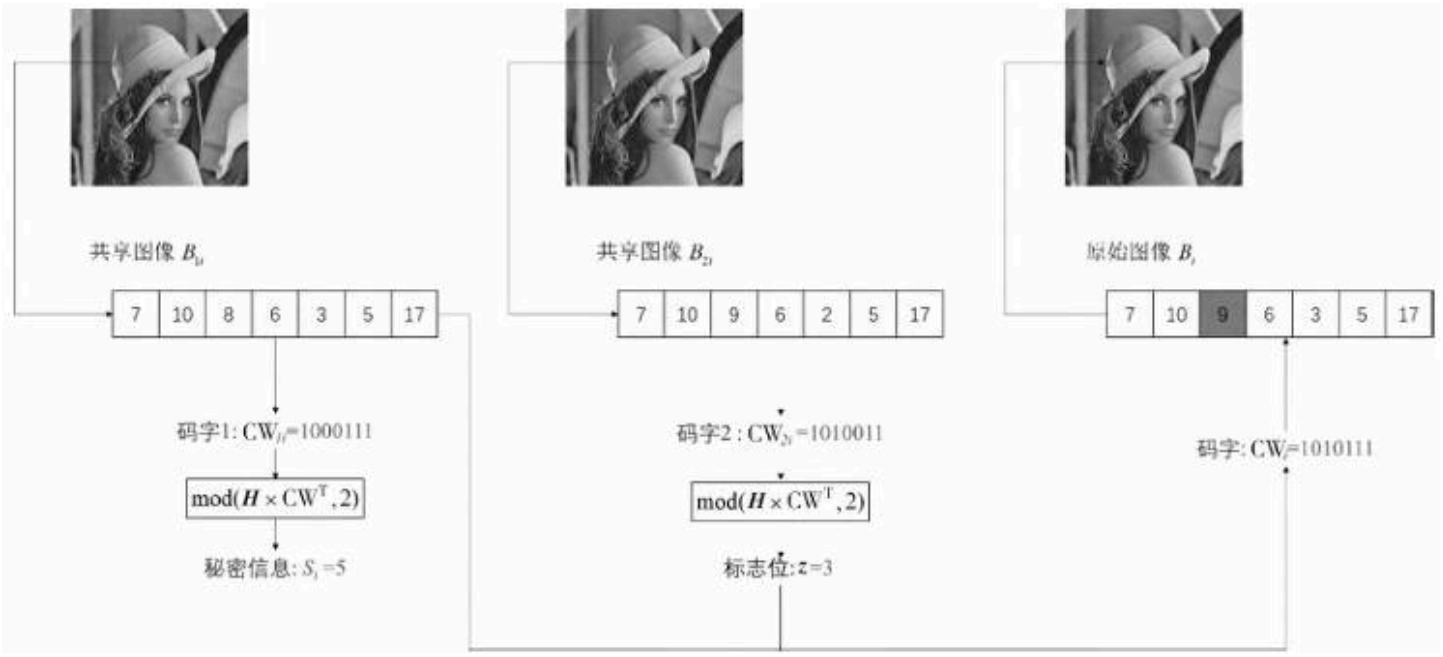

图 3.2.2: 解密流程图

4 代码实现

基础代码实现用到的两张载体图像如下:

图 4.0.3: 载体图像 1

图 4.0.4: 载体图像 2

接下来我要在图一中隐藏一个字符串”Genshin”，然后使用图二复原出图一。 算法类定义如下：

```cpp
class Kobe
{
    public:
        static void hide(const Mat &origin, const vector<uchar> &secrets, Mat &I, vector<uchar> &marks);
        static void extract(const Mat &I, const vector<uchar> &marks, Mat &recover, vector<uchar> &secrets);
        static void hide(const Mat &origin1, const Mat &origin2, const vector<uchar> &secrets, Mat &I1, Mat &I2);
        static void extract(const Mat &I1, const Mat &I2, vector<uchar> &secrets, Mat &recover);
        static vector<uchar> str2oct(const string &str);   // 将字符串 str 转换成八进制数字的数组
        static string oct2str(const vector<uchar> &octs);         // 将八进制数字的数组转换成字符串
    private:
        static uchar getCW(const uchar *p);    // 从图像中某一位置获取并计算一个码字
        static uchar getFlag(uchar &CW);  // 从一个码字中获取标志位(一个八进制数)
};
```

核心成员函数具体实现如下：

```cpp
void Kobe::extract(const Mat &I1, const Mat &I2, vector<uchar> &secrets, Mat &recover)
{
```

```
3       secrets.clear();
4       recover = I1.clone();
5       const uchar *p1 = I1.ptr<uchar>(), *p2 = I2.ptr<uchar>();
6       uchar *p = recover.ptr<uchar>();
7       for (int i = 0; i < SECRET_LEN; ++i)
8       {
9           // 获取需要用到的码字
10          uchar CW1 = getCW(p1);
11          uchar CW2 = getCW(p2);
12
13          // 复原 secret
14          uchar s = getFlag(CW1);
15          secrets.push_back(s);
16          // 复原图像
17          uchar z1 = getFlag(CW2);
18          if (z1 > 0)
19              *(p + z1 - 1) ^= 1;
20          // 光栅扫描方式
21          p += 7;
22          p1 += 7;
23          p2 += 7;
24      }
25  }
26
27  void Kobe::hide(const Mat &origin1, const Mat &origin2, const
    vector<uchar> &secrets, Mat &I1, Mat &I2)
28  {
29      I1 = origin1.clone();
30      I2 = origin2.clone();
31      // const uchar *p = origin1.ptr<uchar>();
32      uchar *p1 = I1.ptr<uchar>(), *p2 = I2.ptr<uchar>();
33      for (int i = 0; i < secrets.size(); ++i)
34      {
35          // 获取需要用到的码字
36          uchar CW1 = getCW(p1);
37          // 计算 z1
38          uchar z1 = getFlag(CW1) ^ secrets[i];
39          // 翻转 I1 中的对应像素
40          if (z1 > 0)
41              *(p1 + z1 - 1) ^= 1;
42          // 计算 z2
43          uchar CW2 = getCW(p2);
44          uchar z2 = getFlag(CW2) ^ z1;
```

```
45          // 翻转 I2 中的对应像素
46          if (z2 > 0)
47              *(p2 + z2 - 1) ^= 1;
48          // 光栅扫描方式
49          // p += 7;
50          p1 += 7;
51          p2 += 7;
52      }
53  }
```

> **解析**
>
> 这段代码实现了基于矩阵编码的秘密共享算法，其中包含了秘密信息的嵌入（hide 方法）和提取（extract 方法）过程。hide 方法首先将两幅原始图像 origin1 和 origin2 分别克隆到 I1 和 I2。在嵌入过程中，它依次获取图像中每个 7 像素组的码字，通过计算和异或操作得到标志位 z1 和 z2。根据这些标志位，它翻转相应像素的最低有效位（LSB），从而将秘密信息嵌入到图像中。通过光栅扫描的方式，依次处理所有像素组，最终生成两幅包含秘密信息的伪装图像 I1 和 I2。 extract 方法则用于从伪装图像中提取秘密信息和恢复原始图像。首先，它将 I1 克隆到 recover，然后通过光栅扫描顺序依次处理每个 7 像素组，获取图像中的码字。利用这些码字和标志位，它从 I1 和 I2 中提取出秘密信息，并在 recover 中恢复原始图像的像素值。通过将这些像素的 LSB 翻转，逐步恢复出原始图像和秘密信息。整个过程确保了秘密信息的安全传输，同时保持了图像的高质量和较低的失真率。

5 实验效果

为检验程序效果，在主函数中编写如下代码:

```cpp
Mat hajimi = imread("resource/hajimi.png", 0), hajiwang = imread("resource/hajiwang.png", 0);
string secret_str = "Genshin";
Mat I1, I2;
Kobe::hide(hajimi, hajiwang, Kobe::str2oct(secret_str), I1, I2);
imwrite("resource/share1.png", I1);
imwrite("resource/share2.png", I2);
vector<uchar> secret_vec;
Mat hajimi_re;
Kobe::extract(I1, I2, secret_vec, hajimi_re);
imwrite("resource/hajimi_re.png", hajimi_re);
cout << Kobe::oct2str(secret_vec) << endl;
```

```
12  if (checkIdenticalImgs(hajimi, hajimi_re))
13    cout << "identical" << endl;
14  else cout << "different" << endl;
```

其中用到了一个辅助函数 checkIdenticalImgs，用于判断两个图像是否完全一模一样，这里我用此函数来判断图像是否复原成功，其实现如下：

```cpp
1  bool checkIdenticalImgs(const Mat &img1, const Mat &img2)
2  {
3      if (img1.type() != img2.type() || img1.size() != img2.size())
4          return false;
5
6      double sum = cv::sum(cv::abs(img1 - img2))[0];
7      return sum == 0;
8  }
```

程序执行完毕，得到了共享图像I_1和I_2：

图 5.0.5: 共享图像 1

图 5.0.6: 共享图像 2

控制台打印出执行结果：

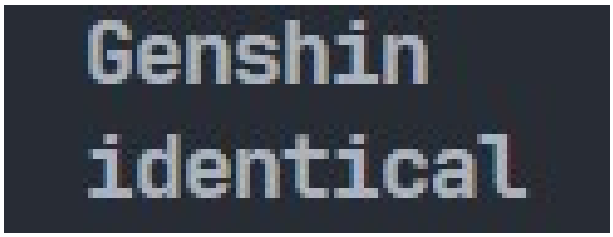

图 5.0.7: 控制台结果

不难看出成功提取出了秘密字符串并且成功复原了图像一。

6 算法改进

改进方案基于论文中反复提及的算法特点，也即此算法能否恢复出载体图像，然而这一点的动机在论文中却没有明确说明。经过思考，本人猜测算法恢复出的“载体图像”可能并不是一张仅仅服务于此算法的图像，它有可能是另一个信息隐藏算法执行得到的带有加密内容的图像，也就是说这个“载体图像”可能事先已经带有了一些秘密信息，如果在接收方成功恢复出“载体图像”，那么就可以根据另一信息隐藏算法的解密算法恢复出原有的秘密信息。

这里采用最基础的 LSB 信息隐藏算法做测试：

```cpp
class LSB
{
    public:
        static void hide(const Mat &origin, const string &secret_str, Mat &I);
        static void extract(const Mat &I, string &secret_str);
};

void LSB::hide(const Mat &origin, const string &secret_str, Mat &I)
{
    I = origin.clone();
    uchar *p = I.ptr<uchar>();
    for (int i = 0; i < SECRET_STR_LEN; ++i)
    {
        for (int j = 0; j < 8; ++j)
        {
            if (((secret_str[i] >> (7-j)) & 1) != (*p & 1))
                *p ^= 1;
            ++p;
        }
    }
}
```

```
22
23 void LSB::extract(const Mat &I, string &secret_str)
24 {
25     secret_str.clear();
26     const uchar *p = I.ptr<uchar>();
27     for (int i = 0; i < SECRET_STR_LEN; ++i)
28     {
29         uchar c = 0;
30         for (int j = 0; j < 8; ++j)
31         {
32             c = (c << 1) | (*p & 1);
33             ++p;
34         }
35         secret_str += c;
36     }
37 }
```

下面尝试将秘密字符串”2111698” 隐藏在图一中，结果作为原图执行本文算法，用于隐藏第二个秘密字符串”Genshin”，整个过程用代码表述即：

```cpp
1  Mat hajimi = imread("resource/hajimi.png", 0), hajiwang =
   imread("resource/hajiwang.png", 0);
2  string secret_str1 = "2111698";
3  Mat hajimi_lsb;
4  LSB::hide(hajimi, secret_str1, hajimi_lsb);
5  string secret_str2 = "Genshin";
6  Mat I1, I2;
7  Kobe::hide(hajimi_lsb, hajiwang, Kobe::str2oct(secret_str2), I1, I2);
8  // imwrite("resource/I1.bmp", I1);
9  // imwrite("resource/I2.bmp", I2);
10 vector<uchar> secret_vec;
11 Mat hajimi_lsb_re;
12 Kobe::extract(I1, I2, secret_vec, hajimi_lsb_re);
13 // imwrite("resource/recover.bmp", recover);
14 string recover_str1, recover_str2;
15 recover_str2 = Kobe::oct2str(secret_vec);
16 LSB::extract(hajimi_lsb_re, recover_str1);
17 cout << recover_str1 << endl;
18 cout << recover_str2 << endl;
```

执行结果：

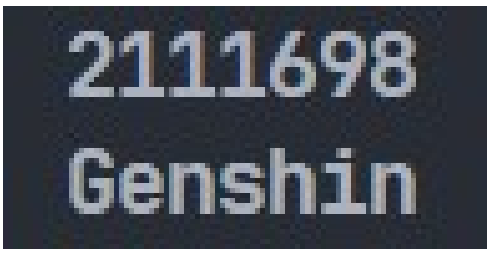

图 6.0.8: 改进算法执行结果

可以看到成功恢复出两个秘密字符串。下面假设另一个场景：攻击者截获共享图像之后，尝试使用常用的信息解密的算法（比如 LSB）提取图像中的秘密信息，在上述改进算法的支持下，他能成功吗？我来模拟这个行为，那就是直接对共享图像 I1，进行 LSB 分析破解。

```cpp
Mat hajimi = imread("resource/hajimi.png", 0), hajiwang = imread("resource/hajiwang.png", 0);
string secret_str1 = "2111698";
Mat hajimi_lsb;
LSB::hide(hajimi, secret_str1, hajimi_lsb);
string secret_str2 = "Genshin";
Mat I1, I2;
Kobe::hide(hajimi_lsb, hajiwang, Kobe::str2oct(secret_str2), I1, I2);
// 尝试直接对 I1 进行 LSB
string recover_str1;
LSB::extract(I1, recover_str1);
cout << recover_str1 << endl;
```

结果如下：

图 6.0.9: 攻击者破解结果

观察到乱码，所以直接攻击会失败。这就意味着如果攻击者事先不知道子算法的细节，也就不知道这些共享图像之间的联系，他对于每个单独图像的破解行为会失败，这是因为矩阵编码算法对于原图做了若干修改(本文基于(7,4) 汉明码矩阵编码算法，若基于(3, 1) 汉明码矩阵编码算法，修改的密度会更高)，这就导致图像受到污染，无法利用常规手段破解。

更进一步，文献中算法只产生了两幅共享图像，我的下一个改进就是让算法可以产生任意幅相互之间存在联系的图像。详细解释：算法共产生 N 幅共享图像构成一条图像链，使用第 N 张图像可以恢复出第 N-1 张图像，使用第 N-1 张图像可以恢复出第 N-2 张图像，以此类推，最后可以得到前 N-1 张图像的原图，如果这 N-1 张原图是通过其他信息隐藏算法（LSB、二值图像隐藏法、变换域隐藏法、叠像术等）嵌入了信

息之后的图像，那么算上第一张图像自带的秘密信息，我们就一共嵌入了 N 份秘密信息，攻击者如果只将这 N 张图像当作单独图像看待的话基本无法提取出秘密信息。

接下来我要做的就是利用上述思想靠着五张图像隐藏小组五名成员的学号，代码如下：

```cpp
vector<string> secret_strs = {"2112492", "2112515", "2113997",
"2111408", "2111698"};
// 先把前四个用 LSB 嵌入到前四张图片
vector<Mat> origins(5);
origins[0] = imread("resource/beila.webp", 0);
origins[1] = imread("resource/xiangwan.webp", 0);
origins[2] = imread("resource/nailin.webp", 0);
origins[3] = imread("resource/jiale.webp", 0);
origins[4] = imread("resource/jiaran.webp", 0);
vector<Mat> shares(5);
for (int i = 0; i < 4; ++i)
  LSB::hide(origins[i], secret_strs[i], shares[i]);
shares[4] = origins[4];
// 把最后一个嵌入到第一幅图片中，产生的标记位嵌入到第二幅，以此类推
vector<uchar> secrets = Kobe::str2oct(secret_strs[4]);
for (int i = 0; i < 5; ++i)
{
  vector<uchar> marks;
  Kobe::hide(shares[i], secrets, shares[i], marks);
  secrets = marks;
}
// 至此嵌入过程结束，下面尝试提取出全部 5 个学号
vector<string> results;
for (int i = 0; i < 4; ++i)
{
  Mat recover;
  vector<uchar> secrets;
  Kobe::extract(shares[i], shares[i+1], secrets, recover);
  if (i == 0)
    results.push_back(Kobe::oct2str(secrets));
  string str_recover;
  LSB::extract(recover, str_recover);
  results.push_back(str_recover);
}
// 打印恢复出的学号
for (int i = 0; i < results.size(); ++i)
  cout << results[i] << endl;
```

结果如下：

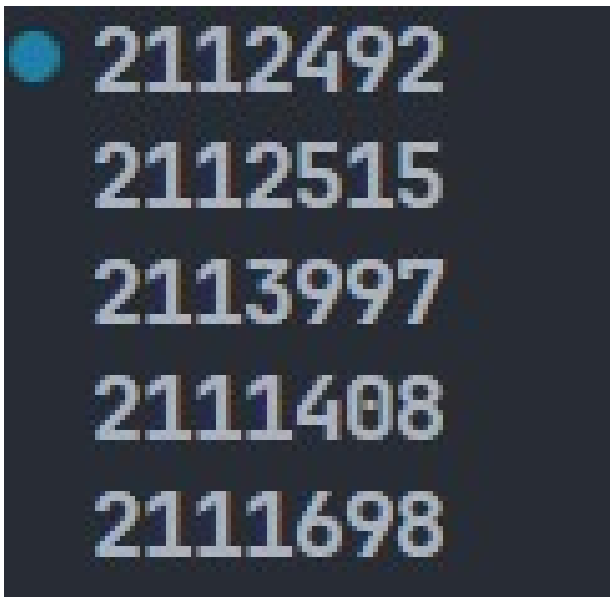

图 6.0.10: 结果 1

这还没有结束，上面采用的是本文算法嵌套别的隐藏算法，如果本文算法里面继续嵌套本文算法会发生什么?

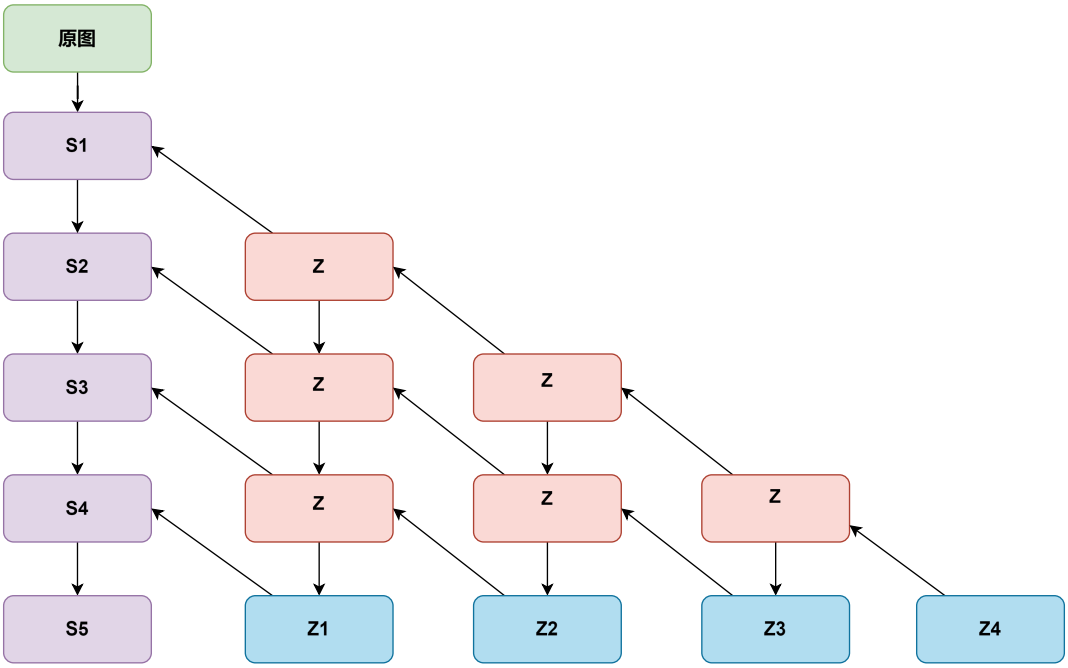

图 6.0.11: 进行嵌套

```cpp
int len = 5;
vector<string> secret_strs = {"2112492", "2112515", "2113997", "2111408", "2111698"};
vector<Mat> origins(len);
origins[0] = imread("resource/beila.webp", 0);
origins[1] = imread("resource/xiangwan.webp", 0);
origins[2] = imread("resource/nailin.webp", 0);
origins[3] = imread("resource/jiale.webp", 0);
```

```
8   origins[4] = imread("resource/jiaran.webp", 0);
9   // 嵌入
10  for (int i = 0; i < len; ++i)
11  {
12    vector<uchar> secrets = Kobe::str2oct(secret_strs[i]);
13    for (int j = 0; j <= i; ++j)
14    {
15      vector<uchar> marks;
16      Kobe::hide(origins[j], secrets, origins[j], marks);
17      secrets = marks;
18    }
19  }
20  // 提取
21  vector<string> results(len);
22  vector<uchar> secrets;
23  Mat recover;
24  for (int i = len-1; i > 0; --i)
25  {
26    for (int j = 0; j < i; ++j)
27    {
28      Kobe::extract(origins[j], origins[j+1], secrets, recover);
29      if (j == 0)
30        results[i] = Kobe::oct2str(secrets);
31      origins[j] = recover;
32    }
33  }
34  Kobe::extract(origins[0], origins[1], secrets, recover);
35  results[0] = Kobe::oct2str(secrets);
36  // 打印
37  for (int i = 0; i < len; ++i)
38    cout << results[i] << endl;
```

结果如下：

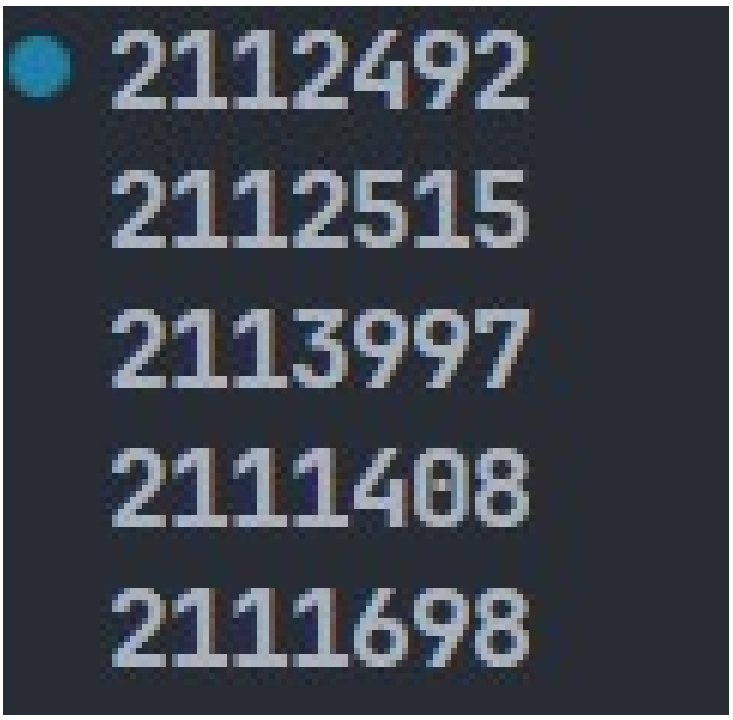

图 6.0.12: 结果 2

7 心得体会

通过此次实验，我深入学习了基于秘密共享的矩阵编码信息隐藏算法，并成功复现和改进了相关算法。该实验不仅让我理解了信息隐藏技术中的核心概念和算法原理，也让我亲身体验了从理论到实际应用的整个过程。在复现过程中，我学会了如何通过矩阵编码技术和汉明码校验矩阵对秘密信息进行有效地编码和解码，从而实现高效、安全的信息隐藏。此外，通过将 LSB 信息隐藏算法与本文算法相结合，我验证了改进算法在信息隐藏和图像恢复方面的优势。

在实际操作中，我进一步感受到算法设计的巧妙之处，尤其是如何在保证信息安全的前提下最大限度地减少对载体图像的影响。此次实验不仅提升了我的编程技能和算法实现能力，也增强了我对信息隐藏技术应用潜力的理解。我意识到，通过不断探索和优化，信息隐藏技术在数字版权保护、数据安全等领域将有广泛的应用前景。总之，此次实验不仅加深了我对信息隐藏技术的认识，也激发了我在这一领域继续研究和探索的兴趣。

参考文献

[1] 余湛;吴红霞;薛醒思, “基于矩阵编码的秘密共享,” 福建技术师范学院学报, vol. 40, no. 5, pp. 477–488, Oct. 2022, [Online]. Available: https://kns.cnki.net/kcms2/article/abstract?v=7qHjwMDcsG2DknGEHDRHwaGAF2Q6HXiB6y7mLBooeG5nQ1vGjbVfF_ZYkiBDqEa0RnzfZ5ssORfsDI3gwOLbLezsSj8arZ6nkZjxkfKkjULvvSj-elL6PZRSFmzrfyRx-PNrfeEevZ8449AMY4mM3g==&uniplatform=NZKPT&language=CHS