浙江大学

课程名称：	多媒体安全
姓　　名：	庞懿非
学　　院：	计算机科学与技术学院
专　　业：	计算机科学与技术
学　　号：	3190104534
指导教师：	黄劲
完成时间：	2022 年 6 月 19 日

实验五：F5 隐写术分析

一、实验目的

1. 了解数字图像隐写术中的典型算法 F5 的基本原理。

2. 了解矩阵编码技术的基本原理。

3. 掌握嵌入效率的计算。

二、实验内容与要求

1. 实现 F5 的隐写系统，包括信息嵌入与信息检测。

2. 设计一份 3KB 左右的文本信息数据，并使用至少两种不同的矩阵编码技术进行信息的嵌入。分析比较不同矩阵编码的嵌入效率，并绘制原图与嵌入后得到的结果图的 DCT 系数直方图。

3. 在 F5 隐写系统中，增加混洗技术，即将 DCT 系数打乱后再使用矩阵编码技术进行信息嵌入。分析混洗技术对信息隐写带来的影响。

三、实验环境

1. **IDE**: MATLAB, Eclipse（Java）

2. **OS**: Windows 10

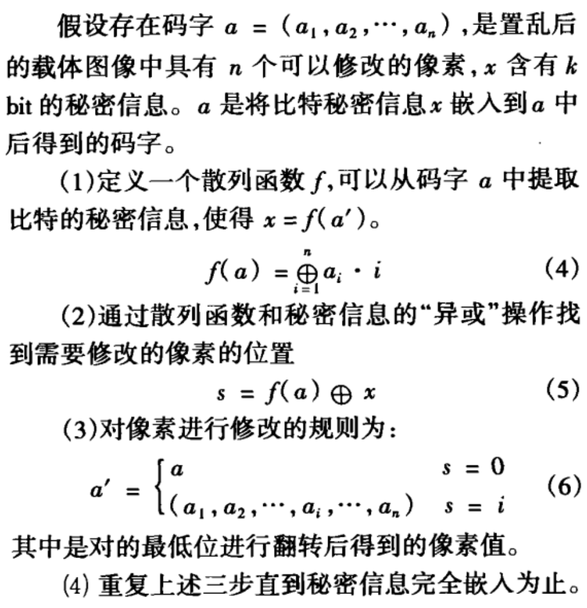
四、实验过程

0. 背景与思路介绍

实验五在实验四的基础上完成，主要是可以直接通过实验 4 的内容获取到 DCT 系数。并且，基于实验 4 的内容，我们知道 DCT 系数在传输过程中可能出错，所以我们直接使用了实验 4 中提到的，从文件中读取隐写后得到的 DCT 系数来解密，避免了可能的错误，更好的分析 F5 隐写系统的性能。

1.矩阵编码实现

不同于课上对矩阵编码的简单讲述，我直接使用了标准的矩阵编码的定义，使之可以完成任意长度信息矩阵编码的需求。



假设存在码字 $a = (a_1, a_2, \cdots, a_n)$，是置乱后的载体图像中具有 n 个可以修改的像素，x 含有 k bit 的秘密信息。a 是将比特秘密信息 x 嵌入到 a 中后得到的码字。

(1)定义一个散列函数 f，可以从码字 a 中提取比特的秘密信息，使得 $x = f(a')$。

$$f(a) = \bigoplus_{i=1}^{n} a_i \cdot i \qquad (4)$$

(2)通过散列函数和秘密信息的“异或”操作找到需要修改的像素的位置

$$s = f(a) \oplus x \qquad (5)$$

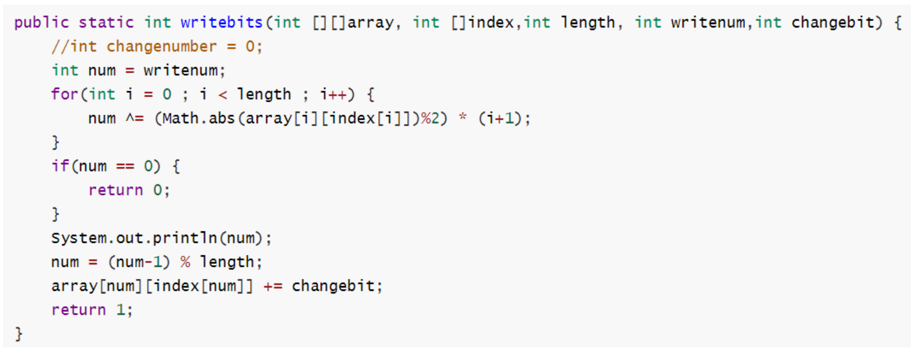
(3)对像素进行修改的规则为：

$$a' = \begin{cases} a & s = 0 \\ (a_1, a_2, \cdots, a_i, \cdots, a_n) & s = i \end{cases} \qquad (6)$$

其中是对的最低位进行翻转后得到的像素值。

(4) 重复上述三步直到秘密信息完全嵌入为止。

简单来说，只要满足上述等式的异或操作，就可以唯一的表示一个定长的信息。式子中 ai 的值用 1 表示奇数，0 表示偶数，既可以在编解码端计算出修改的位置或者解码的信息，如下代码在 *Matrix* 类中实现了矩阵编码：



```java
public static int writebits(int [][]array, int []index,int length, int writenum,int changebit) {
    //int changenumber = 0;
    int num = writenum;
    for(int i = 0 ; i < length ; i++) {
        num ^= (Math.abs(array[i][index[i]])%2) * (i+1);
    }
    if(num == 0) {
        return 0;
    }
    System.out.println(num);
    num = (num-1) % length;
    array[num][index[num]] += changebit;
    return 1;
}
```

当计算结果为 0，则表示不用修改，所以对于 p 位 bit 的信息，有 2^p-1 个信息都需要修改，有一个信息正好不需要修改，每个信息只需要修改 1 位就能完成传输，也就是需要 2^p-1 位 bit 来传输信息，F5 正是使用了这种方法，牺牲了 capacity 但是增大的 embedding efficiency。因为对于 p 位的有效信息，只需要修改 1-1/2^p 就可以完成传输，embedding efficiency 达到了 p/(1-1/2^p)，随着 p 的增大，也会受到图片中像素数目的约束，并不是无限增大的，但是在一定范围内 embedding efficiency 仍然是单调递增的。

```
public static int readbits(int [][]array, int []index,int length) {
    //int changenumber = 0;
    int num = 0;
    for(int i = 0 ; i < length ; i++) {
        num ^= (Math.abs(array[i][index[i]])%2) * (i+1);
    }
    System.out.println(num);
    num = (num) % (length+1);
    return num;
}
```

解码端的计算同样简介明了。

在实验中，我使用了 2 位的矩阵编码和 3 位的矩阵编码，分别需要用 3 位和 7 位来嵌入。

2. F5 系统实现

F5 系统直接从 embedding efficiency 上入手，而不是和 F3F4 一样针对模型来对抗检测。在实际的 F5 系统中，结合 wet paper 来更隐蔽的隐写会有更好的隐蔽效果，不过我们认为使用共享的位置嵌入信息会有更加直观的效果。

F5 系统仍然是使用图片变换得到的 DCT 系数进行编码，但是使用的信息从 bit 串变成定长 bit 串组成的数据组（比如 2bit 信息对应的 0,1,2,3 组），嵌入信息的目标也变成了多位系数共同判断，不过调用矩阵编码的系统可以很容易的实现。

系统编码解码的基本逻辑如下所示：

```
void add_F5(String txtfile,String imagePath,int bitsnumber,boolean if_randomwalk) {
    //read pixels from imagePath and DCT, Quantization, Zigzag
    //use bitsnumber's bits for matrix coding
    //store new DCT cofficient
    //restore the new image
}
```

```
void decode_F5_FromFile(String decodefile,int bitsnumber){
    //read new DCT coefficients
    //decode using matrix coding, get the bitsstream
    //decode the message
}
```

3. Random walk 实现

Random walk 的实现有很多中方式，我对块的顺序进行 random，但是不对系数进行打乱，即交换了块的嵌入顺序，使用下面简单的数据结构存储信息。

```java
static HashMap<Integer, Integer> random_walk = new HashMap<>();
static HashMap<Integer, Integer> un_random_walk = new HashMap<>();
```

随机 random 的过程如下所示：

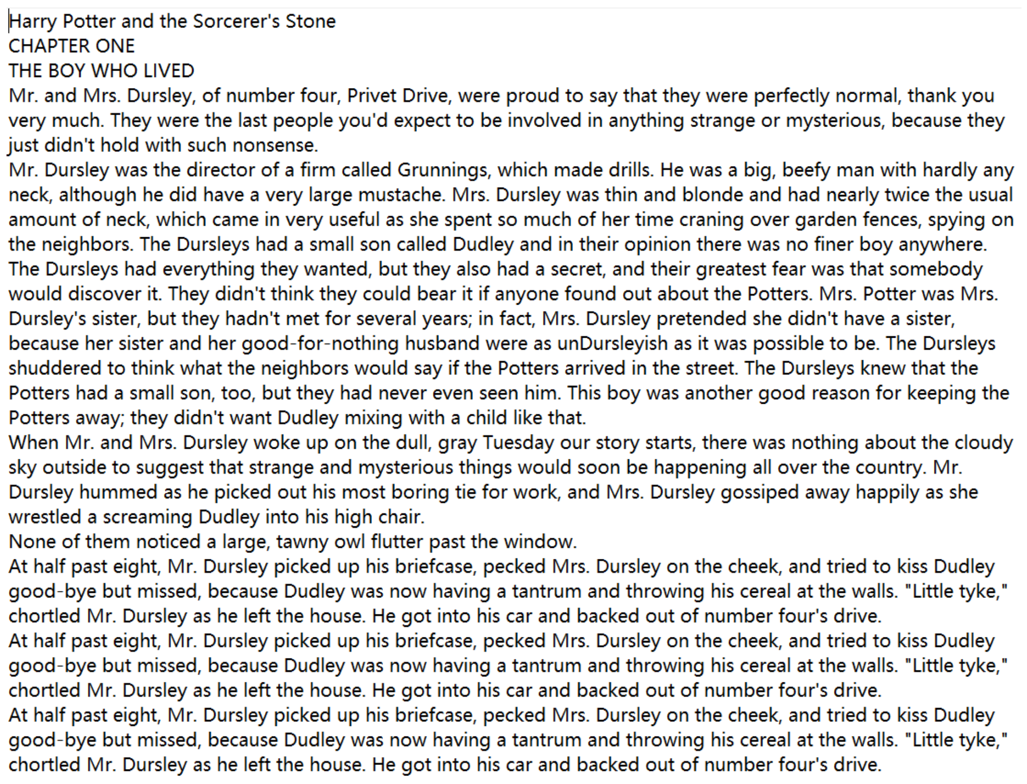
```java
public static void Randomwalk(int [][]input,int times) {
    random_walk.clear();
    un_random_walk.clear();
    Random random = new Random(4534);
    int size = input.length;
    int [][]tmp = new int[1][];
    int num1 = 0, num2 = 0;
    if(times > size / 4) times = size / 4;
    for(int index=0; index < times; index++) {

        while(true) {
            num1 = random.nextInt(size);
            if(random_walk.containsKey(num1) == false) break;
        }
        while(true) {
            num2 = random.nextInt(size);
            if(random_walk.containsKey(num2)== false & num1 != num2) break;
        }

        tmp[0] = input[num1];
        input[num1] = input[num2];
        input[num2] = tmp[0];

        random_walk.put(num1,num2);
        un_random_walk.put(index, num1);
    }
}
```

与上面的数据结构相对应，第一个 *Hashmap* 存储了交换的两个块的块号对应关系，第二个 *Hashmap* 存储了第一个块的块号记录，之后可以快速恢复。当然，由于我设定了随机数种子为我的学号后四位，其实不用这种方式也可以恢复原顺序，但是我们提供了另一种更便捷的思路，每一种方法均可以实现混淆的结果。

可以看到，在 F5 系统中最后一个参数 *if_randomwalk* 就是是否混淆的选择，混淆在对矩阵编码调用前进行打乱，之后生成图片前进行恢复。我们将调用矩阵编码生成的 DCT 系数存储在 code.txt，之后可以用于解码，在 new.txt 中存储了恢复顺序后的 DCT 系数，之后可以对两个进行对比分析，查看混淆带来的帮助。

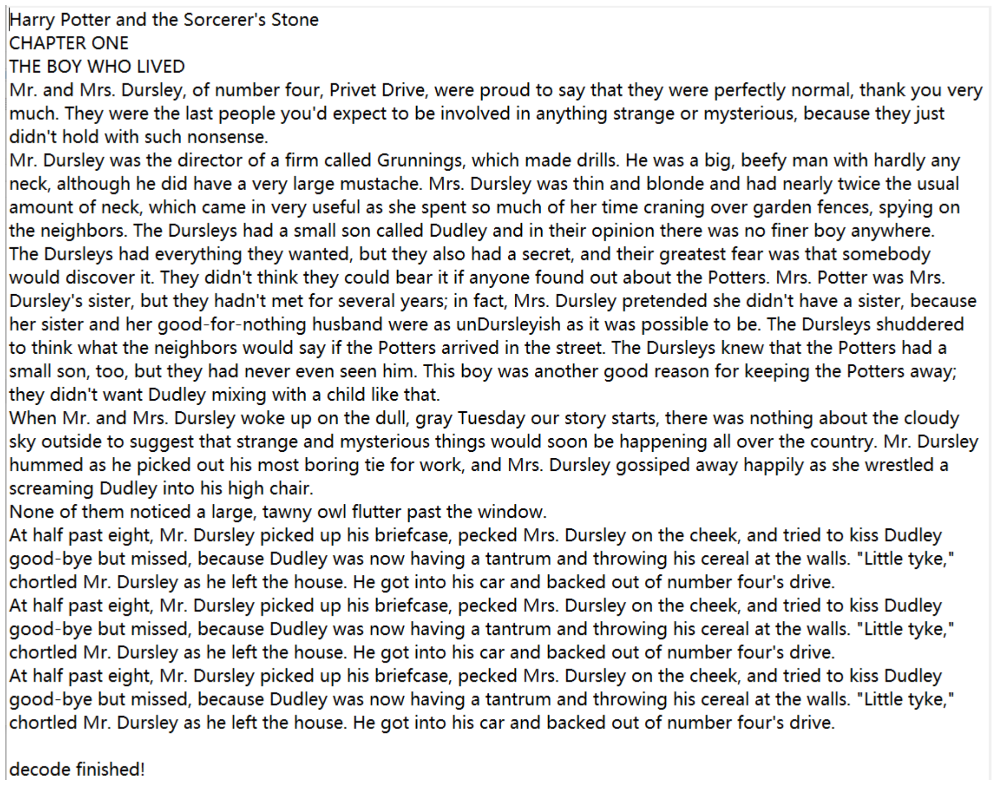
五、实验分析与结论

我使用了一个 3KB 的文本进行测试，观察系统的隐写效果和读取结果，文本文件的内容选择哈利波特英文版。为了简便，只使用英文文本作为输入，使用实验四使用的文本输入系统作为辅助。



Harry Potter and the Sorcerer's Stone
CHAPTER ONE
THE BOY WHO LIVED
Mr. and Mrs. Dursley, of number four, Privet Drive, were proud to say that they were perfectly normal, thank you very much. They were the last people you'd expect to be involved in anything strange or mysterious, because they just didn't hold with such nonsense.
Mr. Dursley was the director of a firm called Grunnings, which made drills. He was a big, beefy man with hardly any neck, although he did have a very large mustache. Mrs. Dursley was thin and blonde and had nearly twice the usual amount of neck, which came in very useful as she spent so much of her time craning over garden fences, spying on the neighbors. The Dursleys had a small son called Dudley and in their opinion there was no finer boy anywhere.
The Dursleys had everything they wanted, but they also had a secret, and their greatest fear was that somebody would discover it. They didn't think they could bear it if anyone found out about the Potters. Mrs. Potter was Mrs. Dursley's sister, but they hadn't met for several years; in fact, Mrs. Dursley pretended she didn't have a sister, because her sister and her good-for-nothing husband were as unDursleyish as it was possible to be. The Dursleys shuddered to think what the neighbors would say if the Potters arrived in the street. The Dursleys knew that the Potters had a small son, too, but they had never even seen him. This boy was another good reason for keeping the Potters away; they didn't want Dudley mixing with a child like that.
When Mr. and Mrs. Dursley woke up on the dull, gray Tuesday our story starts, there was nothing about the cloudy sky outside to suggest that strange and mysterious things would soon be happening all over the country. Mr. Dursley hummed as he picked out his most boring tie for work, and Mrs. Dursley gossiped away happily as she wrestled a screaming Dudley into his high chair.
None of them noticed a large, tawny owl flutter past the window.
At half past eight, Mr. Dursley picked up his briefcase, pecked Mrs. Dursley on the cheek, and tried to kiss Dudley good-bye but missed, because Dudley was now having a tantrum and throwing his cereal at the walls. "Little tyke," chortled Mr. Dursley as he left the house. He got into his car and backed out of number four's drive.
At half past eight, Mr. Dursley picked up his briefcase, pecked Mrs. Dursley on the cheek, and tried to kiss Dudley good-bye but missed, because Dudley was now having a tantrum and throwing his cereal at the walls. "Little tyke," chortled Mr. Dursley as he left the house. He got into his car and backed out of number four's drive.
At half past eight, Mr. Dursley picked up his briefcase, pecked Mrs. Dursley on the cheek, and tried to kiss Dudley good-bye but missed, because Dudley was now having a tantrum and throwing his cereal at the walls. "Little tyke," chortled Mr. Dursley as he left the house. He got into his car and backed out of number four's drive.

输入文本（test1.txt）如上所示，解密文本如下所示，二者没有区别。



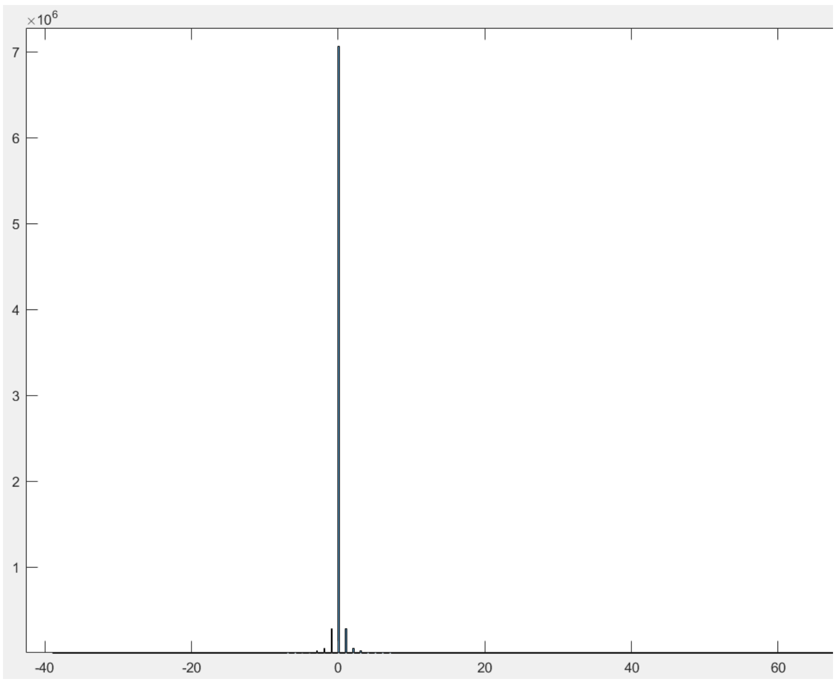
Harry Potter and the Sorcerer's Stone
CHAPTER ONE
THE BOY WHO LIVED
Mr. and Mrs. Dursley, of number four, Privet Drive, were proud to say that they were perfectly normal, thank you very much. They were the last people you'd expect to be involved in anything strange or mysterious, because they just didn't hold with such nonsense.
Mr. Dursley was the director of a firm called Grunnings, which made drills. He was a big, beefy man with hardly any neck, although he did have a very large mustache. Mrs. Dursley was thin and blonde and had nearly twice the usual amount of neck, which came in very useful as she spent so much of her time craning over garden fences, spying on the neighbors. The Dursleys had a small son called Dudley and in their opinion there was no finer boy anywhere.
The Dursleys had everything they wanted, but they also had a secret, and their greatest fear was that somebody would discover it. They didn't think they could bear it if anyone found out about the Potters. Mrs. Potter was Mrs. Dursley's sister, but they hadn't met for several years; in fact, Mrs. Dursley pretended she didn't have a sister, because her sister and her good-for-nothing husband were as unDursleyish as it was possible to be. The Dursleys shuddered to think what the neighbors would say if the Potters arrived in the street. The Dursleys knew that the Potters had a small son, too, but they had never even seen him. This boy was another good reason for keeping the Potters away; they didn't want Dudley mixing with a child like that.
When Mr. and Mrs. Dursley woke up on the dull, gray Tuesday our story starts, there was nothing about the cloudy sky outside to suggest that strange and mysterious things would soon be happening all over the country. Mr. Dursley hummed as he picked out his most boring tie for work, and Mrs. Dursley gossiped away happily as she wrestled a screaming Dudley into his high chair.
None of them noticed a large, tawny owl flutter past the window.
At half past eight, Mr. Dursley picked up his briefcase, pecked Mrs. Dursley on the cheek, and tried to kiss Dudley good-bye but missed, because Dudley was now having a tantrum and throwing his cereal at the walls. "Little tyke," chortled Mr. Dursley as he left the house. He got into his car and backed out of number four's drive.
At half past eight, Mr. Dursley picked up his briefcase, pecked Mrs. Dursley on the cheek, and tried to kiss Dudley good-bye but missed, because Dudley was now having a tantrum and throwing his cereal at the walls. "Little tyke," chortled Mr. Dursley as he left the house. He got into his car and backed out of number four's drive.
At half past eight, Mr. Dursley picked up his briefcase, pecked Mrs. Dursley on the cheek, and tried to kiss Dudley good-bye but missed, because Dudley was now having a tantrum and throwing his cereal at the walls. "Little tyke," chortled Mr. Dursley as he left the house. He got into his car and backed out of number four's drive.

decode finished!

使用 2-3 比特矩阵进行传输（2bit 信息嵌入 3bitDCT 系数）嵌入信息长度为 23800（后续 3 位辅助信息），实际改动为 8826 次，改动比例约为 2.698。与理论值 2/0.75 = 2.667 相差不大。

```
length = 23803
the number of changes = 8826
```

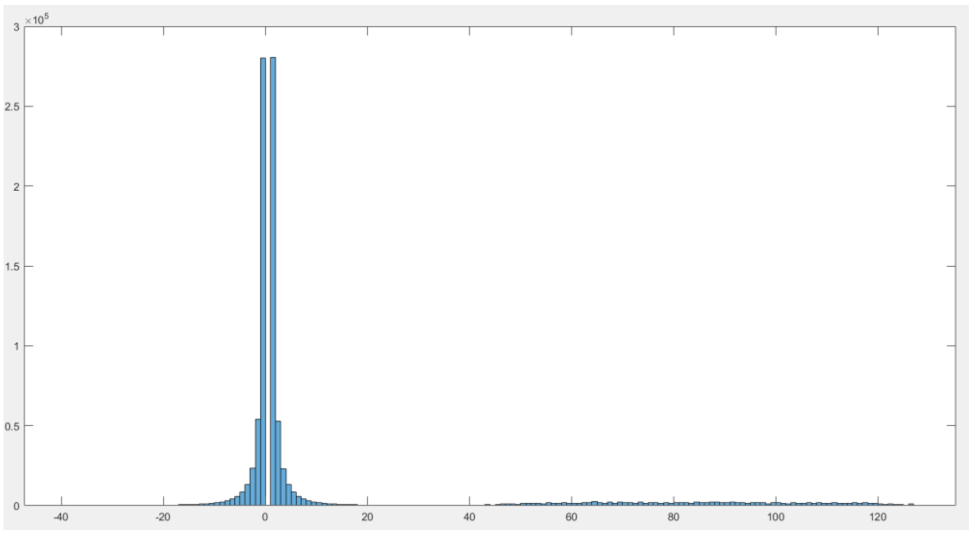
如下两张图，左为原图，右为隐写图，二者从视觉上无差异，那么在统计性质（直方图）上效果将在下面直方图展现。（直方图系数直接保存在文件 ori.txt 和 new.txt 中）



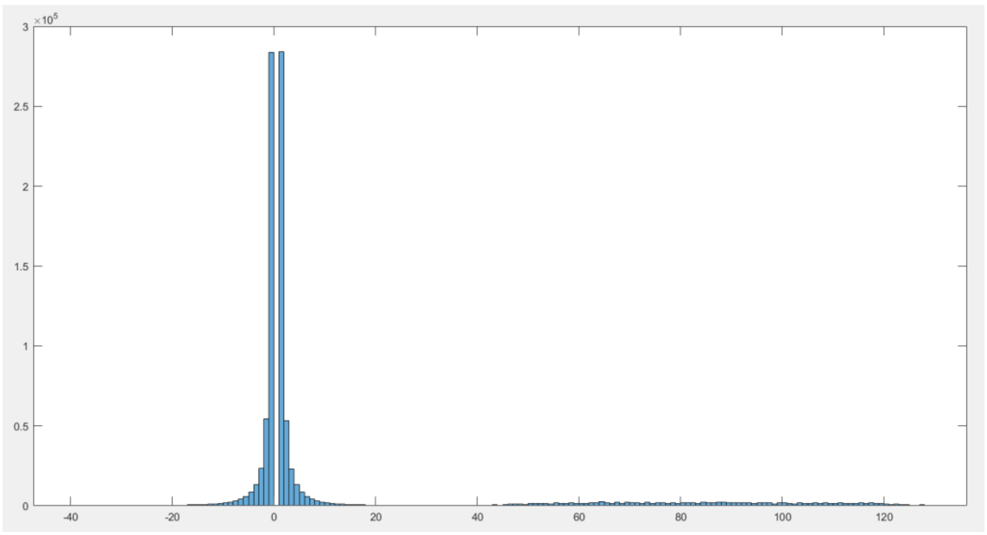
如上是原图（测试全部使用 more_data_48.jpg）由于 DCT 系数中 0 的数量过多，我们直接将 0 去掉查看结果（代码如下）。

```
data=load("ori.txt");
data(data==0) = [];
data=int8(data);
histogram(data)
```

原图示意图效果如下：



隐写之后效果如下：



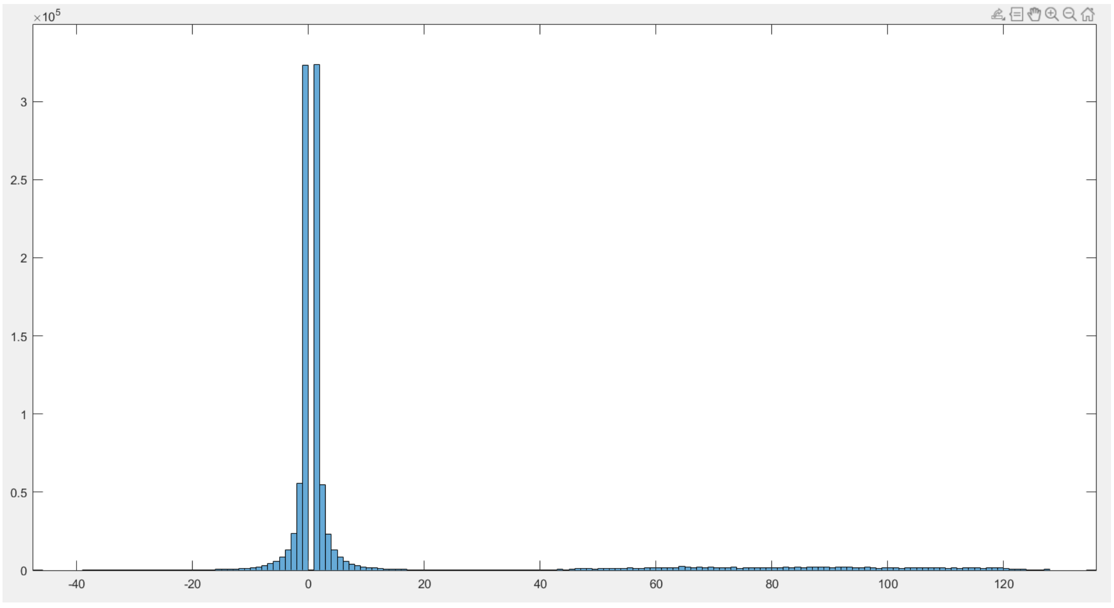
直观上几乎没有变化，我将每个值的统计数绘制表格如下展示具体差异（以正数系数为例，负数同理）：

	1	2	3	4	5	6
原图中个数	280633	52986	22916	13052	8257	5726
2-3bit 隐写	284013	53137	22928	13049	8259	5731

可以看到，每个系数的数量几乎无改动，并且正负非常对称，**在矩阵编码改动期间，在奇偶调整上，我使用+1 和-1 轮换修改，尽量也保证系统整体的稳定性**。

之后，我们隐写 33KB 的英文文本（test2.txt），观察统计值变化。

```
length = 289067
the number of changes = 107023
```



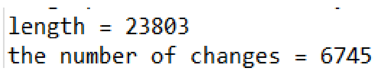
在这个图中，F3 系统的缺陷更加明显了一些，我们统计其数据如下：

	1	2	3	4	5	6
原图中个数	280633	52986	22916	13052	8257	5726
2-3bit 隐写	323701	54544	23021	13047	8282	5745

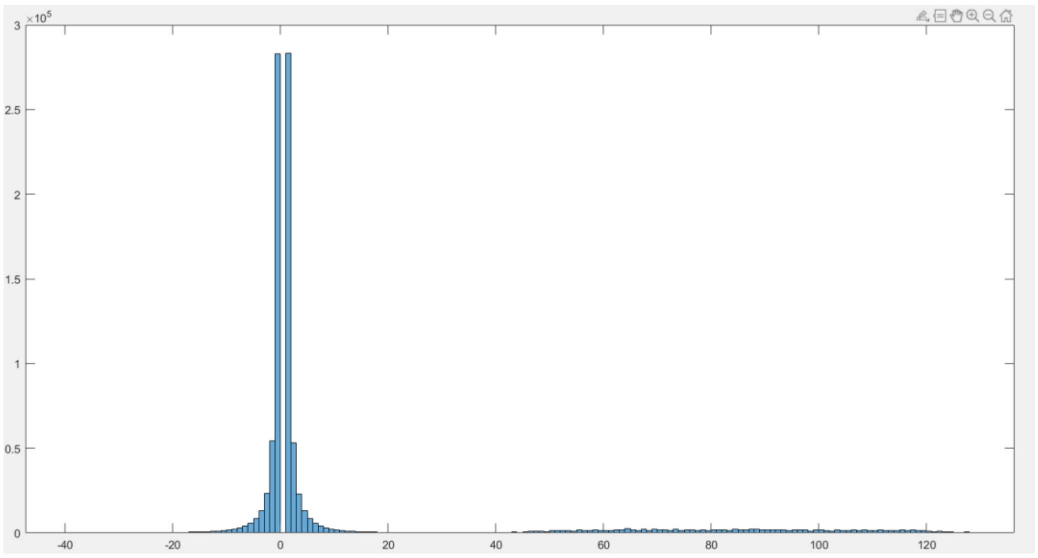
到此，我们看到 embedding efficiency 为 2.701。并且直方图统计上，除了+-1 变化较大（由于 0 改成了 1)，其他均基本无变化，F5 系统的隐写还是非常隐蔽性。

在这里，这张图片还远远没到达隐写容量的上限了，对于这张 2448*3264 的图片，大概可在 2-3 比特矩阵（前面已经解释）嵌入 650KB 的信息，我之后也没有继续测试更大的文本数据。因为 F5 算法会修改很多高频系数，考虑到在实验四中出现的错误，所以我就没有直接从生成的 jpg 图片去解密信息，而是使用了生成图片的 DCT 系数直接去解密信息。如果想直接从图片解密信息，可以尝试多张图片和少量文本进行不断尝试和适配。

同样，对于 3-7 比特矩阵编码（3bit 信息嵌入 7bitDCT 系数），我们只使用 3KB 文本信息进行隐写，结果如下：


```
length = 23803
the number of changes = 6745
```

Embedding efficiency 达到了 3.529，与理论值 3/0.875=3.429 相差不大，但与 2-3 比特矩阵编码 2.69 左右的效果要上升了接近 1，有很大的提升。

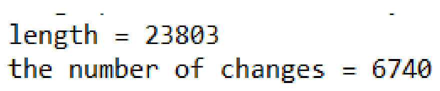


	1	2	3	4	5	6
原图中个数	280633	52986	22916	13052	8257	5726
2-3bit 隐写	284013	53137	22928	13049	8259	5731
3-7bit 隐写	283233	53123	22915	13049	8261	5728

F5 在 3-7 比特矩阵编码的直方图统计结果于 2-3 比特矩阵编码并无太大差异，童昂，生成图片也几乎没有视觉差异。

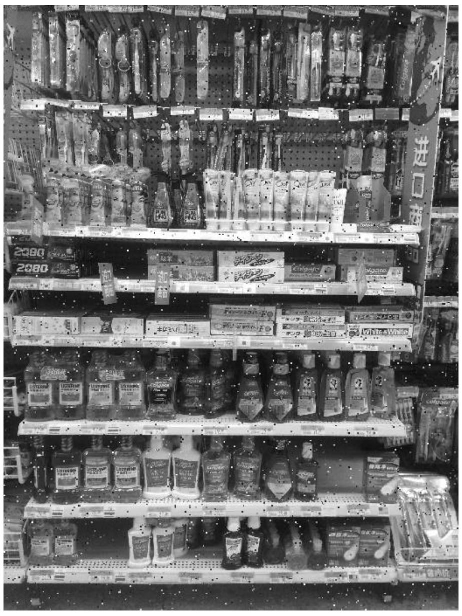
无论是 F3 算法还是 F4 算法，只需要使用简单的 DCT 系数直方图检测就可以破解，也可以使用与 GCD 模型的差异或者比较相似的块做差来获得差异（左为原图，右为隐写图）。

使用 random walk 之后，再次进行 DCT 系数的统计分析，结果如下（使用 3-7bit 编码）：

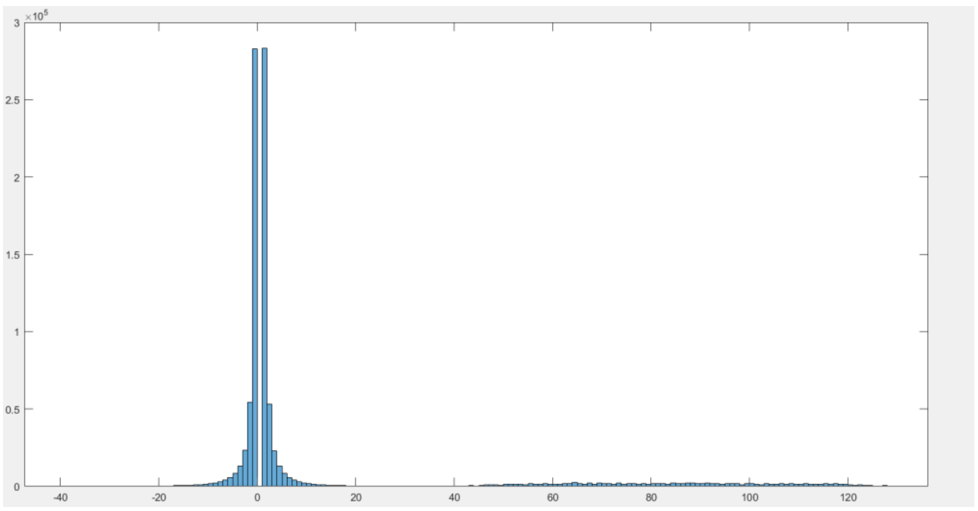


```
length = 23803
the number of changes = 6740
```

Random walk 对嵌入效率影响并不大，对图片质量也是如此（左为原图，右为隐写图）。



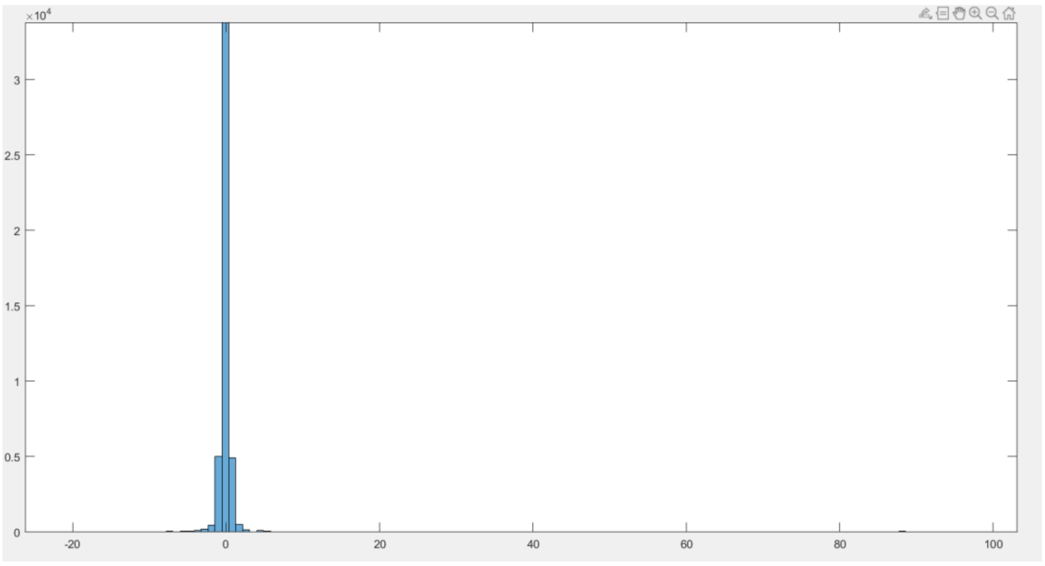
统计全部 DCT 系数与修改部分的 DCT 系数分布：



	1	2	3	4	5	6
原图中个数	280633	52986	22916	13052	8257	5726
3-7bit 隐写	283233	53123	22915	13049	8261	5728
3-7-random 隐写	283265	53118	22910	13056	8257	5728

Random walk 对于整张图片来说并没有太大的变化，但对于顺序读取的文件来说可能有

比较大的变化。（下图为进行矩阵编码的块的统计直方图）



	0	1	2	3	4	5
现有个数	42600	4941	503	156	100	66

上述数据是从 556（23800/2*3/64）个被修改过的块的数据中得到的，不过 F5 统计上确实没有太多的漏洞可以利用。

思考题

混洗技术对信息隐写带来的影响。

Random walk 使得整个图片被随机的加密（但是通信双方知道顺序），warden 很难知道这个图片被隐写的顺序，也很难去找到相似的 block 发现差异（其实每部分被修改的概率都很相近，很难找到一大片区域没有被修改或者被修改很多）。同时，random walk 可以避免顺序统计直方图时出现的反常现象（比如刚开始不符合统计特征但是后面又慢慢符合统计特征，例如 outguess 的方法），打乱顺序使得这一切分析变得非常困难。

六、实验感想

这个实验借鉴了实验四的经验，我没有试图从图片来解码信息，而是直接从 DCT 系数解码信息，更加准确的得到了实验结果。同时，F5 良好的隐写特性，不易被检测的直方图统计特征确实也是我之前没有想到的，尽管 F5 隐写仍然可以被攻击。

当然，在真正隐写时，我们要找到可以完整传输信息而不出错的图片才算隐写成功。

更多的信息可以参考报告内容。