



随机间隔法和随机置换法



为什么要用随机数控制信息隐秘

- 顺序隐秘结果的不可见性差，仔细观察不难发现在图像中隐藏有信息。
- 信息顺序的隐藏到图像中，将不存在密钥的应用空间。任何一个人都可以逐一将秘密信息提取，信息隐藏将毫无意义。

原始图像



隐藏信息后的图像



请注意这条分界线



随机间隔法

- 随机间隔法的思想比较简单，主要是利用随机数的大小来控制前后两个嵌入位的距离。
- 比如我们得到一个长为 N 的服从 $U(0,1)$ 的随机序列 $R=\{r_1, r_2, \dots, r_N\}$ ， N 大于秘密信息长度。取第一个嵌入位为 i ，伪C代码描述有：

```
inbedding address=i;
for(j=1; j<=length(message);j++)
    {if (rj>0.5)
        imbedding address+=k;
    else
        imbedding address+=p;
    }
```



随机间隔法

- 通过判断相应的随机数与0.5的大小，若大于0.5，则选择的嵌入位与前一个嵌入位间隔k-1位，否则间隔p-1位。
- 我们这样定义了k与p:

$total$ =图像载体总像素点;

$quantity$ =为要选择的像素点;

$$k = \left\lfloor \frac{total}{quantity} \right\rfloor + 1$$

$$p = k - 2;$$

随机间隔法

我们在一个 8×8 的范围内对像素进行20点选择，输入：

```
>>test=zeros(8);
```

```
>>[row,col]=randinterval(test,20,1983);
```

```
>>for i=1:20
```

```
    test(row(i),col(i))=i;
```

```
end
```

随机间隔的结果见右图：

seed=1983 的随机序列

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0.09842	0.13928	0.83403	0.47569	0.96968	0.4134	0.96501	0.84192	0.22852	0.68507	0.034194	0.70657	0.38195

	1	2	3	4	5	6	7	8
1	1		2					3
2	4				5			6
3			7					8
4	9				10			11
5			12		13			14
6	15		16		17			18
7	19		20					
8								



随机置换法

设每一个输入 i ,
 i 为小于载体总嵌入
单位数的一个整数。
由 i 均能得到一个数 j_i
表示秘密信息中第 i
个bit相应的嵌入载体
的索引, 且 j_i 不会发
生重复。 j_i 的生成步
骤为:

$$v = [i/X];$$

$$u = i \bmod X;$$

$$v = (v + \text{MD5}(u, k_1)) \bmod Y;$$

$$u = (u + \text{MD5}(v, k_2)) \bmod X;$$

$$v = (v + \text{MD5}(u, k_3)) \bmod Y;$$

$$j_i = vX + u;$$

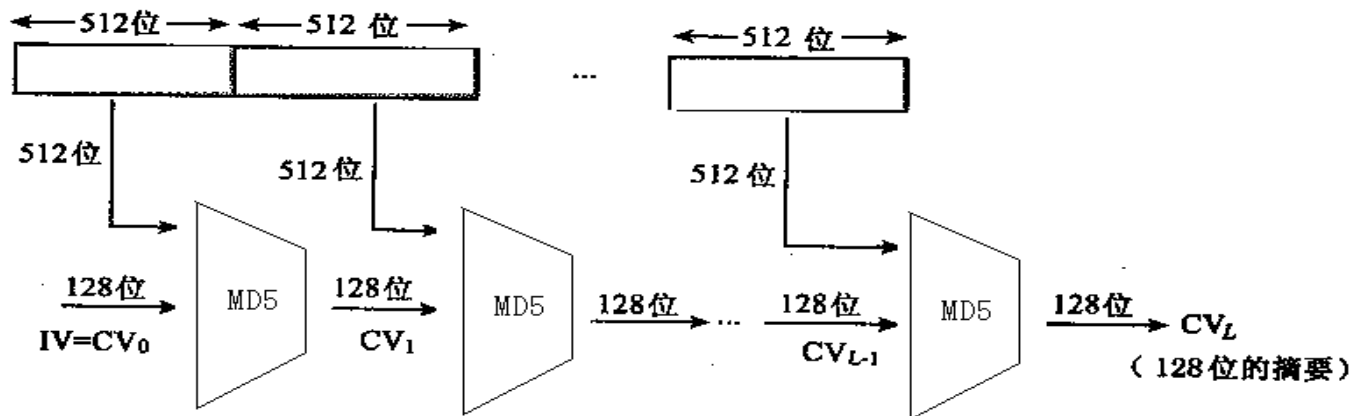


随机置换法

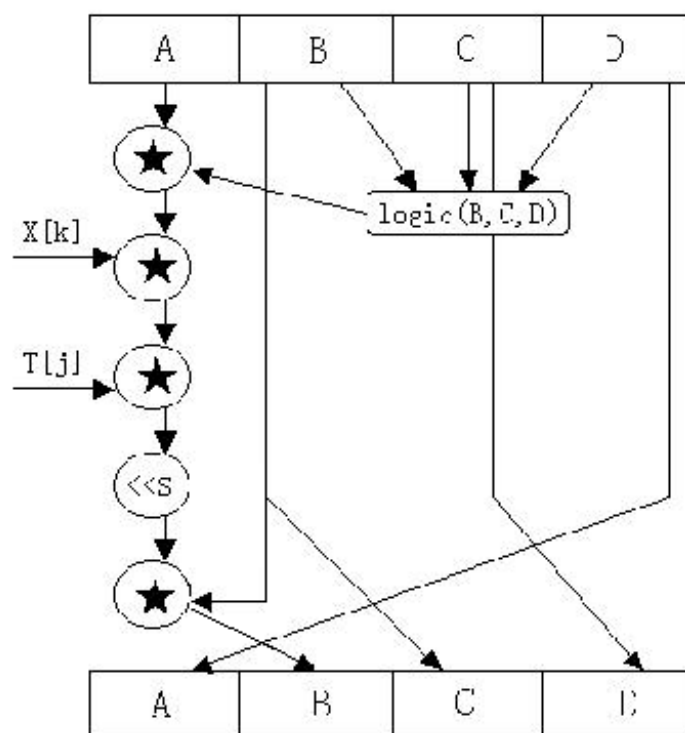
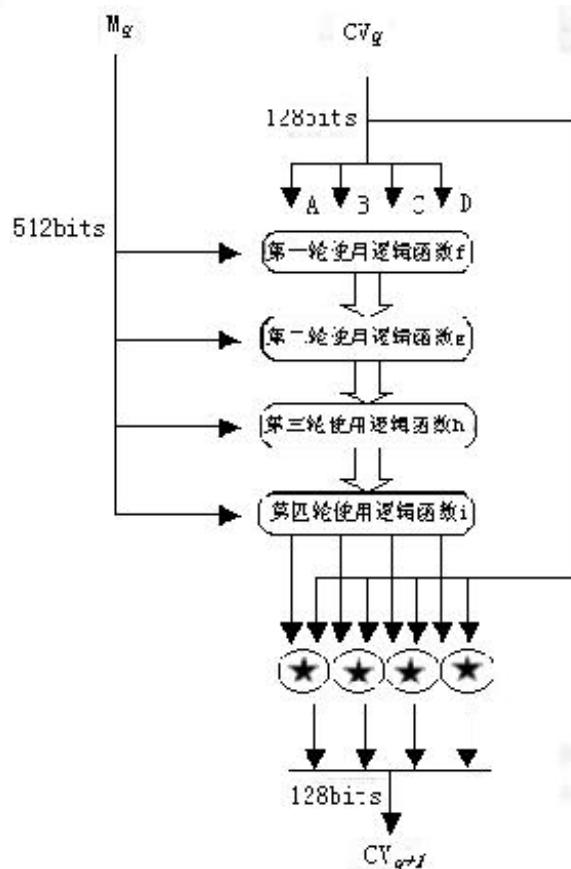
- 这个算法就是一个新的伪随机发生器， $\{j_i\}$ 是一个随机序列。由于这个随机序列仅在地址选择上起作用，所以我们并不考虑它的概率分布和相关性质。
- 使用到安全Hash函数的伪随机置换算法，该算法可以解决碰撞(collision)问题。
- 我们使用的安全Hash函数是MD5。

安全Hash函数：MD5

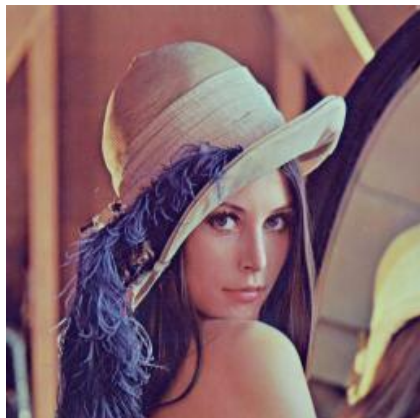
- 填充报文
- 初始化缓冲区
- 循环执行压缩函数
- 输出Hash码



MD5的压缩函数工作框图



MD5输出示例



- lenna图像矩阵的MD5输出:

E6FFB2A9F14AA001D642CEE180B92245

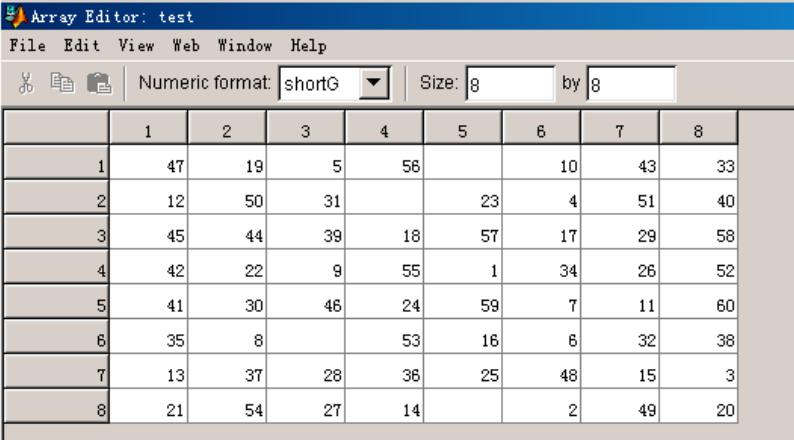
- woman图像矩阵的MD5输出:

46C537D1E0D78EC50FF5A5D5F7F54A25

随机置换法

我们以一个 8×8 的矩阵test为例，用上述方法进行嵌入位选择。输入：

```
>>test=zeros(8);  
>>[row,col,j]=hashreplaceme  
nt(test,60,1983,421,1121);  
>>for i=1:60  
    test(row(i),col(i))=i; end
```



The image shows a screenshot of the 'Array Editor: test' window. The window has a menu bar with 'File', 'Edit', 'View', 'Web', 'Window', and 'Help'. Below the menu bar is a toolbar with icons for cut, copy, paste, and a numeric format dropdown set to 'shortG'. To the right of the toolbar are input fields for 'Size: 8' and 'by 8'. The main area of the window displays an 8x8 matrix of values. The columns are numbered 1 through 8, and the rows are numbered 1 through 8. The values in the matrix are as follows:

	1	2	3	4	5	6	7	8
1	47	19	5	56		10	43	33
2	12	50	31		23	4	51	40
3	45	44	39	18	57	17	29	58
4	42	22	9	55	1	34	26	52
5	41	30	46	24	59	7	11	60
6	35	8		53	16	6	32	38
7	13	37	28	36	25	48	15	3
8	21	54	27	14		2	49	20