其他载体的信息隐藏与水印算法----软件、文本

钮心忻、杨榆、雷敏

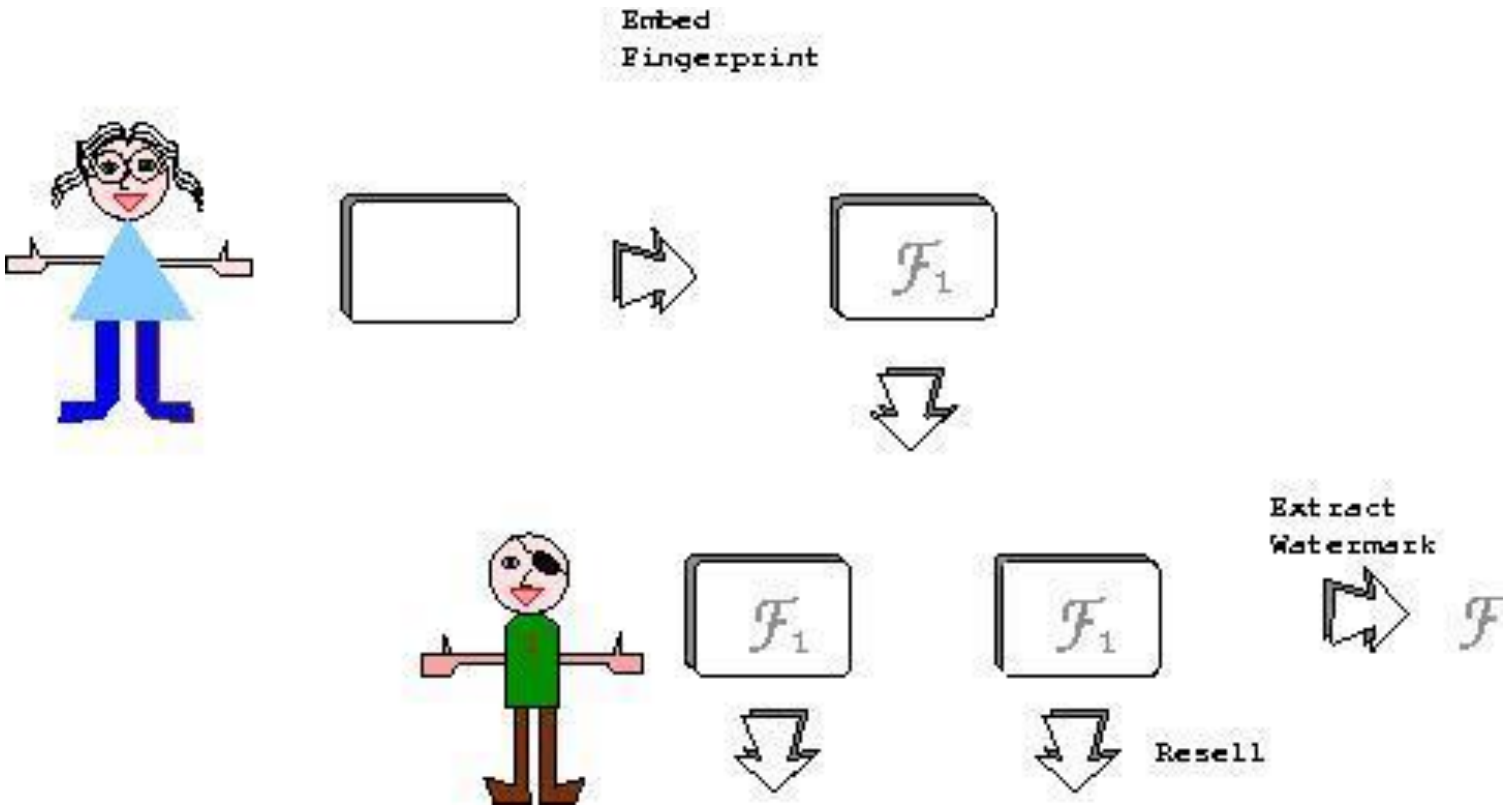
yangyu@bupt.edu.cn

北京邮电大学信息安全中心

软件数字水印技术

- 软件水印
  - 把程序的版权信息和用户身份信息嵌入到程序中
- 软件水印与多媒体水印有本质的不同
- 软件（通常是一段可执行程序）与一般的数字产品不同，它不能在进行大量的、深层次的修改后仍保持原有的特征

应用场景


Embed
Fingerprint
$\mathcal{F}_1$
Extract
Watermark
$\mathcal{F}$
$\mathcal{F}_1$
$\mathcal{F}_1$
Resell

软件数字水印技术

- 对软件水印的要求
  - 能够证明软件的产权所有者
  - 具有鲁棒性。能够抵抗攻击、防止篡改，软件的正常压缩解压以及文件传输不会对水印造成破坏
  - 软件水印不应依赖于某一具体的体系结构
  - 软件水印应该便于生成、分发以及识别
  - 对软件已有功能和特征的影响在实际环境下可以忽略

软件水印的分类

- 按照水印的嵌入位置
  - 代码水印：隐藏在程序的指令部分
  - 数据水印：隐藏在数据中（如头文件、字符串和调试信息等）
- 根据水印被加载的时刻
  - 静态水印：存储在可执行程序代码中
    - 静态代码水印，静态数据水印
  - 动态水印：保存在程序的执行状态中
    - Easter Egg水印，数据结构水印，执行状态水印
    - 动态水印需要有预先输入，根据输入，程序会运行到某种状态，这些状态就代表水印

针对软件水印的攻击

- 保持软件语义的篡改攻击
- 裁剪攻击
- 增添水印攻击
- 共谋攻击

保持软件语义的篡改攻击

- 控制流程变换
  - 插入支路
  - 增加冗余操作数
  - 模块并行化
  - 简单流程图复杂化
  - 循环语句变换
  - 内嵌技术
- 数据变换
  - 数据编码
  - 改变变量的存储方式和生存周期
  - 拆分变量

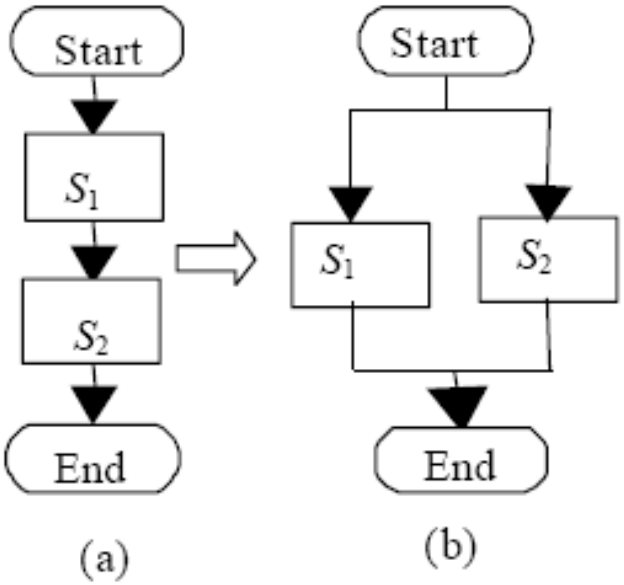
增加冗余操作数

- P=2
- P=Q>0

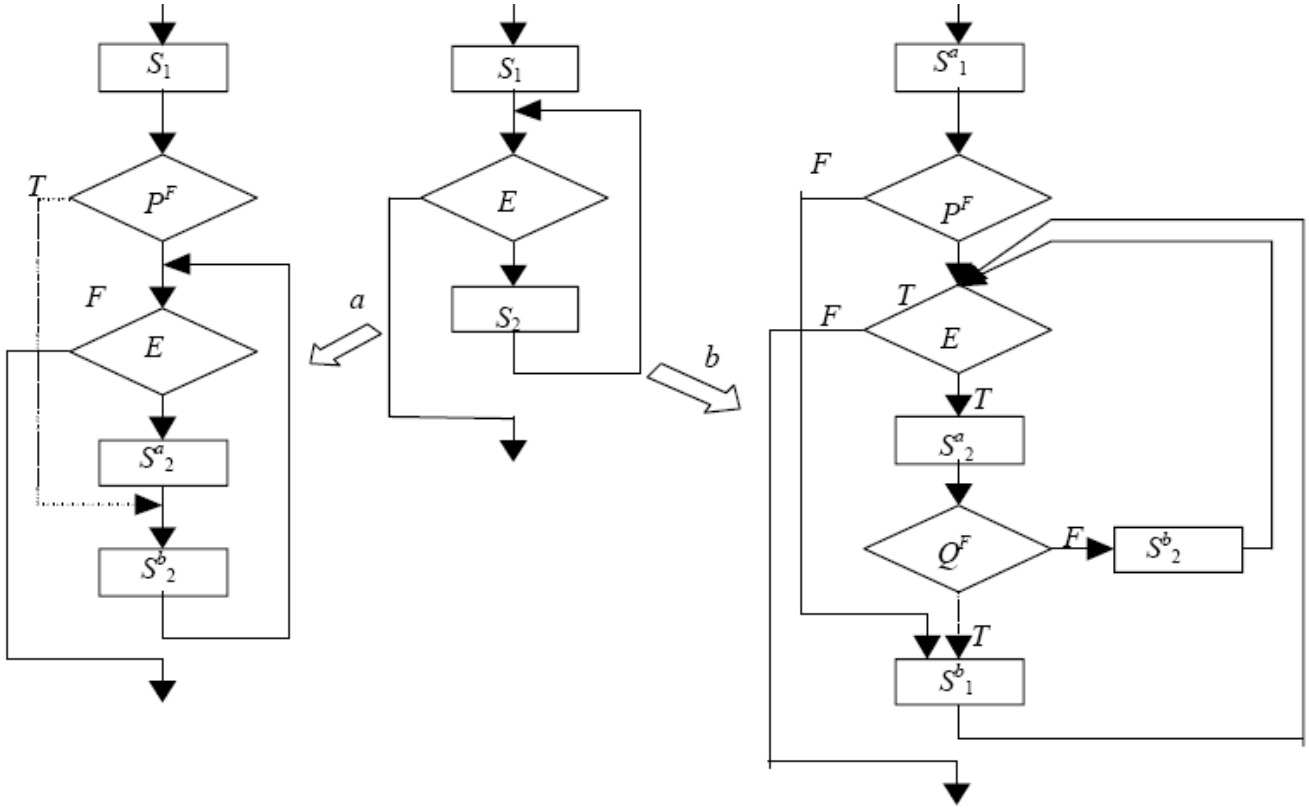
(1) $X=Y+Z$; (1′) $X=P*(Y+Z)/2$;

(2) $X=Y+1$; (2′) $X=Y+\log_Q^P$.

模块并行化



简单流程图复杂化



循环语句变换

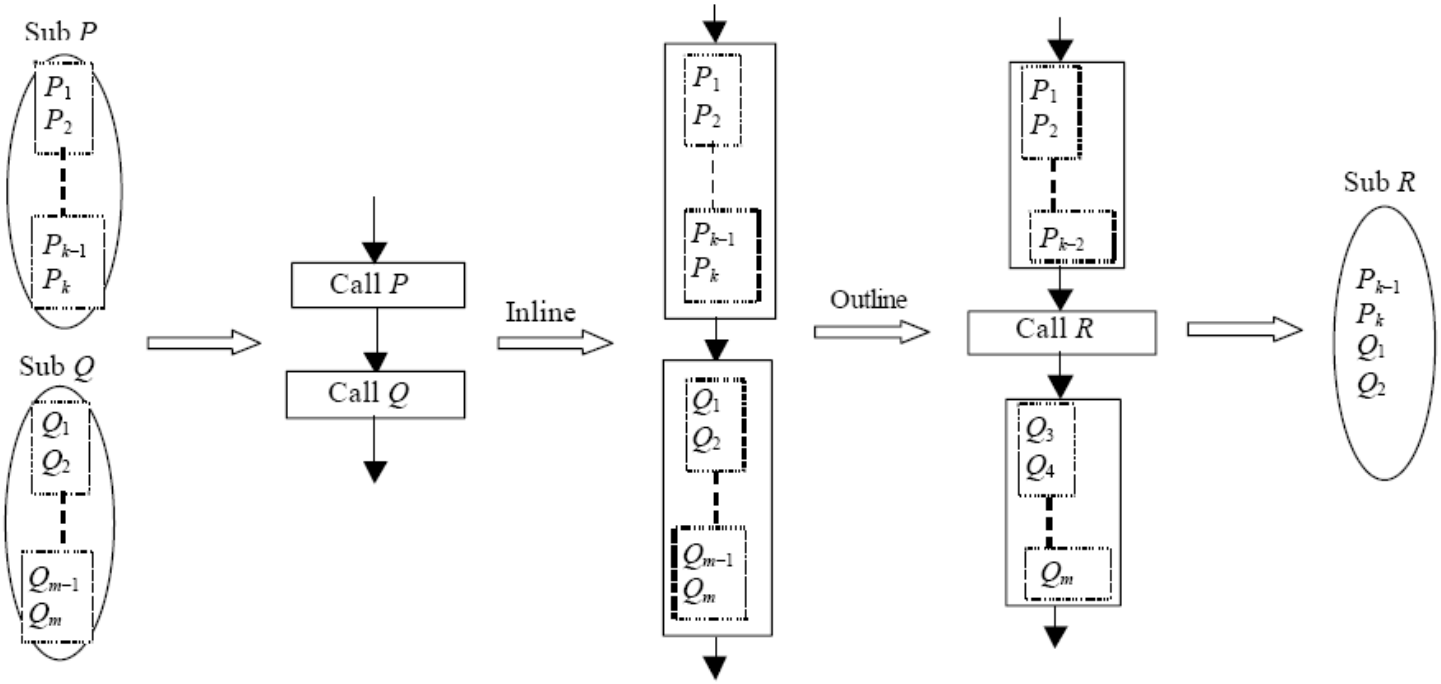
```
for(int i = 0; i < n; i++)
 for(int j = 0; j < n; j++)
   a[i][j] = b[i][j];
```

```
 for(int i = 0; i < n; i+=64)
  for(int j = 0; j < n; j+=64)
   for(int l = i; l < min((i + 63), n); l++)
    for(int m = j; m < min((m + 63), n); m++)
      a[l][m] = b[l][m];
```

```
      for (int i=0; i<n; i++)
{     a[i]=c*b[i];          ⇒
      d[i]=i*(i–1);
}
```

```
for (int i=0; i<n; i++)
      a[i]=c*b[i];
for ( i=0; i<n; i++)
      d[i]=i*(i–1);
```

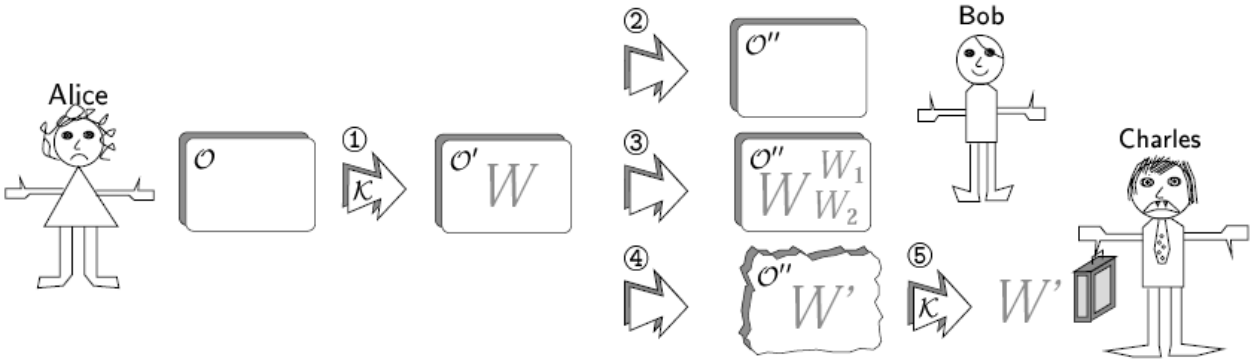
内嵌技术


Sub P
P_1
P_2
P_{k-1}
P_k
Sub Q
Q_1
Q_2
Q_{m-1}
Q_m
Call P
Call Q
Inline
Outline
Call R
P_{k-2}
Q_3
Q_4
Sub R

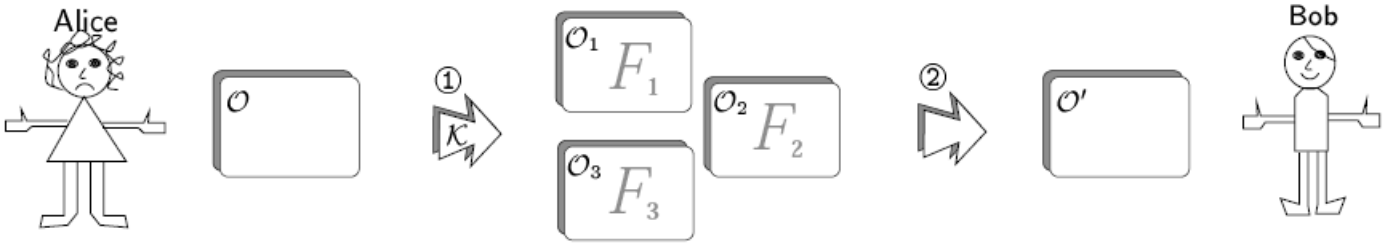
数据编码

```
int i=1;                          int j=c1+c2;
while (i<n){                          while(j<n*c1+c2){
…                          ⇒          …
   A[i]=. .;.                            A[(j–c2)/c1]=. .;.
   …                                  …
}                                      }
```

剪裁、添加、失真攻击



共谋攻击



一种设计良好的水印方案，应该提供针对同一个水印的尽可能多的嵌入方式，最好能够找出同一种方案的水印库，其中含有尽可能多的水印嵌入方式的选择方案，这样可以在生成水印时随机的（或依照某些方案）选择其中一种，从而能够有效的抵抗此种攻击

静态软件水印

- 静态（static）
  - 生成时机在软件的编制过程，而不是运行过程
  - 一旦生成，就不再改变
- 优点
  - 验证方便、快速
  - 生成方式多样和灵活
- 缺点
  - 攻击者容易发现水印存放位置
  - 通用性差：依赖于物理文件格式和具体的程序文件

静态软件水印

- 静态数据水印
  - 将水印信息嵌入在程序的一些数据中，但是它很容易被迷乱攻击破坏。
  - 例：把所有的数据分解成一系列数据，然后散布到整个程序中，这样代表水印信息的数据也被分解，增加了水印检测的困难程度。

静态软件水印

- 静态数据水印
  - 例：用一个产生这些数据的子程序来代替这些数据，这样在程序中就找不到该数据的原型，也就无法检测水印 。

静态软件水印

静态数据水印例：

```
> strings /usr/local/bin/netscape | \
      grep -i copyright
Copyright (C) 1995, Thomas G. Lane
```

静态软件水印

- 静态代码水印
  - 利用目标代码中包含冗余信息嵌入水印。
  - 比如：通过调整两条无依赖关系指令的顺序可以嵌1bit的水印信息。
  - IBM提出了一种把寄存器出入栈的顺序作为水印的方法，同样可以通过排列有m个分支的case语句的顺序来编码比特信息。

静态软件水印

- 静态代码水印

```
switch(i%10){
  case 0://do something
       break;
  case 1://do something
       break;
   …..
  case 9://do something
};
//(0,1,2…..9)
```

```
switch(i%10){
  case 8://do something
       break;
  case 1://do something
       break;
   …..
  case 6://do something
};
//(8,1,2…..6)
```

  - 经不起一些简单的攻击，如调整指令的顺序。

静态软件水印

- 静态代码水印特点
  - 水印嵌入软件代码
  - 抗分析性能强于静态数据软件水印
- 静态代码水印嵌入位置
  - 中间代码
  - 源代码

静态软件水印

- 例：中间代码软件水印
- 嵌入水印
  - 程序中设计不会执行的“死流程”（dummy procedure）
  - 用合法语句填充死流程
  - 编译源代码
  - 用水印替换死流程对应的中间代码
- 提取水印
  - 找到死流程对应位置，提取水印

静态软件水印

- 例： 中间代码软件水印
- 死流程示例：

```
Switch(2*n+1) //将原来的n扩展出条件式
{
 case 0: v = ‘C’;
      break;
 case 1: //一些正常的执行指令
      break;
 case 2: v = ‘o’;
      break;
 case 3: //正常指令
      break;
 case 4: v = ‘p’;    …
}
```

静态软件水印

- 例： 中间代码软件水印
- 死流程示例：

```
//利用永假式来构造不会执行到的
    死流程
if ((7*n*n+1)%7)
{
  //一些正常的执行指令
}
else
{
  char v[]=”copy”;
  char k1 = ‘r’;
  …
}
```

静态软件水印

- 例： 中间代码软件水印
- 水印示例：

指令	编码	指令	编码
iadd	0	irem	4
isub	1	iand	5
imul	2	ior	6
idiv	3	ixor	7

(2003)D->(3723)O-> idiv ...
ixor ...
imul ...
idiv ...

针对静态代码软件水印的攻击

- 自动化的攻击
  - Profiling
    - 列举出所有函数的执行时间，定位从未执行的函数
  - 指令乱序攻击
    - 在不影响软件功能的情况下调整指令的顺序
  - 统计分析攻击
    - 水印多位于不会执行指令，根据软件多次执行的时间分布信息，可以很有效地猜测出水印的隐藏位置，进而加以破坏，甚至篡改
- 手工攻击
  - 相对比较困难，需要对整个程序的理解

动态软件水印

- 动态（dynamic）
  - 生成时机在软件的执行过程中
  - 动态软件水印随着程序运行状态的改变而改变
  - 软件水印的验证和提取必须依赖于软件的具体运行状态，而跟软件文件的内容或存储不相关
  - 生成困难、验证困难、攻击也困难

动态软件水印

- Easter Egg水印
  - 通过一个输入产生输出。
  - 比如：输入一个字符串，屏幕上显示出版权信息或一幅图像。

动态软件水印

- Easter Egg水印
  - 实例： 基于Mozilla开放源代码系统的NetScape 4.0，如果在浏览器的地址栏中输入[about:Mozilla]，就会出现一个喷火的怪兽图案。

动态软件水印

- Easter Egg水印
  - 主要问题：水印在程序中的位置容易找到，一旦输入正确信息，用调试软件就可以跟踪程序执行情况，进而找到水印的位置，所以这种水印不是很安全。

动态数据结构水印

- 涵义
  - 在一定的触发条件下，将具有特殊意义的水印信息存放到**栈、堆或全局变量**中，或者构造特殊的数据结构，比如**图、树或链**表，用数据结构的拓扑属性表示特殊的水印信息。

动态数据结构水印

- 优点
  - 水印仅存在于程序运行空间，而不向用户人机接口输出任何信息，避免了给攻击者以任何提示。

动态数据结构水印

- 验证识别方式
  - 要求水印识别器与包含水印的程序运行在统一进程空间内，或者水印识别器以调试程序的形式来检查软件的内部数据结构的具体值

动态数据结构水印的分类

- 值类型动态数据结构水印
- 拓扑类型动态数据结构水印

值类型动态数据结构水印

静态代码水印

```
char e;
switch (e)
{
 case 1: v =’c’;break;
 case 3: v=’o’; break;
 case 5: v=’p’; break;
 case 7: v=’y’; break;
 case 9: v=’r’; break;
 case 11: v=’i’; …
}
```

值类型动态数据结构水印

```
stack v;
if (getchars() == “hjskei4Xo”)
{
 push (v, ’c’);
 push (v, ‘o’);
 push (v, ‘p’);
 push (v, ‘y’);
 push (v, ‘r’);
 push (v, ‘i’); …
}
```

值类型动态数据结构水印

值类型动态数据结构水印

- 区别
  - 静态代码水印
    - 程序段一般都不会被执行
    - 不需要预先设置触发条件
  - 值类型动态数据结构水印
    - 要触发
    - 与程序流程紧密相关
    - 攻击者无法通过简单统计分析定位
- 验证
  - 水印验证识别程序一直监视软件的堆栈

拓扑类型动态数据结构水印

- 与值类型动态数据结构水印的区别
  - 不需要借助于通用数据结构
  - 直接构造树、图或链表等拓扑数据结构
  - 充分利用现代操作系统的虚拟内存管理方式，使用**指针**或是借助**地址引用**的办法来生成拓扑图。
  - 程序使用的逻辑内存地址在每一次运行时都会被映射到不同的物理内存地址上，这就使得水印信息隐藏在一个不断变化的拓扑图之中，鲁棒性和隐蔽性大大增加

拓扑类型动态数据结构水印

- 与值类型动态数据结构水印的区别
  - 拓扑类型的动态数据结构水印的鲁棒性要好于值类型的水印，
  - 拓扑图中不但可以包含水印信息，同时也具有自身的一些显著的拓扑特征
  - 这样如果攻击者对水印进行了篡改或破坏，不可避免地会带来整个拓扑图的某些特征的改变，我们就可以推测攻击的手段甚至自动恢复原来的水印信息

基于大数分解难题的软件水印方案

- 基本思想
  - 找到某大数是两个足够大素数的乘积
  - 通过拓扑类型的动态数据结构水印隐藏此大数
  - 有限时间内提取出此大数并对其进行有效分解

基于大数分解难题的软件水印方案

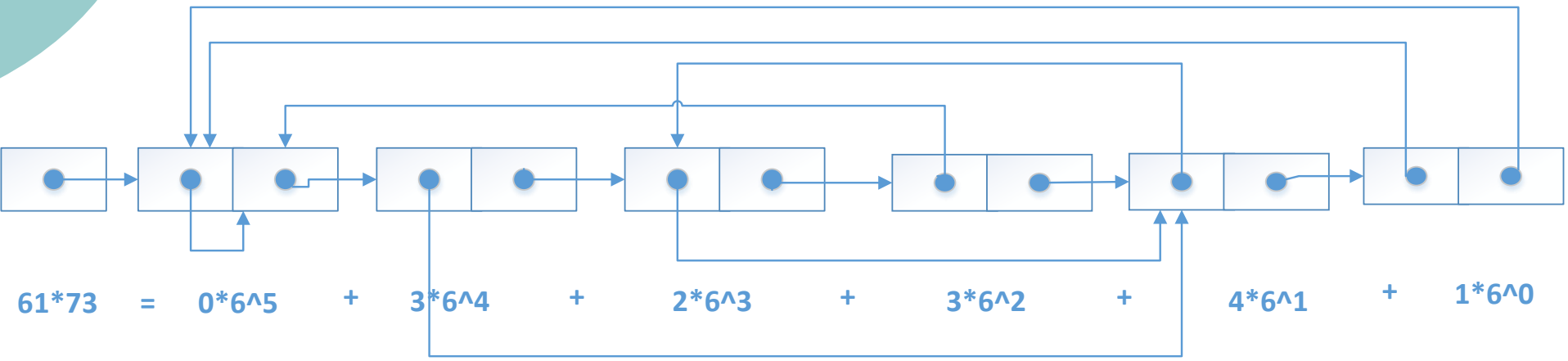
- 要求
  - 大数表示的效率要高：即所采用拓扑图的节点数与表示的整数的大小之比应尽量小
  - 应能表示连续范围内的大数：无论采用何种表示法，必须能够表示连续的整数，其间不能有空缺；
  - 应能在图节点数多项式函数时间内提取出大数信息
  - 对同一个数，应该有尽可能多的表示方法，以对抗共谋攻击。

基于大数分解难题的软件水印方案

- 基数k链表水印
  - 用一个循环双指针链表来构造一个基数k编码（radix-k）的水印，
  - 在链表中除头指针以外，每一个节点的其中一个指针始终指向下一个节点，最后一个节点的指针指向第一个节点；
  - 另外一个指针用来编码水印信息，取值为从这个指针指向的节点（不含）返回原节点需要经过的节点数。这样，指向自身的指针表示0，指向下一个节点的指针表示1，以此类推

基于大数分解难题的软件水印方案

图中所示的链表表示基数6编码的整数4453，作为一个大数水印信息。其质因子为61和73。



基于大数分解难题的软件水印方案

- 例：
  - 已知水印数字为23，基底为3，请问如何用基数图表示这个水印？
  - 解，先将23转换为3进制
  - 23%3=2，23/3=7，**2
  - 7%3=1，7/3=2，*12
  - 2%3=2，2/3=0，212

基于大数分解难题的软件水印方案

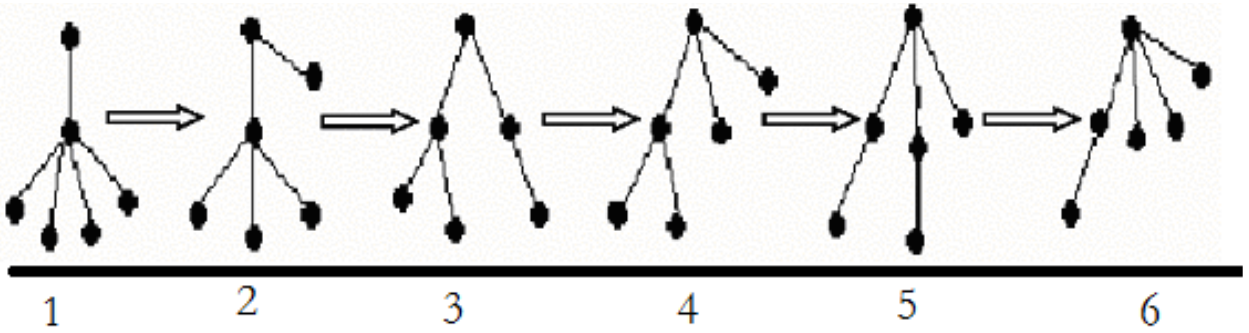
- 例：
  - 已知水印数字为23，基底为3，请问如何用基数图表示这个水印？
  - 然后，确定底数图个节点指向。
  - 除根节点外，其他节点，按据根节点的距离，由近到远依次编号为：2，1，0，对应3^2项系数，3^1项系数，和常数项系数。
  - 则：2号节点左指针应指向0号节点，
  - 1号节点左指针应指向0号节点，
  - 0号节点左指针应指向1号节点，

基于大数分解难题的软件水印方案

- 树枚举水印
  - 用树的枚举信息来表示一个整数，我们定义一个树的枚举方式，然后任意给定一个树，都可以按照定义的枚举方式得出它的顺序枚举值，从而表示一个整数
  - 一个较通用的树比较规则：
    1. 有两个树A和B，若A的深度大于B，则A>B,若A的深度小于B，则A<B;
    2. 若深度相等，则比较节点数：若A的节点数大于B,则A>B，若A的节点数小于B,则A<B;
    3. 若深度，节点数都相等，则比较A与B的子树：规定有较大子树的树较大，如果当前被比较的子树相等，则比较下一个子树

基于大数分解难题的软件水印方案

具有6个节点和深度为3的树，按照从大到小的排序如图



文本文档的信息隐藏

文本文档的信息隐藏

- 文本是信息传递的一种重要媒体，应用的广泛性甚至超过图像、语音等其他媒体形式
- 当电子传播方式被人们广泛接受，文本信息的版权保护和鉴别成为亟待解决的问题（如数字出版物和数字图书馆的出现，大有取代传统书籍之势）

文本文档的分类

- 文本文档
  - 文字信息的数字化表示所形成的电子文件
- 文本文档分类
  - 文本格式
    - 以ASCII码表示内容并包含格式的文档格式文件
    - .doc, .htm, .rtf, .txt
  - 图像格式
    - 以像素点阵的方式描述内容
    - 传真、乐谱

文本文档的分类与特点

- 文本信息的传输形态包括：
  - 电子文档
    - 其文字内容、文字图形特征（连通度、颜色、灰度...）都可以被用于隐藏信息，是其优势。但电子文档传输过程中，其格式可能被重排，其内容可能被编辑，可用于隐藏的“特征”又受到限制
  - 电子文档图像
    - 文档图像的“内容”一般不发生变化，且以数字形态存在时，灰度、色彩、黑白像素比等等都可以用于隐藏信息，但可能遭受图像格式转换，锐化、模糊等图像处理，可用于隐藏的“特征”又受到限制。

文本文档的分类与特点

- 纸介文档
  - 在打印扫描过程中，字符边缘的像素值、字符大小、笔画连通度、位置、角度等均有较大变化，且多以非彩色形式输出，所以可用于隐藏的“特征”不多。
  - 纸介文档还有可能遭受污损、复印、传真等处理，因此算法必须强稳健。

文本文档信息隐藏的特点

- 文本信息隐藏
  - 以一定的方式对文本内容及格式等进行修改，嵌入所需信息但不易被察觉
- 文本文档的特点
  - 数据与内容的高度一致性（对数据的修改直接影响其内容）
- 要求
  - 信息嵌入不影响文本文档的可读性
  - 信息嵌入不在内容表征上产生可被视觉感知的异常

文本信息隐藏的原理及分类

- 文本文档是由内容和格式构成的，内容包括字、词、句、行、段落等元素
- 文本信息隐藏可分为
  - 语义隐藏
  - 格式特征隐藏

  像素翻转隐藏、变换域隐藏、行间距编码、字间距编码、字高编码、字符拓扑编码
  - 显示特征隐藏
  - 零水印

语义隐藏

- 利用语言文字自身及其修辞方面的知识和技巧，通过对原文进行一定规则下的重新排列或剪裁，从而隐藏和提取信息
  - 例如：
  - 根据文字表达的多样性进行同义词置换
  - 将嵌入信息与单词或语句进行映射

语义隐藏

- 基于句法的隐藏方法

变换方式	变换后的句子
未作变换的原始句子	周洋最后夺得了金牌
移动附加语的位置	最后周洋夺得了金牌
加入形式主语	是周洋最后夺得了金牌
主动式变被动式	金牌最后被周洋夺得了
在句子中插入“透明短语”	正如我们看到的，周洋最后夺得了金牌

语义隐藏

- 句法变换
  - bread, butter, and milk
  - bread, butter and milk
- 典型句法变换
  - 主动式-被动式
    - The slobbering dog kissed the boy
    - The boy was kissed by the slobbering dog.

语义隐藏

- 典型句法变换
  - 主题语前置
    - I like bagels.
    - Bagels, I like.
  - 宾语前置
    - I like swimming.
    - Swimming are what I like.
  - 加入形式主语
    - He bought a brand new car.
    - It was a brand new car that he bought.
  - 。 。 。

语义隐藏

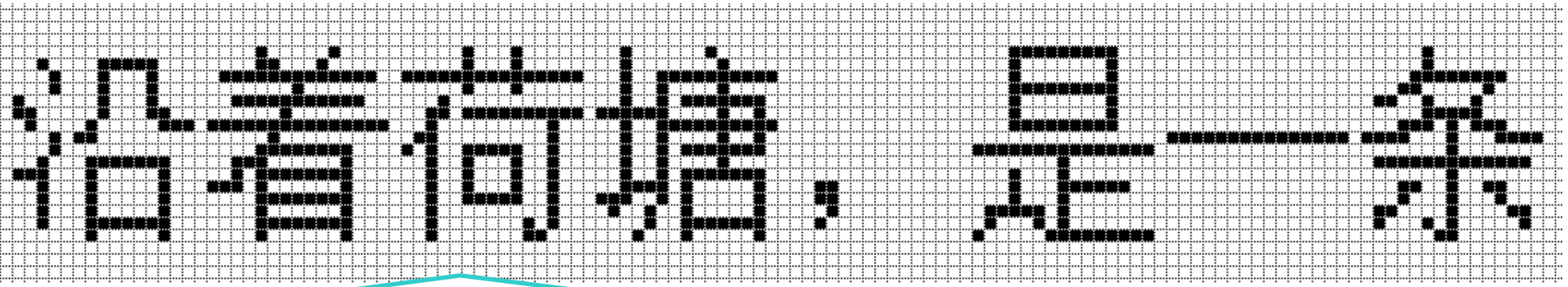
- 同义词替换
  - big large
  - small little
  - chilly cool
  - smart clever
  - pretty lovely

语义隐藏

- 特点
  - 文档内容发生了变化，
  - 不适合打印追踪、印刷品版权认证等场景，
  - 可用于保密通信，
  - 可以抵抗打印扫描，文档格式变化（字体、字号、颜色等等）等处理。

像素翻转隐藏

- 像素翻转隐藏原理
  - 文档图通常为仅包含黑白两种像素的二值图像，因此，许多研究者提出了基于像素翻转的隐藏算法，有的算法仅适于纯数字传输信道，有的算法能够抵抗打印扫描。

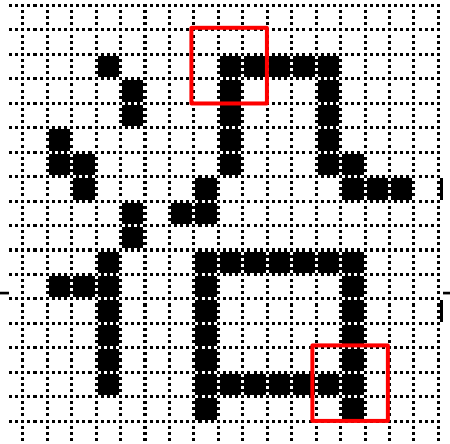


文档经过打印、截屏等操作，由文本变为图像，其语义不变，形态发生了本质变化。上图是《荷塘月色》文档图像局部。可见，一个“字”是二值图像的部份连通区域的组合。

像素翻转隐藏

- 像素翻转基础
  - 文档图像通常具有纹理简单特性，可视为点和线的集合。
  - 随意翻转二值图像的像素，可能会显著降低图像的感官质量。
  - 一个像素是否能被翻转，通常通过其对领域像素的平滑度和连通度所带来的变化，来评估。

像素翻转隐藏



- 像素翻转基础
  - 平滑度度量：以待评估像素为中心的，3×3区域内，水平方向、垂直方向、对角线方向上，像素值的变化次数。
  - 连通度度量：以待评估像素为中心的，3×3区域内，黑白像素块的个数。
  - Wu等研究者，依据这个思路，对69种中心像素的可翻转性进行了评估。得分越高，表示可翻转性越强，也就是翻转中心像素的像素值后，视觉质量下降越小。

像素翻转隐藏

- 像素翻转基础

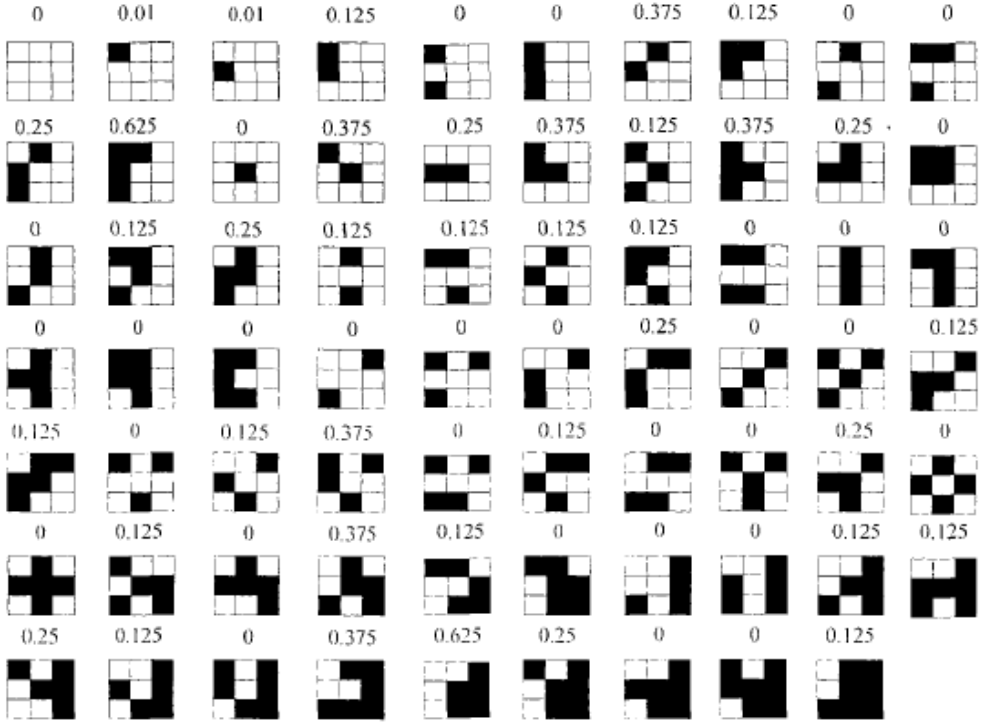
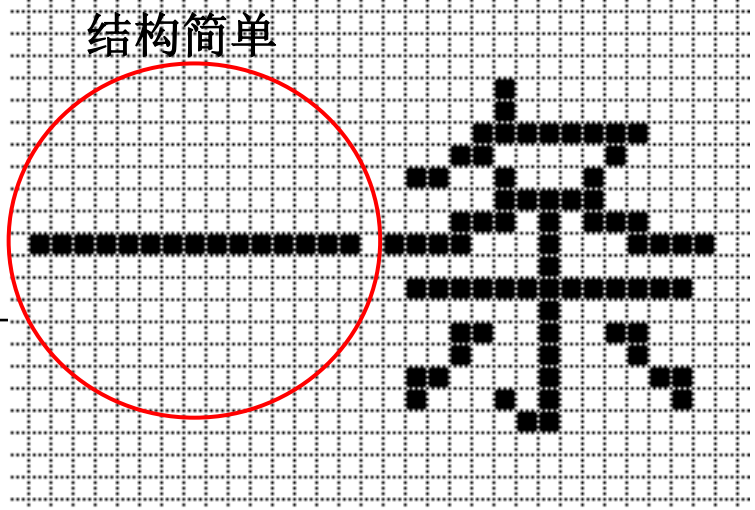


图 4-1 文献[11]所列二值图像像素点可翻转性评分



像素翻转隐藏

- 像素翻转基础
  - 实际使用过程中，如果字符结构过于单一或黑色像素较少时，字符像素得不到充份翻转，则提取时容易受随机因素干扰。
  - 一种有效的解决方法是：根据字符复杂度，筛选可用于隐藏的字符。
  - 根据赵家佑等学者的研究，汉字复杂度与笔画数、笔画节点数、笔画密度、笔画线段量等字符特征关系密切，

像素翻转隐藏

- 像素翻转基础
  - 加之上述特征不受打印扫描影响，因此这些特征可以用作复杂度计算指标，筛选字符。
  - 其中，笔画线段量和笔画数与复杂度关系尤其密切，且较容易计算，算法实现时，可用做实际计算指标。

像素翻转隐藏

- 算法原理
  - 打印扫描包含多个复杂过程，最终扫描所得文档只是看起来与原始文档相似，字符实际发上了较大变化，如下图所示：

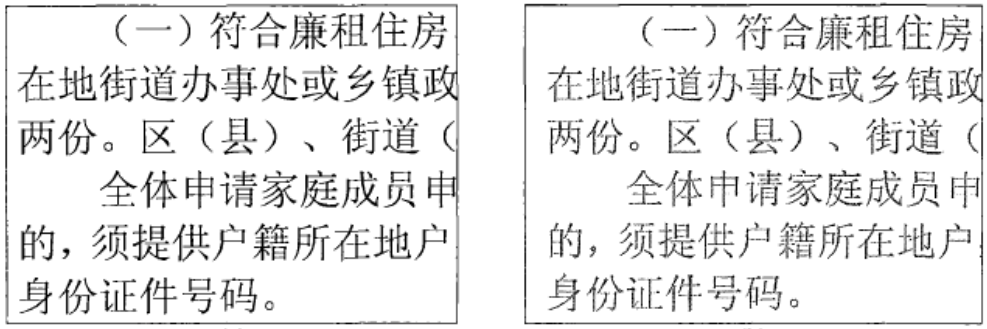


图 4-4 打印扫描前后文本图像对比：(a) 打印扫描前的文本图像；(b) 打印扫描后的文本图像

像素翻转隐藏

- 算法原理
  - 亓文法等研究者结合理论分析与实验观察，得出结论：每个字符所含黑色像素个数，与所在行全部字符平均所包含黑色像素个数的比值，在打印扫描前后基本不变。
  - 亓文法指出，可以利用这个不变量隐藏信息。
  - 具体方法为：
    - 对文档图进行字符分割，把分割的字符分为两组
    - 一组用于嵌入信息，一组用于平衡改动，维持黑色像素个数的均值不变。

像素翻转隐藏

- 算法原理
  - 嵌入方法为：调整字符黑色像素个数，使其与均值的比例为参数k的奇数或偶数倍。
  - 其中，所翻转像素调整应根据前述规则评估，参数k为经验值（实验所得），通常大于1，例如选1.5。
  - 提取方法：计算字符黑色像素个数，全部字符的黑色像素个数的平均值的比值，该值为参数k的奇数倍，则提取1，否则提取0。

像素翻转隐藏

- 例:
  - 已知一行6个文字,黑色像素的像素个数分别是
  - 12，19，11，14，18，16
  - 嵌入策略定为：嵌入0，则把比值调整为0.8；嵌入1，则把比值调整为1.2；一半汉字用于嵌水印，一半用于平衡，使嵌入前后均值不变；
  - 那么嵌入0,1,1三个比特后，这行汉字的黑色像素变为?
  - 比值即:
  - 汉字黑色像素个数/汉字所在行所有汉字平均黑色像素个数

像素翻转隐藏

- 例:
  - 已知一行6个文字,黑色像素的像素个数分别是
  - 12，19，11，14，18，16
  - 解:
  - 均值为：(12+19+11+14+18+16)/6=15;
  - 嵌入0时，汉字黑色像素个数应调整为:
    15*0.8=12;
  - 嵌入1时，汉字黑色像素个数应调整为:
    15*1.2=18

像素翻转隐藏

- 例:
  - 已知一行6个文字,黑色像素的像素个数分别是
  - 12，19，11，14，18，16
  - 解:
  - 要嵌入0,1,1，因此前三个汉字的黑色像素个数为：12，18，18
  - 增加了(12-12)+(18-19)+(18-11)=6个黑色像素，后三个汉字应减少6个黑色像素
  - 均分变更，后三个汉字黑色像素个数变为:
  - (14-2)=12，(18-2)=16，(16-2)=14

像素翻转隐藏

- 算法视觉效果

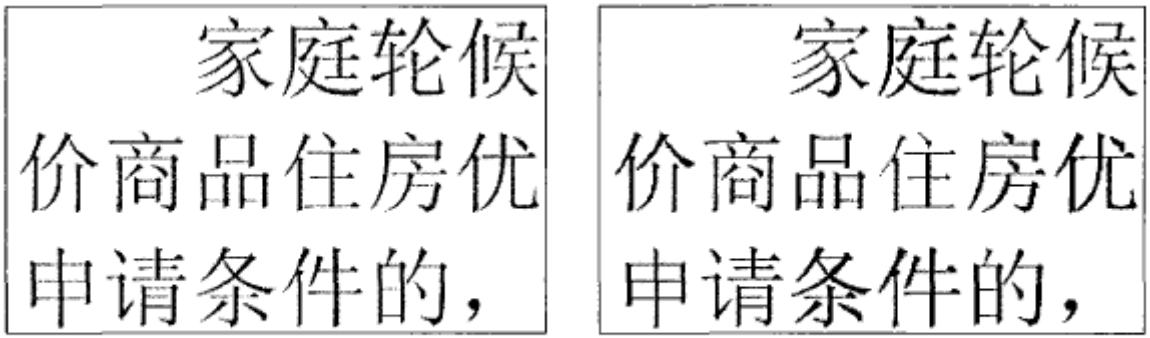


图 5-2 字符翻转前后对比：(a)翻转前；(b)翻转后

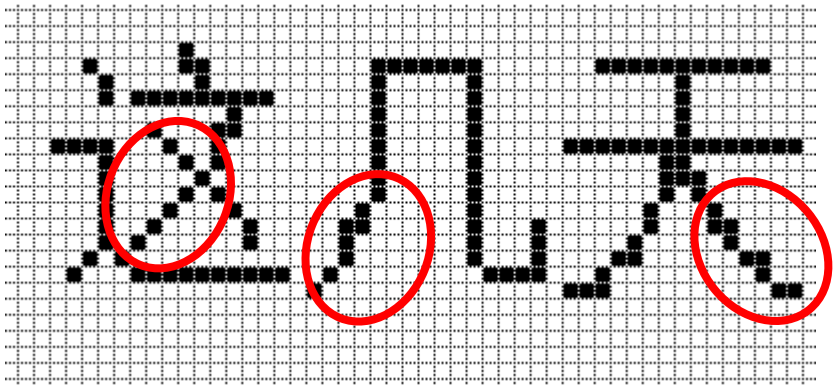
摘自参考文献1

像素翻转隐藏

- 算法鲁棒性
  - 亓氏算法误码性能与打印扫描器材有关，改进思路包括：
    - 根据复杂度筛选字符
    - 增加纠错编码等。

像素翻转隐藏
——阶梯边沿调整算法

- 原理
  - 翻转像素，使黑色像素数量满足一定关系。
  - 判断像素是否可以翻转的依据，不是连通性和平滑度。
  - 算法只翻转阶梯边沿像素。



像素翻转隐藏
——阶梯边沿调整算法

- 原始图像

抗打印扫描的水印算法的设计是本系统设计中的一个难点。现有的水印算法集中在对颜色纹理丰富的图像上，对颜色、纹理极其简单的打印文档的水印研究较少。打印扫描过程中引入的噪声和模糊等失真对水印算法的设计也带来了更高的要求。本系统提出的水印算法的创新点如下：

- 边沿调整

抗打印扫描的水印算法的设计是本系统设计中的一个难点。现有的水印算法集中在对颜色纹理丰富的图像上，对颜色、纹理极其简单的打印文档的水印研究较少。打印扫描过程中引入的噪声和模糊等失真对水印算法的设计也带来了更高的要求。本系统提出的水印算法的创新点如下：

- 平滑滤波

抗打印扫描的水印算法的设计是本系统设计中的一个难点。现有的水印算法集中在对颜色纹理丰富的图像上，对颜色、纹理极其简单的打印文档的水印研究较少。打印扫描过程中引入的噪声和模糊等失真对水印算法的设计也带来了更高的要求。本系统提出的水印算法的创新点如下：

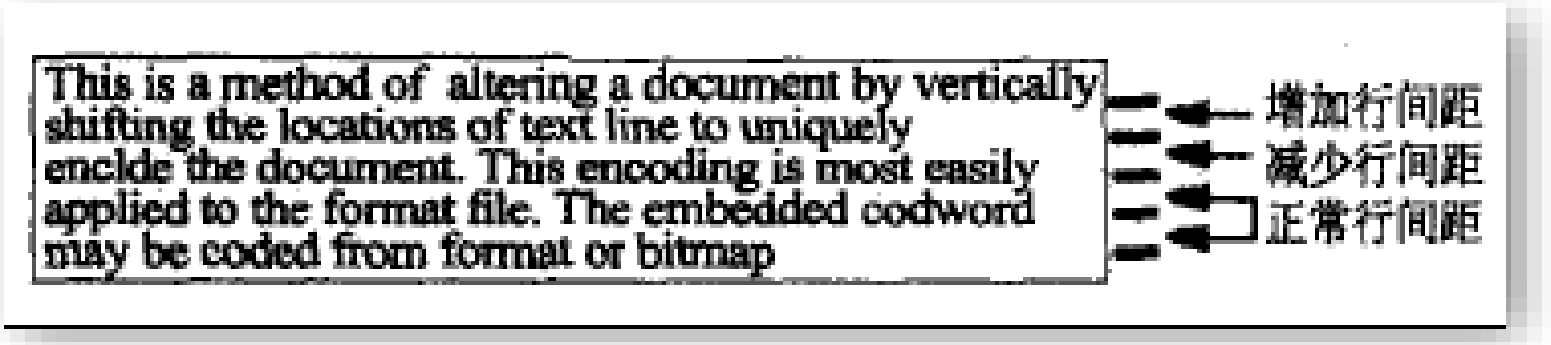
像素翻转隐藏

- 小结
  - 像素翻转法，稳健性根据参数各有不同，可用于纯数字通道传输的文档图，也可用于打印扫描的应用场景。

格式特征隐藏

- 利用视觉冗余，以一定的方式对文档的各元素特征（如字形、字体、位置、下划线、笔划高度和方向等）进行修改，而不引起阅读者的怀疑
- 通常用于在文档图像中的隐藏
  - 在文档图像的字、行、段等位置做少量修改。对行进行上下偏移，字词左右偏移代表01

格式特征隐藏——行间距编码



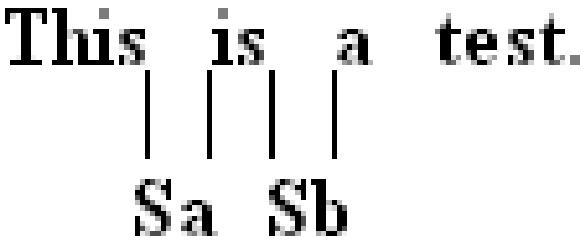
- 利用文本的行间距携带水印信息
  - 根据经验，当垂直位移量等于或小于1/300英寸时，人眼将无法辨认
  - 需嵌入信息的行进行轻微的上移和下移.

格式特征隐藏——字间距编码

- 将文本某行中的一个单词水平左移或右移来嵌入水印信息
- 相邻的单词并不移动，作为解码过程中的位置参考
- 人眼无法辨认1/150 英寸以内的单词的水平位移量

格式特征隐藏——字间距编码

- 正常情况
  - $\Delta = Sa - Sb = 0$
- 隐写
  - 嵌入0
  - $\Delta = Sa - Sb = \varepsilon$
  - 嵌入1
  - $\Delta = Sa - Sb = -\varepsilon$



格式特征隐藏——字间距编码

- 例：

a)

Now is the time for all men/women to ...

Now is the time for all men/women to ...

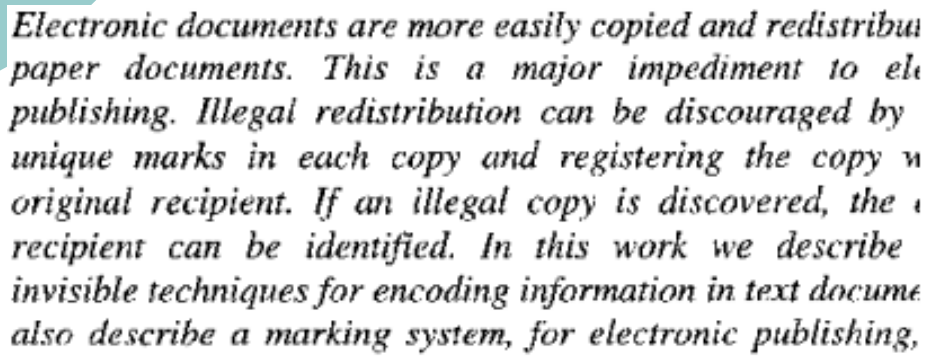
b)

Now is the time for all men/women to ...

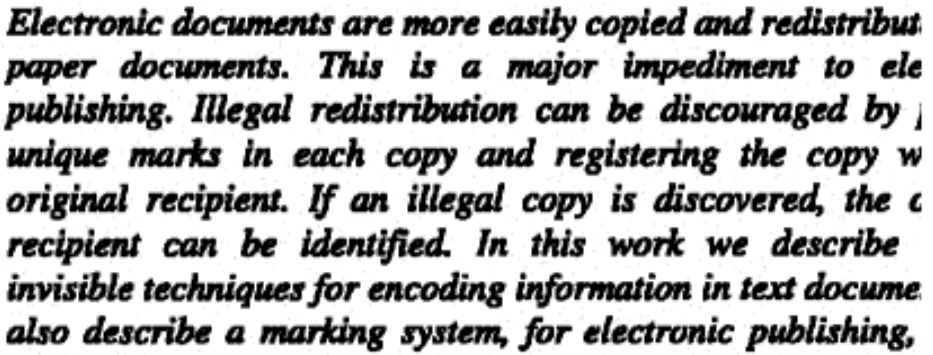
Now is the time for all men/women to ...

格式特征隐藏——字间距编码

原始图像



8次复印后图像

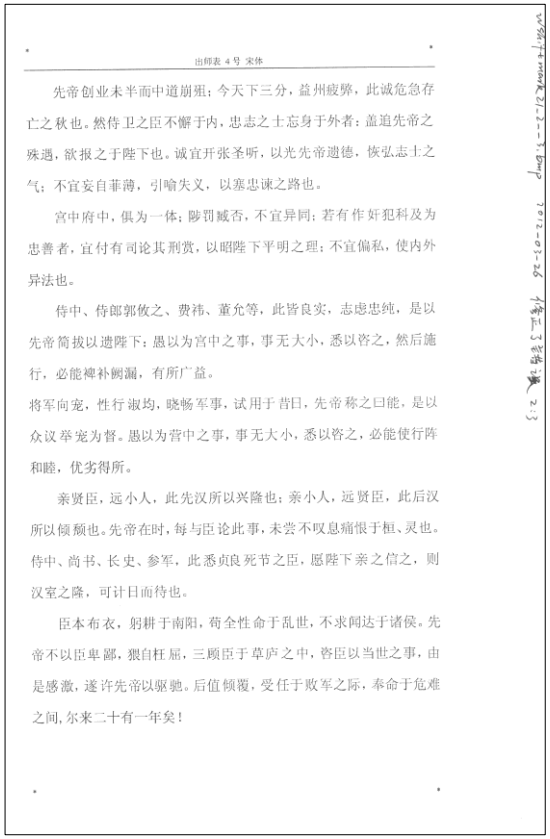


格式特征隐藏——字间距编码

- 打印追踪实例分析
  - 第一步：使用者打印文档。
    - 系统截获打印任务，把文档转换为图像，嵌入水印，再输出到打印机。这些过程用户无感知。
  - 第二步：管理者扫描或拍照需检测的文档
  - 第三步：图像预处理
    - 阈值化、矫正几何失真、去噪
  - 第四步：提取水印，识别文档输出者

格式特征隐藏——字间距编码

扫描文档图像



出师表 4号 宋体

先帝创业未半而中道崩殂；今天下三分，益州疲弊，此诚危急存亡之秋也。然侍卫之臣不懈于内，忠志之士忘身于外者；盖追先帝之殊遇，欲报之于陛下也。诚宜开张圣听，以光先帝遗德，恢弘志士之气；不宜妄自菲薄，引喻失义，以塞忠谏之路也。

宫中府中，俱为一体；陟罚臧否，不宜异同；若有作奸犯科及为忠善者，宜付有司论其刑赏，以昭陛下平明之理；不宜偏私，使内外异法也。

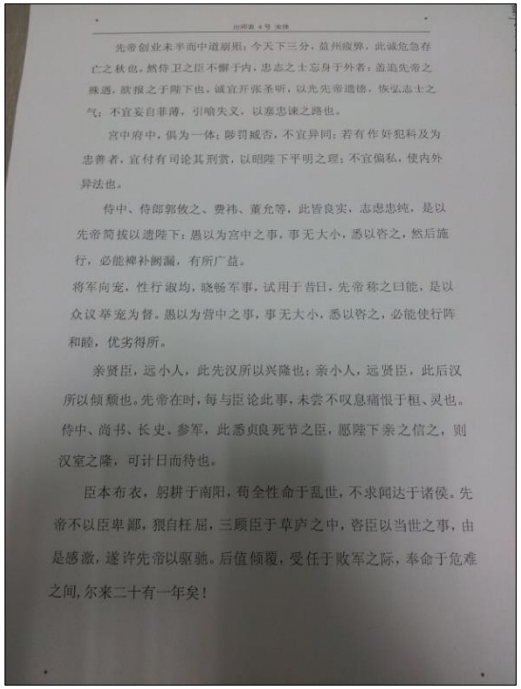
侍中、侍郎郭攸之、费祎、董允等，此皆良实，志虑忠纯，是以先帝简拔以遗陛下：愚以为宫中之事，事无大小，悉以咨之，然后施行，必能裨补阙漏，有所广益。

将军向宠，性行淑均，晓畅军事，试用于昔日，先帝称之曰能，是以众议举宠为督。愚以为营中之事，事无大小，悉以咨之，必能使行阵和睦，优劣得所。

亲贤臣，远小人，此先汉所以兴隆也；亲小人，远贤臣，此后汉所以倾颓也。先帝在时，每与臣论此事，未尝不叹息痛恨于桓、灵也。侍中、尚书、长史、参军，此悉贞良死节之臣，愿陛下亲之信之，则汉室之隆，可计日而待也。

臣本布衣，躬耕于南阳，苟全性命于乱世，不求闻达于诸侯。先帝不以臣卑鄙，猥自枉屈，三顾臣于草庐之中，咨臣以当世之事，由是感激，遂许先帝以驱驰。后值倾覆，受任于败军之际，奉命于危难之间，尔来二十有一年矣！

文档照片



出师表 4号 宋体

先帝创业未半而中道崩殂；今天下三分，益州疲弊，此诚危急存亡之秋也。然侍卫之臣不懈于内，忠志之士忘身于外者；盖追先帝之殊遇，欲报之于陛下也。诚宜开张圣听，以光先帝遗德，恢弘志士之气；不宜妄自菲薄，引喻失义，以塞忠谏之路也。

宫中府中，俱为一体；陟罚臧否，不宜异同；若有作奸犯科及为忠善者，宜付有司论其刑赏，以昭陛下平明之理；不宜偏私，使内外异法也。

侍中、侍郎郭攸之、费祎、董允等，此皆良实，志虑忠纯，是以先帝简拔以遗陛下：愚以为宫中之事，事无大小，悉以咨之，然后施行，必能裨补阙漏，有所广益。

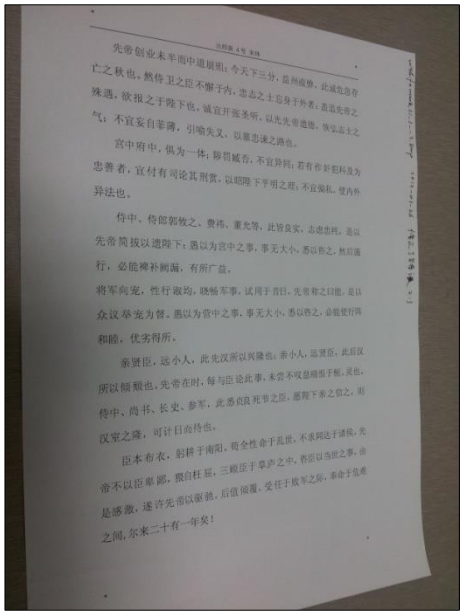
将军向宠，性行淑均，晓畅军事，试用于昔日，先帝称之曰能，是以众议举宠为督。愚以为营中之事，事无大小，悉以咨之，必能使行阵和睦，优劣得所。

亲贤臣，远小人，此先汉所以兴隆也；亲小人，远贤臣，此后汉所以倾颓也。先帝在时，每与臣论此事，未尝不叹息痛恨于桓、灵也。侍中、尚书、长史、参军，此悉贞良死节之臣，愿陛下亲之信之，则汉室之隆，可计日而待也。

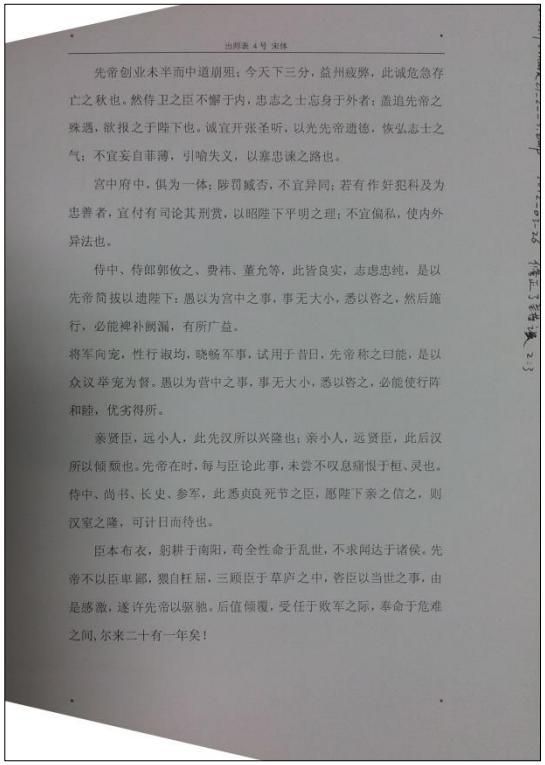
臣本布衣，躬耕于南阳，苟全性命于乱世，不求闻达于诸侯。先帝不以臣卑鄙，猥自枉屈，三顾臣于草庐之中，咨臣以当世之事，由是感激，遂许先帝以驱驰。后值倾覆，受任于败军之际，奉命于危难之间，尔来二十有一年矣！

格式特征隐藏——字间距编码

透视矫正之前的图片

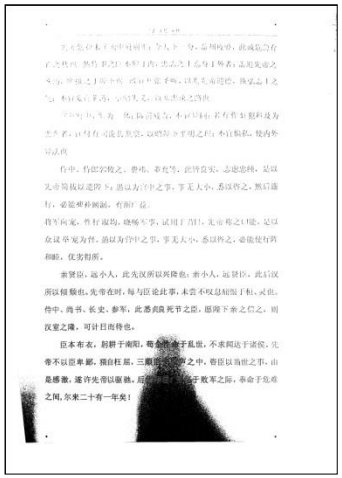


透视矫正之后的图片

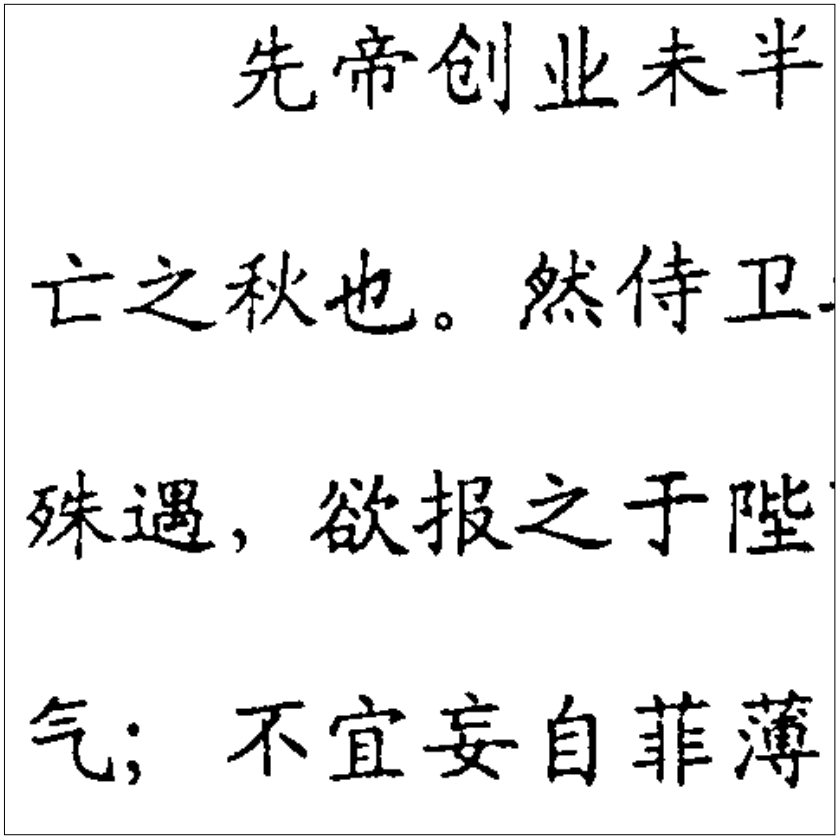


格式特征隐藏——字间距编码

固定阈值分割结果



动态阈值分割结果（局部）



格式特征隐藏——字间距编码

- 原始文档

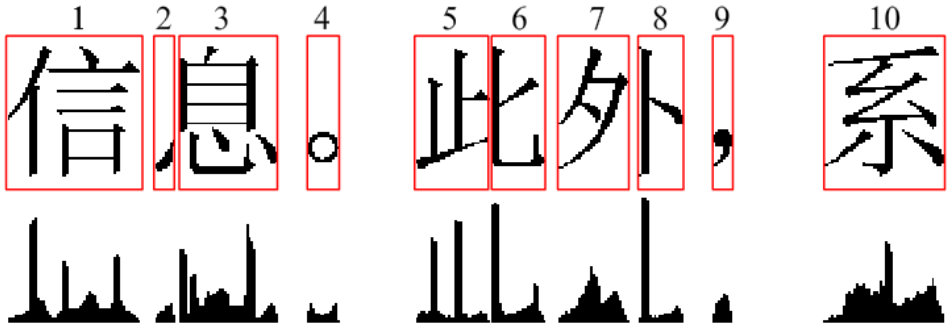
基于抗打印扫描水印的文档泄密追踪系统

- 水印文档

基于抗打印扫描水印的文档泄密追踪系统

格式特征隐藏——字间距编码

- 关键问题
  - 字符的拆分与合并



  - 可通过文档图的垂直投影来划分文字区块，需要识别标点，左右结构文字...

格式特征隐藏——字间距编码

- 关键问题
  - 字符的拆分与合并

结合行平均字间距调整合并

结合行平均字宽自动合并

  - 可结合行平均字宽等进行合并，如上图所示
  - 但中英文混排，或左右结构居多时，合并失效

格式特征隐藏——字符拉升算法

由于纸质文档的原始性、真实性等优点，在网络技术发达的今天，文档的打印仍然是各个组织单位的业务流程中不可或缺的环节，而现有的内网安全措施主要是对数字媒体的监管，对纸质文档的监管近于空白。打印的监管及涉密文档的泄密追踪已经成为内网安全中最薄弱、最难监管的环节。

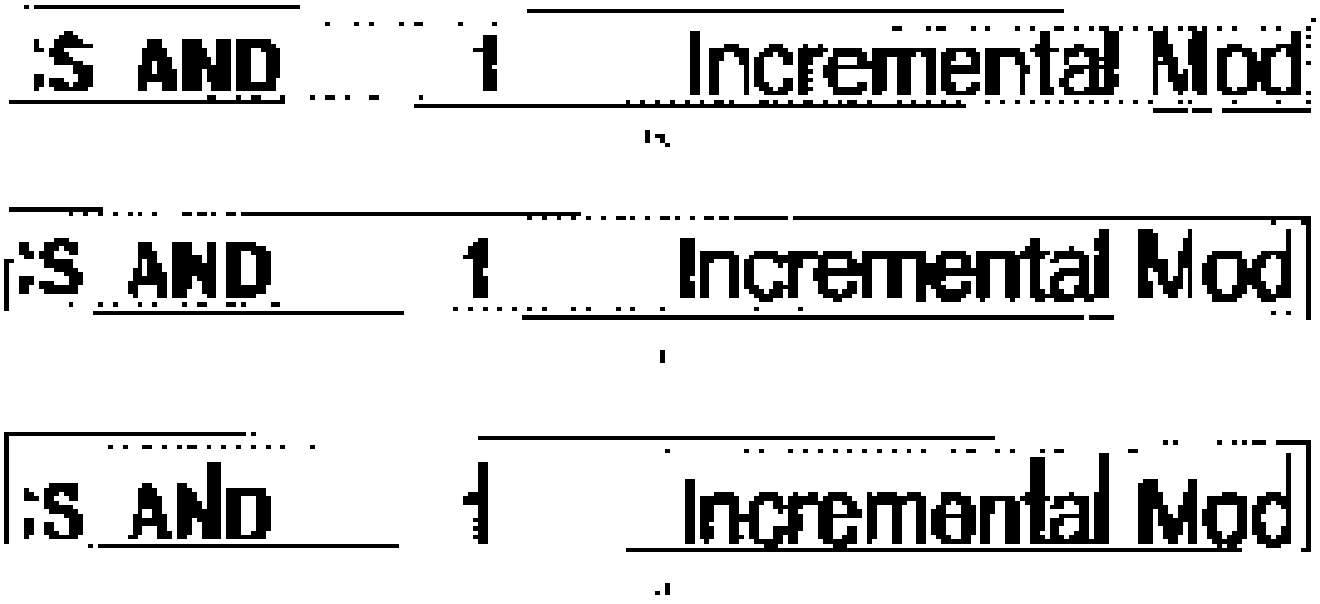
基于抗打印扫描水印的文档泄密追踪系统通过改写和替换打印处理器对打印文档实施拦截，在打印文档中嵌入抗打印扫描的身份水印信息。此外，系统提供的水印提取模块可以在发现涉密文档后进行身份信息的盲提取，为文档的泄密追踪提供依据。

文档泄密追踪系统提供了两种抗打印扫描的水印算法：基于字间距的水印算法和基于字体拉伸的水印算法，算法容量大、稳健性好。通过文字薄弱区域加强、噪声去除与边缘补偿、倾斜检测与自恢复、有效字符判定等精巧的预处理，水印算法在保证水印稳健性的前提下，最大限度了保持了打印文档的视觉效果和图像质量。此外，基于打印处理器的打印拦截独立于上层软件多样化的打印路径，有较强的监管力度。系统还提供了安装程序和简单友好的图形界面，系统部署容易，透明性好。

通过打印处理器的文档拦截、抗打印扫描的身份水印序列的嵌入，文档泄密追踪系统能够实现文档打印的监管和内部文档的涉密追踪，形成有效的威慑力，解决打印文档的监管问题，提高内网的安全性。

格式特征隐藏——调整字符高度

- 针对英文的特征编码法：修改字符的高度



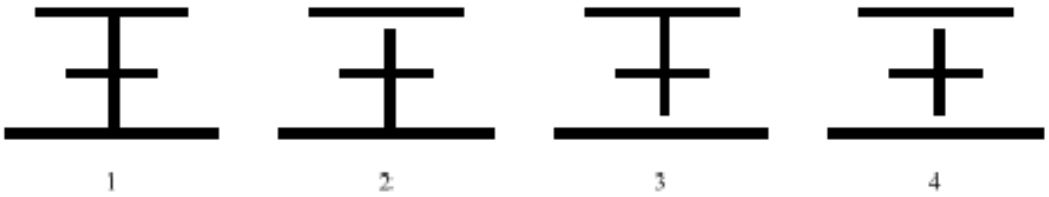
格式特征隐藏——调整字符高度

- 用于汉字的特征编码方法：基于汉字笔划的水印技术

(a) 修改前　　(b) 修改后

格式特征隐藏——调整拓扑信息

- 修改文本的拓扑结构携带信息



TAE CAT

fi fi

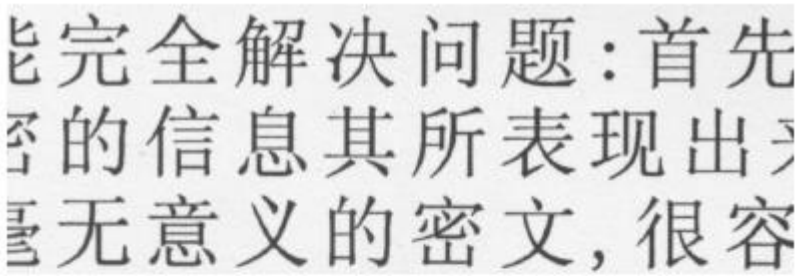
变换域算法

通过修改DWT细节系数正负符号个数的关系完成嵌入。

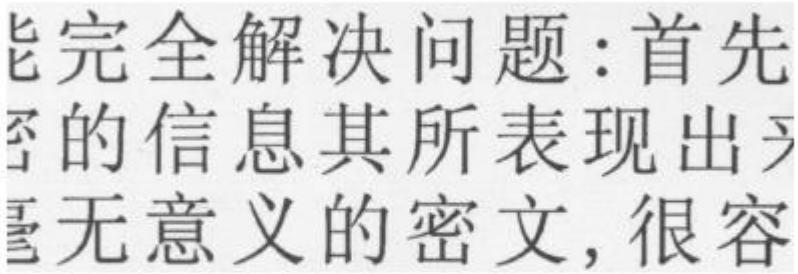
a) 原始图像

b) 含水印图像

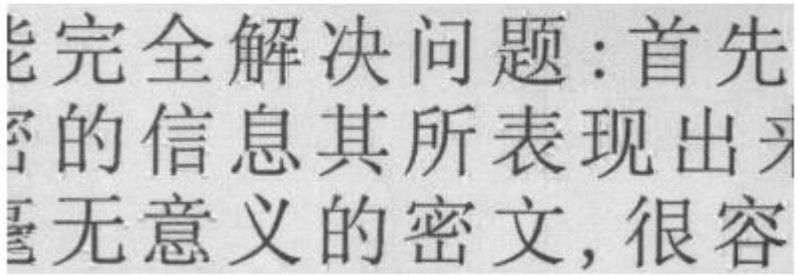
c) 打印扫描后的图像



a) 原始图像



b) 含水印图像



c) 打印扫描后的图像

显示特征隐藏

- 针对相关编辑显示软件的特点，利用可编辑、但无法屏幕显示的字节，将数据嵌入文本文档中，而文档的显示内容不变
  - 例如：在文件头、尾嵌入数据
  - 文件格式的隐藏

显示特征隐藏——演示

- 文本隐藏软件
  - snow
  - snow –f msg.txt in.txt out.txt（将msg.txt内容隐藏如in.txt中得到out.txt）
  - snow out.txt（从out.txt中提取信息）

```
000000a0h: 65 6E 20 6D 65 73 73 61 67 65 73 2E 20 0D 0A 0D ; en messages. ...
000000b0h: 0A 54 61 62 73 20 61 6E 64 20 73 70 61 63 65 73 ; .Tabs and spaces
000000c0h: 20 61 72 65 20 69 6E 76 69 73 69 62 6C 65 20 74 ;  are invisible t
```

自然文本（十六进制显示）

```
000000a0h: 65 6E 20 6D 65 73 73 61 67 65 73 2E 0D 0A 09 20 ; en messages....
000000b0h: 20 09 20 20 09 09 20 20 20 09 20 20 09 20 20 20 ;  . ..   . . .
000000c0h: 20 20 09 20 20 20 20 20 09 20 09 20 20 20 20 20 ;   .     . . .
```

携密文本（十六进制显示）

```
snow is a program for concealing messages in text files by appending tabs and spaces on the end of lines, and for extracting messages from files containing hidden
messages.

Tabs and spaces are invisible to most text viewers, hence the steganographic nature of this encoding scheme.

Examples
```

自然文本

```
snow is a program for concealing messages in text files by appending tabs and spaces on the end of lines, and for extracting messages from files containing hidden
messages.

Tabs and spaces are invisible to most text viewers, hence the steganographic nature of this encoding scheme.

Examples
```

携密文本

显示特征隐藏——在HTML文件中隐藏信息

- HTML文件是文本文件，在浏览器端仅能显示ASCII码中的可见字符。利用这一特点，可以在HTML的标记之间插入隐藏的数据
- 比如，如果要隐藏的二进制比特值为1，在选定的HTML标记后插入ASCII码值为9的字符；如果要隐藏的二进制比特值为0，则在选定的HTML标记后插入ASCII码值为32的字符

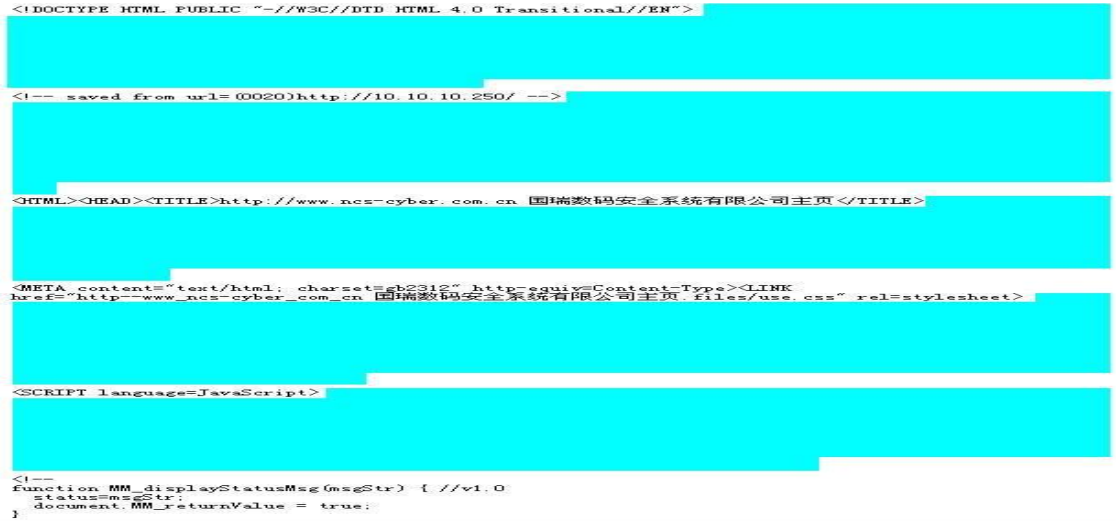
显示特征隐藏——例

- 隐藏信息前的HTML文件的部分文本

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!-- saved from url=(0020)http://10.10.10.250/ -->
<HTML><HEAD><TITLE>http://www.ncs-cyber.com.cn 国瑞数码安全系统有限公司主页</TITLE>
<META content="text/html; charset=gb2312" http-equiv=Content-Type><LINK
href="http--www_ncs-cyber_com_cn 国瑞数码安全系统有限公司主页.files/use.css" rel=stylesheet>
<SCRIPT language=JavaScript>
<!--
function MM_displayStatusMsg(msgStr) { //v1.0
  status=msgStr;
  document.MM_returnValue = true;
}
```

显示特征隐藏——例

- 隐藏信息后的HTML文件的相应文本

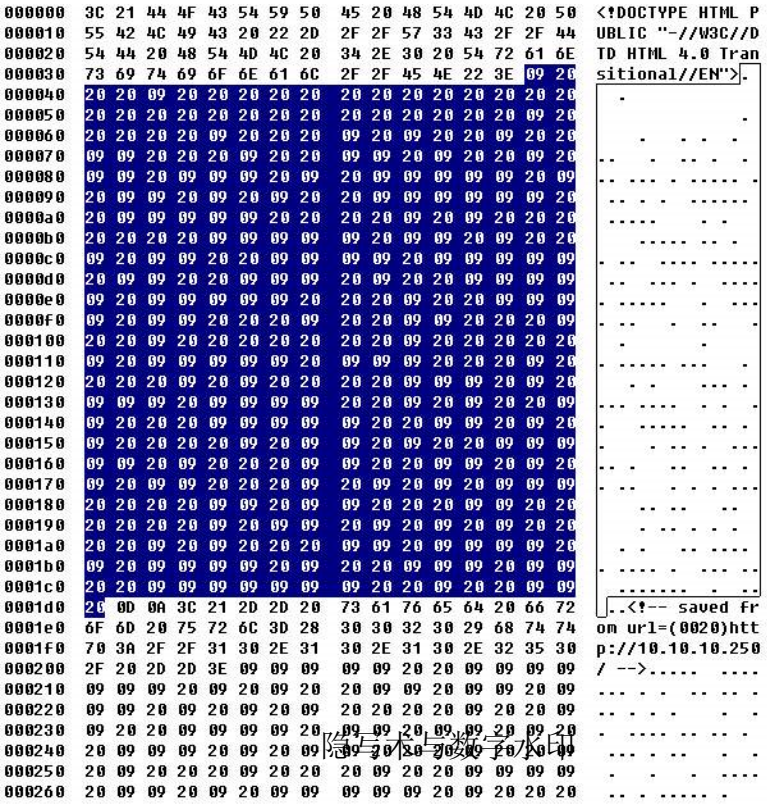


```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<!-- saved from url=(0020)http://10.10.10.250/ -->

<HTML><HEAD><TITLE>http://www.ncs-cyber.com.cn 国瑞数码安全系统有限公司主页</TITLE>

<META content="text/html; charset=gb2312" http-equiv=Content-Type><LINK
href="http--www_ncs-cyber_com_cn 国瑞数码安全系统有限公司主页_files/use.css" rel=stylesheet>

<SCRIPT language=JavaScript>

<!--
function MM_displayStatusMsg(msgStr) { //v1.0
  status=msgStr;
  document.MM_returnValue = true;
}
```

- 深颜色部分为隐藏的数据。尽管在代码文本上可以明显的看出二者之间的差异，但在浏览器端则显示不出任何差异

显示特征隐藏——例

- 在HTML文件中隐藏了秘密数据的载体文件的部分数据



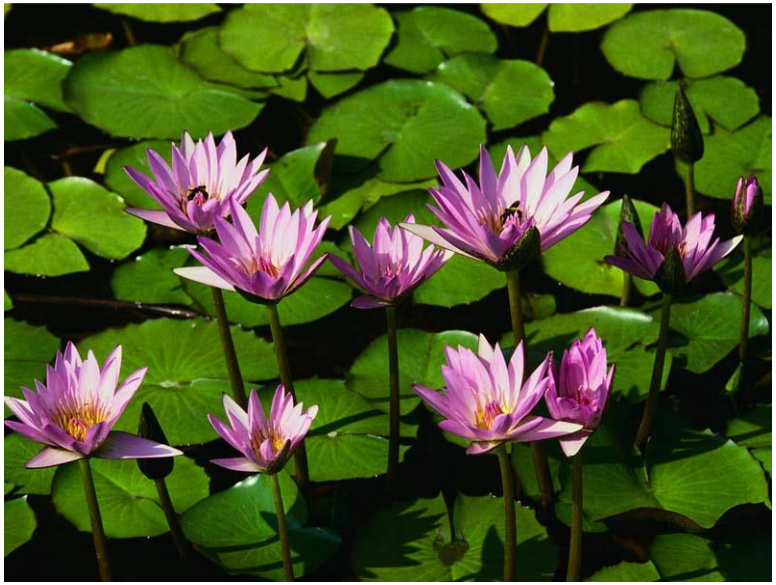
显示特征隐藏——例

（a）隐藏数据之前的浏览器显示

（b）隐藏数据之后的浏览器显示

显示特征隐藏——例

- HTML文件中隐藏的秘密数据



显示特征隐藏——在HTML文件中隐藏信息

- **回车控制符隐藏法**
  - 根据HTML语法，HTML元素或其他内容可书写为多行文本，此时，浏览器一般忽略其中的回车符。因此，可以将秘密信息中的比特“0”或“1”编码为0x0A和0x0D，插入到HTML文件，实现隐藏。
- **语法冗余隐藏法**
  - HTML语法相对宽松，存在大量冗余。例如，标记字符不区分大小写，则可根据秘密信息控制HTML文件中标记的字符选择，比特“0”则使用小写，比特“1”则使用大写。除了字符大小写外，还可以利用的冗余包括，标记是否成对使用，文件名或路径名是否使用双引号等等。

显示特征隐藏——在HTML文件中隐藏信息

属性和参数隐藏法

- HTML允许为元素增加属性和参数说明，多个参数或属性项**说明顺序不受限定**，可根据秘密信息控制HTML文件中属性和参数的顺序，比特“0”则调整属性按升序排列，比特“1”则反之。属性的顺序与网页的呈现效果无关，因此隐藏的信息不会被暴露或显示。

显示特征隐藏——在XML和OFFICE文件中隐藏信息

空格法

XML规范运行XML文件元素之间存在空格，空格可以是0X20或0X09。空格的存在不影响XML的正常解析和使用，因此，把秘密信息中的比特“0”和“1”编码为0X20和0X09，**再插入到文档元素间或文档尾部**，就可以实现信息的隐藏。

引号法

- XML规定，所有属性值必须加引号，引号本身则既可以是单引号，也可以是双引号。由于单双引号均符合语法规范，可以把秘密信息中的比特“0”和“1”编码为单引号和双引号，再根据秘密信息，替换不同属性值的引号，使两者匹配，从而实现信息的隐藏。

显示特征隐藏——在XML和OFFICE文件中隐藏信息

僵尸元素法

XML文档允许定义不同的元素，在大量的元素中，增加少量元素往往不被发现。同时，增加的元素与一个XML文件自身固有的元素不存在冲突，应用仅会检索操作预设的元素，所以新增的元素不会被使用，故而称为“僵尸元素”

- **僵尸属性法。**
  - 在载体XML文件中插入额外的属性和属性值对。由于额外插入信息不会被应用使用，所以这些僵尸属性也就不会干扰到一个XML文件的既有用途。另一方面，由于XML中已然存在大量属性和属性值对，所以插入属性值对比插入元素更容易被忽略，安全性更高。

显示特征隐藏——在XML和OFFICE文件中隐藏信息

OFFICE文档格式解析

Word 2007之后的版本按XML文件组织内容后保存在ZIP压缩文件中。下图显示典型WinWord文件解压缩内容。

显示特征隐藏——在XML和OFFICE文件中隐藏信息

OFFICE文档XML隐藏法

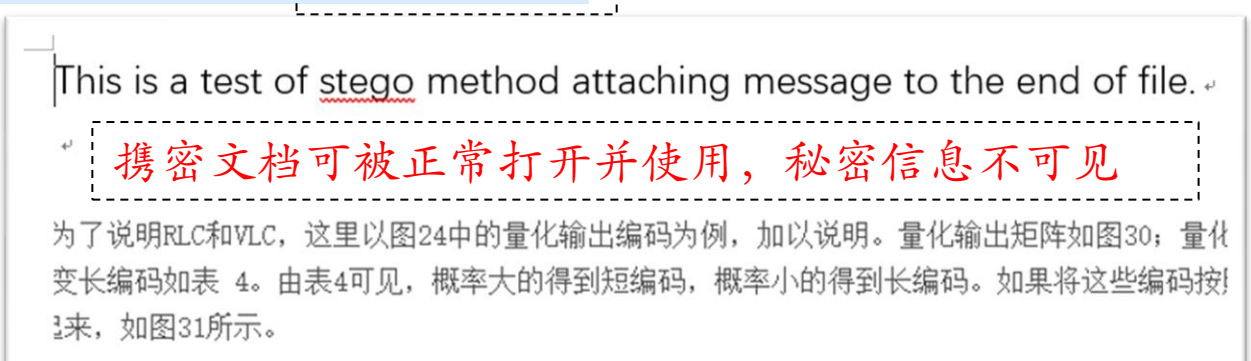
OFFICE解压缩文件中，document.xml文档是主XML文档，包含OFFICE文件主要内容。适用于XML文档的隐藏技术，均可运用于OFFICE文档。

```
><w:docGrid w:type="lines" w:linePitch="
312" w:custom="12345678"/></w:sectPr></w:
body></w:document>
```

僵尸属性法

```
><w:docGrid w:type="lines" w:linePitch="
312"/><w:newDefs w:secmesg="ABCDEFGH"/></
w:sectPr></w:body></w:document>
```

僵尸元素法



基于汉字数学表达式的零水印

- 汉字数学表达式
  - 研究汉字结构特征后，研究者们指出：汉字包括两大要素：
    - 汉字的基本组件
    - 基本组件之间的位置关系
  - 把汉字基本组件视为操作数，基本组件之间的关系视为操作符，则汉字可用数学表达式描述

基于汉字数学表达式的零水印

部份汉字基本组件及其编号　　**汉字基本组件关系**

图 4-16 部分部件及其编号[76]

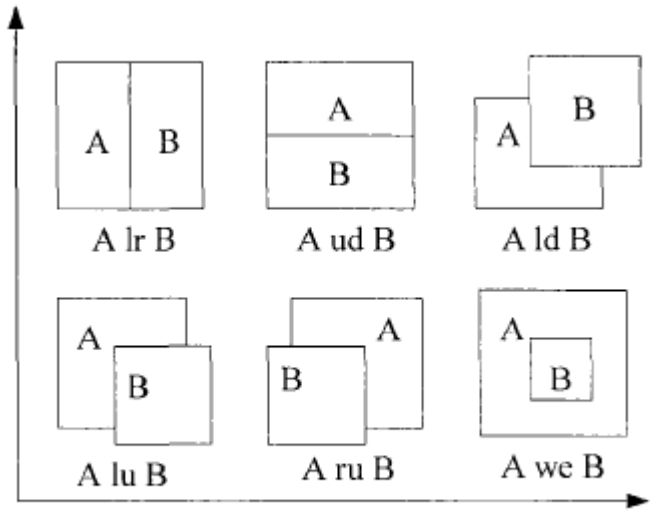


图 4-17 汉字数学表达式运算符号与部件结构的关系[76]

汉字基本组件件及其关系摘自参考文献[1]

基于汉字数学表达式的零水印

- 汉字数学表达式
  - 运用上述思想，用数学表达式描述汉字，如下例所示：

表 4-7 汉字及对应的汉字数学表达式

汉字	决	定	阶
数学表达式	418 lr 457	330 ud 369	422 lr (9 ud 299)

汉字数学表达式摘自参考文献[1]

基于汉字数学表达式的零水印

- 汉字数学表达式的应用
  - 汉字数学表达式，实质上提供了一种简单的汉字结构获取方法。
  - 有研究者使用图像处理技术完成了汉字数学表达式的自动生成，极大地方便了汉字的存储、传播和后续利用，其他应用包括自动生成汉字字形、互联网上跨平台传播汉字信息、挖掘汉字结构知识等。
  - 研究者也利用汉字数学表达式隐藏信息。例如：水印信息关联上下结构是否拆分，拆分时，利用视觉冗余，不会引起视觉失真。

基于汉字数学表达式的零水印

- 基于汉字数学表达式的零水印方案

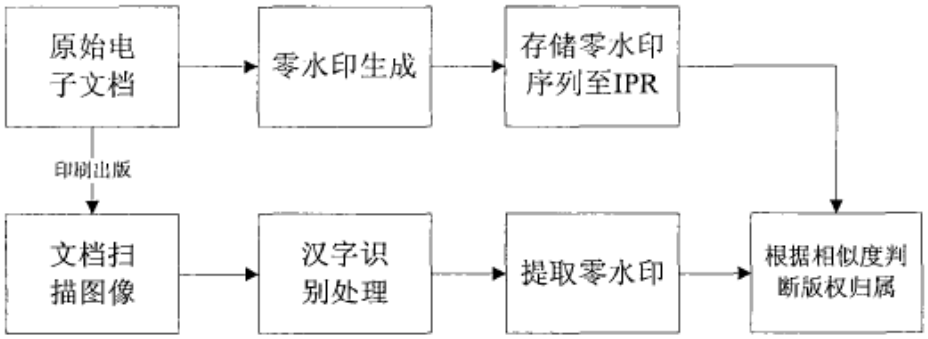


图 4-18 零水印版权认证框架

基于汉字数学表达式的零水印

- 基于汉字数学表达式的零水印方案
  - 零水印1：所选汉字的数学表达式的哈希值
  - 零水印2：为了减少误识别字形结构相似的汉字，将文档中汉字部件关系（操作符）出现的频次，作为输入产生哈希，作为零水印2。
  - 验证版权时，对于提取的零水印1与注册的零水印1，如果两者相似度为1，则待验证作品的版权得到认定。
  - 如果相似度小于1，但大于阈值，则比较零水印2。若从待检测作品中提取的，与注册的水印一致，则作品版权得到认定，否则，认为待检测作品未注册版权。

研究方向

- 提高隐藏容量
- 提高鲁棒性
- 约束多
- 难度大

参考文献

1. 抗打印扫描数字水印技术及其应用研究，郭承青（导师：胡正名）