基于文本的信息隐藏算法①

An Algorithm of Text Steganography

白 剑 杨 榆 徐迎晖 钮心忻 杨义先 （北京邮电大学信息安全中心 100876）

摘要：本文提出了一种新的基于文本的信息隐藏算法，算法利用人类的视觉系统(HVS)对标点符号和上下文之间以及字与字之间的间隔并不敏感这一特点进行信息隐藏。大量的实验结果表明这一算法具有很好的隐藏效果。每一个字可以隐藏一比特的密文信息。算法利用纠错交织分帧技术获得一定的抗增、删攻击能力。

关键词：文本 信息隐藏 文本隐藏

1 引言

信息隐藏技术是近几年来信息安全领域出现的一种新技术。所谓的信息隐藏(或者更严格的称为信息伪装)是将敏感信息秘密地隐藏于另一非机密的载体内容之中。载体形式可为任何一种媒体，如图像、声音、视频或一般的文档等。在这一领域中，研究的是如何对机密信息增加一层伪装，使得机密信息的传输不会引起注意，从而实现隐蔽通信。从目前发表的大量研究论文看，研究得最多和最深入的是在图像载体中隐藏信息和嵌入数字水印，这一方面是由于图像处理的直观性，另一方面是由于图像中存在大量的冗余信息，由于这些冗余信息的存在，使得我们可以在其中隐藏一些信息而不致引起观察者的怀疑。但是对于文本信号就不同了，文本信号中不存在冗余，文本的一个比特发生变换，文本就发生错误，因此在文本中进行信息伪装的难度比较大。

文献[1]中总结了在目前发表的论文中，关于文本伪装的一些方法，它们主要是利用文本字符的字间距、行间距、标点符号等位置隐藏几个比特的信息。另外文献[1]中还提到了另外一种方法，称为自由上下文语法，它是用一些常用主、谓、宾单词根据要隐藏比特进行组合，组合出一些具有正常含义的句子。文献[2]中提出将不具有冗余度的文本信号经过变换后，得到具有冗余度的信号，再在冗余空间中进行文本的伪装。文献[3,6]中提出了利用标点信息隐藏的算法、利用字体信息隐藏的算法。

上述各种基于文本的信息隐藏方法具有很好的隐蔽性。但是在实际的使用中，上述方法往往需要原始文档或者大容量密钥作为解隐藏的参考值。而利用排版格式进行信息隐藏的方法，如利用行间距、字间距以及字体进行信息隐藏，在纯文本编辑器中无法使用。由于 ICQ、QQ、MSN 等纯文本传输工具在网络上被大量使用，如果能够利用纯文本编辑、传输软件作为秘密信息传输途径，将会显著提高其隐蔽性。此外，由于某些应用场合的条件限制(如用户使用公用机器上网进行秘密信息传输)，要求隐藏和解隐藏的算法简单易用，并且要求算法能够抵御一定的破坏性攻击(如文字的删剪和增加，重新排版等)。因此实际使用中需要一种隐藏容量大，实现简单，算法运算量小而且能够抵御一定攻击的算法。

在我们的研究工作中，发现人类的视觉系统(HVS)对标点符号和上下文之间的间隔、字与字之间的间隔并不敏感。因此我们设计出一种新的基于文本的信息隐藏算法，它的基本出发点是对密文信息经过纠错交织编码，然后分帧打包。利用标点符号的左右是否出现空格来代表密文的帧头信息，以字与字之间不同间隔来代表密文 01，以达到隐藏密文信息的目的。大量的实验结果表明这一算法具有很好的隐藏效果。每一个字可以隐藏一个比特的密文信息，具有较好的隐藏容量。同时具有一定的抵御密文解码出错，明文词句增加、删减攻击的能力。

① 本课题得到国家重点基础研究发展规划项目(TG1999035804)、北京市自然科学基金项目、教育部优秀青年教师资助计划项目的资助

2　隐藏算法

在一篇文档中，标点符号和文字之间是否有空格是不容易引起人们的注意。比如下面这段文字：Hello，This is ISC of BUPT . Welcome you to BUPT . Thank you！在上面这段英文中存在如下几种空格排列方案：

（1）标点符号和文字之间没有空格，

（2）标点符号和前面文字有一个空格，

（3）标点符号和前后文字之间都有空格。但是从视觉效果来看，并不能特别的引起检测者的关注。同样，字与字之间的空格不同也不容易引起检测者的注意。这就为我们进行信息隐藏提供了一种途径。本文的算法正是利用这一特点进行隐藏的。

为了抵御攻击，先对密文信息进行纠错交织编码然后分帧打包处理。纠错编码采用的是 GRAY 码（23，12，3），交织深度为 22。处理过后的密文信息排列示意图如下：

表 1　密文信息包排列图

帧头 1	密文信息（BK bit）	帧头 2	密文信息（BK bit）
帧头 3	密文信息（BK bit）	帧头 4	密文信息（BK bit）
帧头 5	密文信息（BK bit）	帧头 6	密文信息（BK bit）
帧头 7	密文信息（BK bit）	帧头 1	密文信息（BK bit）
帧头 2	密文信息（BK bit）	帧头 3	密文信息（BK bit）

在算法中，我们利用标点符号的两边不同的空格数量来代表不同类型的帧头。

共设计 7 类帧头（以下 S 代表空格、‘，’代表标点、X 代表任意字符）：

帧头 1：XXS，XXX

帧头 2：XSS，XXX

帧头 3：XXX，SXX

帧头 4：XXX，SSX

帧头 5：XXS，SXX

帧头 6：XSS，SXX

帧头 7：XSS，SSX

密文比特是利用字与字之间的空格数目来代表。如果字与字之间空格数目为 1 个，那么代表比特 0，如果空格数目为 2 个，那么代表比特 1。一帧密文为 BK（大小由收发双方自定义）比特，需要利用 BK 字隐藏。例如，需要在明文 Hello，This is ISC of BUPT . Welcome you to BUPT. Thank you！中隐藏帧头为第 2 类帧，密文信息为 010101 的数据，隐藏后的明文如下：

Hello ，This is _ISC of _ BUPT . Welcome you _to BUPT .

帧头 2 0 1 0 1 0 1

帧头的作用是标志其后的密文信息为第几个 BK，密文信息排列如图 1 所示。这样一旦出现密文增删情况，错误不会累积。下面举例来说明这个问题。

如在 BK = 23 比特的情况。接收端解隐藏中，帧头 2 后面发生信息丢失，仅仅解出 10 比特信息。紧接出现了帧头 3。此时算法由于以前知道两个帧头之间应该有 BK 比特信息，所以自动填充 13 比特 0。这样，虽然出现了密文丢失情况，但是仅仅影响第二帧数据，后面数据仍然能够正常解码。同理这种分帧打包技术也可以防止信息增加情况。而对于信息解码错误情况，可以利用纠错编码来恢复。因此算法具有一定的抵抗攻击、传输出错的能力。

其隐藏算法描述如下：

step 1：将待隐藏的密文转换为 01 比特流，同时进行纠错交织编码。

step 2：对明文信息进行格式化，主要工作为将明文信息中标点符号两边的空格删除，字与字之间空格都变为一个空格。

Step 3：初始化帧头数量计数器 Hn = 0 ，隐藏信息数量计数器 Sn = 0。

Step 4：查找下一个标点符号出现的位置。令 Sn = 0，Hn 值加一。如 Hn > 6 ，then Hn = 0。

并在该标点位置根据 Hn 值插入对应的帧头类型。例如 Hn = 3 那么插入帧头 3。

Step 5：读取待隐藏信息 s ，查找下一空格出现位置。If s = = 1 ，then 该位置插入一个空格。同时 Sn 加一。Step 6：if Sn > = BK ，then 跳转 Step 4。If Sn < BK ，then 跳转 Step 5。

Step 7：隐藏完毕，退出程序。

为了显示算法隐藏效果，下面给出一个隐藏例子，由于论文篇幅限制，这里仅给出隐藏效果的一部分。

需要隐藏的密文为：Hello，This is ISC of BUPT. 明文信息如下所示：

The largest shark is the whale shark. At over 45 feet，it's also the world's largest fish. The whale shark strains out tiny animals that drift in the sea，such as plankton，shrimp，and small fish. The smallest shark is

the spined pygmy shark. It´s less than 10 inches long and can be found in the cold, deep oceans. It's bioluminescent which means it glows in the dark. Most sharks are efficient swimmers. The highly streamlined mako has been clocked at speeds up to 22 mph. Most sharks have 5 to 15 rows of teeth in each jaw. Unlike human teeth, shark teeth don´t have roots to hold them in place, so their teeth are easily broken off. A tooth usually lasts about a week before it falls out. When this happens, the tooth behind it moves up to replace it. A new tooth can be replaced in as little as 24 hours. Sharks keep replacing their teeth all their lives. As the shark grows, its new teeth keep pace and grow larger than the ones that are replaced. Sharks are some of the oldest and most successful vertebrates on the earth. The first sharks appeared in ancient oceans 400 million years ago.

利用上述算法隐藏了密文后的明文如下所示：

The largest shark is the whale shark. At over 45 feet, it´s also the world´s largest fish. The whale shark strains out tiny animals that drift in the sea , such as plankton, shrimp, and small fish. The smallest shark is the spined pygmy shark. It´s less than 10 inches long and can be found in the cold , deep oceans. It's bioluminescent which means it glows in the dark. Most sharks are efficient swimmers. The highly streamlined mako has been clocked at speeds up to 22 mph. Most sharks have 5 to 15 rows of teeth in each jaw. Unlike human teeth, shark teeth don't have roots to hold them in place, so their teeth are easily broken off. A tooth usually lasts about a week before it falls out. When this happens, the tooth behind it moves up to replace it. A new tooth can be replaced in as little as 24 hours . Sharks keep replacing their teeth all their lives. As the shark grows, its new teeth keep pace and grow larger than the ones that are replaced . Sharks are some of the oldest and most successful vertebrates on the earth. The first sharks appeared in ancient oceans 400 million years ago.

可以看出，原始明文和隐藏信息后的密文仅仅在上下文之间的间隙呈现随机性的差别，很容易被误认为书写者的习惯问题，所以很难引起检测者的怀疑。此外，从上述示例可以看出这种算法的编码过程非常的简单，算法实现复杂度小，易于推广使用。

3 解隐藏算法

在接收方收到了隐藏了密文信息的明文以后，即可进行解码处理。其解隐藏算法如下：

step 1：打开接收到明文文件，读取明文。

step 2：初始化帧头数量计数器 Hn = 0，隐藏信息数量计数器 id = 0，帧数寄存器 Hst = 0，

帧内信息计数器 Sn = 0。

Step 3：if Sn < HBK ，then 计算每个字之间空格数目，进行译码，并将译出码放入译码缓冲区第 id 个位置，同时 Sn + + 。然后循环执行本步。If Sn > = HBK then 暂停译码，跳转 step 4。

step 4：查找下一标点出现的位置，根据标点左右空格分布译出 Hn 值。

If Hn = =1 ，then Hst + =HBK * 7；

同时将 id 设置为 Hst + Hn * HBK. 同时将 Sn 设置为 0。

Step 5：跳转 Step 3。

Step 6：滑动到明文尾位置，译码结束。

不难看出，这种算法的解隐藏非常的简单，大大降低了算法实现的复杂度，可以在不同的场合使用。同时，由于这种方法解码过程并不需要原始的文档作为标点符号位置参考，因此使用起来很方便。

4 抵抗攻击性能分析

经过研究我们发现可能会对上述算法产生影响的攻击有下面 3 种：

（1）对隐藏后的明文重新排版，调整某些字之间间距，即调整字与字之间的空格数目。

（2）删除某些句子。

（3）增加某些句子。

我们进行了大量仿真试验来测试算法抵抗以上攻击能力。试验条件首先将含有 78 个字符的密文：Hello, This is ISC of BUPT. Hello, This is ISC of BUPT. Hello, This is ISC of BUPT. ！ 经过纠错交织分帧以后隐藏到一段共有 11482（TC）个字符、2473 个空格、254 个标点符号的明文中，得到待传明文 P。

为了测试算法抵抗重新排版攻击的能力，我们建立了一个仿真明文传输信道(PTC)，该信道参数为明文空格差错率(ESR)。该信道随机改变 P 中 EN 个空格的值，得到通过信道的明文 TP。改变有两种情况，一种删除空格，一种为增加空格。具体过程如下：我们统计 P 中空格总数量(SC)，仍然随机生成一个随机序列 R，总长为 SC，每个值为[0 1 2]之一。其中 1,2 出现的概率都为 ESR/2。将 R 中每个值和 P 中的每个空格一一对应，如果 R 中某个值为 0，那么 P 中对应空格不变；如果为 1，那么在空格前增加一个空格；如果为 2，那么删除该空格。这种情况基本反映了重新排版的过程，其中 ESR 反映了重新排版对明文 P 的修改程度。然后对明文 TP 进行解隐藏算法，计算密文的恢复率。我们选择了有代表性的 6 种 ESR。对每种 ESR 下的 P 进行 100 次仿真试验，并记录实验结果。如表 2 所示：

表 2　算法仿真实验结果(明文字符总数 1148200，密文字符总数 7800，空格总数 247300)

空格差错率	空格差错数	密文出错率	密文出错字符数	无错传输次数
0.001	247	0.007	56	98
0.002	495	0.041	320	91
0.005	1237	0.07	560	86
0.01	2473	0.15	1206	51
0.015	3710	0.24	1923	23
0.02	4946	0.3	2355	9

为了测试算法抵抗增、减句子攻击的能力，我们建立了另一个仿真明文传输信道(PTC1)，该信道参数为明文句子增减率(SADR)。该信道随机在 P 中增加或者删减 SN 个句子，得到通过信道的明文 TP1。改变有两种情况，一种删除句子，一种为增加句子。具体过程如下：我们统计 P 中句子总数量(SC1)，仍然随机生成一个随机序列 R1，总长为 SC1，每个值为[0 1 2]之一。其中 1,2 出现的概率都为 ADR/2。将 R1 中每个值和 P 中的每个句子对应，如果 R1 中数值为 0，那么 P 中对应句子不变；如果为 1，那么在该句子前插入句子的拷贝；如果为 2，那么删除该句。然后对明文 TP1 进行解隐藏算法，计算密文的恢复率。我们选择了有代表性的 5 种 SADR，对每种 SADR 下的 P 进行 100 次仿真试验，并记录实验结果。实验结果如表 3 所示：

表 3　算法仿真实验结果(明文字符总数 1148200，密文字符总数 7800，句子总数 25400)

句子增减率	句子增减数	密文出错率	密文出错字符数	无错传输次数
0.0004	9	0.02	174	96
0.0008	18	0.06	509	85
0.0015	38	0.1	782	77
0.002	48	0.2	1645	61
0.007	176	0.52	4098	20

从以上实验结果可以看出，由于采用了分帧和纠错编码处理，本文提出的算法具有一定的抵抗增加、删减句子攻击能力。同时，算法也具有一定的纠随机译码错误能力。

5　结论

本文提出了一种利用文字、标点之间的空格排列规则实现信息隐藏的算法。此算法利用标点符号左右是否出现空格代表密文帧头信息，利用字与字之间的间隔代表密文信息，在视觉效果上仅仅表现为上下文之间的间隙呈现随机性的差别，不易引起检测者的怀疑。这种算法的编解码过程非常的简单，不需要原始文档作为参考，算法实现复杂度小。同时由于一个字可以隐藏一比特的信息，信息隐藏量大，便于推广使用。

参考文献

1 Stefan Katzenbeisser，Fabien A P Petitcolas. Infomation Hiding Techniques for Steganography and Digital Watermarking [M]. Artech House Publishers，2000.

2 钮心忻、杨义先，文本伪装算法研究 [J]，电子学报，2003 年，3 期。

3 曹卫兵、戴冠中、夏煜、慕德俊，基于文本的信息隐藏技术 [J]，计算机应用研究，2003 年，10 期。

4 W Bender，et al. Techniques for data hiding [J]. IBM System Journal，1996，35(3/4)：3132336.

5 贾英江、傅孝忠、于鑫，数字文档的数字水印 [J]，小型微型计算机系统，2000，21(10)：106721068。

6 汪小帆、戴跃伟、茅耀斌，信息隐藏技术方法与应用 [M]，机械工业出版社. 50－90.