一种新的基于 PE 文件的信息隐藏方案与实现*

吴振强 1,2　马建峰 1　冯绍东 2

1 西安电子科技大学，计算机网络与信息安全教育部重点实验室，西安，710071

2　陕西师范大学，计算机科学学院，西安，710062

Email: zqiangwu@snnu.edu.cn

摘要：本文从一个全新的角度来研究信息隐藏技术，首次将 PE 文件作为掩护媒体引入信息隐藏领域. 在分析 PE 文件的结构和 PE 文件实现信息隐藏原理的基础上，给出了基于 PE 文件的信息隐藏方案，实验分析表明 PE 文件完全可以作为掩护媒体.

关键词：信息隐藏，PE 文件 ，掩护媒体， 软件水印

中图分类号：TP393.08　　**文献标识码： A**

A Novel Scheme and Implementation of Information Hinding Based on PE File Format

WU Zhen-qiang 1，2　MA Jian-feng1　Feng Shao-dong2

1 The Key Lab. of the Ministry of Education for Computer Networks and Information Security, Xidian Univ., Xi'an 710071

2 School of Computer Science, Shaanxi Normal University, Xi ' an, 710062

Abstract：This paper deals with the information hiding technology of PE file format in a new field of vision，and tries to introduce the PE file format into the information hiding. On the basis of research on the PE file format and the principle of implementation information hiding in PE file, we give a scheme of information hiding based on the PE file. The experiment shows that the PE file can be a cover-media .

Key Words information hiding, PE file format, cover-media, Software Watermarking

信息隐藏(Information Hiding)是将某一签字信号(Signature Signal)嵌入(embedding)另一主信号(Host Signal，或称掩护媒体，cover-media)的过程．掩护媒体经嵌入信息后变成一个伪装媒体(stegano-media)，这一嵌入过程需要满足下列三个条件 [1]： (1)签字信号的不可感知性(Imperceptibility)，即签字信号嵌入后，主信号的感知特性没有明显的改变，签字信号被主信号隐藏起来； (2)签字信号的鲁棒性 (Robustness)，签字信号对主信号的各种失真变换，如失真信号压缩、仿射变换、加噪、 A/D 或 D/A 转换等，以及各种恶意性攻击(Malicious Attack)，都应体现出一定的鲁棒性．除非主信号的感知特性被明显的破坏，签字信号将很难被去除； (3)签字信号的嵌入不增加主信号的存储空间和传输带宽，即签字信号嵌入后，“表面”上很难觉察到信息的改变．针对这三个条件，目前较常用的掩护媒体是图像、声音、视频和文本， 文献[2，3]全面论述了信息隐藏的发展历史和研究现状，对目前在图像、声音、文本等方面的研究成果进行了总结和论述.

本文从一个全新的角度来研究 PE 文件的信息隐藏技术，在分析 PE 文件框架结构的基础上，发现了 PE 文件存在着大量的冗余空间，于是尝试将 PE 文件作为掩护媒体引入到信息隐藏领域，提出了一种新的基于 PE 文件的信息隐藏方法．本文先对 PE 文件的结构进行介绍，分析了 PE 文件实现信息隐藏的原理，然后给出了 PE 文件的信息隐藏实现方案，实验分析表明 PE 文件满足掩护媒体所要求的三个条件，即 PE 文件完全可以作为掩护媒体.

1　PE　文件的框架结构

PE(Portable Executable，可移植的执行体)文件格式是 Microsoft 制定的一种文件标准 ，它是从 UNIX 操作系统的 COFF (Common Object File Format) 文件格式发展而来 ，在

* 863 计划(2002AA143021)，国家自然科学基金重大计划(No. 90204012)、教育部优秀青年骨干教师资助计划、教育部科学技术重点研究项目.

作者简介：吴振强（1968－），男，陕西商洛人，博士生，副教授，主要研究兴趣为计算机通信网络安全；马建峰（1963－），男，陕西临潼人，博士，教授，博士生导师，主要研究领域为信息安全与密码学.

Windows 操作系统中扮演着非常重要的角色，其格式中的数据结构通常在 WINNT.H 中定义．可移植性意味着 PE 文件格式是可以跨 win32 平台，任何 win32 平台的 PE 装载器都能识别和使用该文件格式．所有 Win32 执行体 (除了 VxD 和 16 位的 Dll)都使用 PE 文件格式，包括 NT 的内核模式驱动程序(kernel mode drivers)等，并且 Win64 平台系统也兼容 PE 文件格式．

PE 文件格式中包含一些固定的字段，这些字段用来定义 PE 文件中其它部分的内容．文献[4]给出了 PE 文件的结构规范，通用的 PE 文件格式见图 1．PE 文件都是从 MS-DOS Header 头开始，程序通过此字段在 DOS 下执行时，DOS 就能识别出这是有效的执行体，并执行紧随 MS-DOS Header 之后的 MS-DOS Stub．其中 MS-DOS Stub 程序不支持 Win32 环境，若在不支持 PE 文件格式的操作系统下执行，程序将简单显示一个错误提示，如"This program cannot be run in DOS mode" ．在 WINNT.H 中的 IMAGE_NT_HEADERS 头部后面是段表 (Section Table)，段表是一个数组，数组中的每一个元素用来描述后继每一段的信息，如段名、段大小、段偏移、段属性等．在段表的后面是各个段的段体 ，如代码段 (.text)、数据段 (.data)、Import 表 (.idata)、Export 表 (.edata)、资源段 (.rsrc)等．执行体在支持 PE 文件结构的操作系统中执行时，PE 装载器将从 MS-DOS Header 中找到 PE Header 的起始偏移量，跳过 MS-DOS Stub 直接定位到真正的文件头 PE Header．PE 文件的内容划分成段(sections)，每段是一块拥有共同属性的数据．

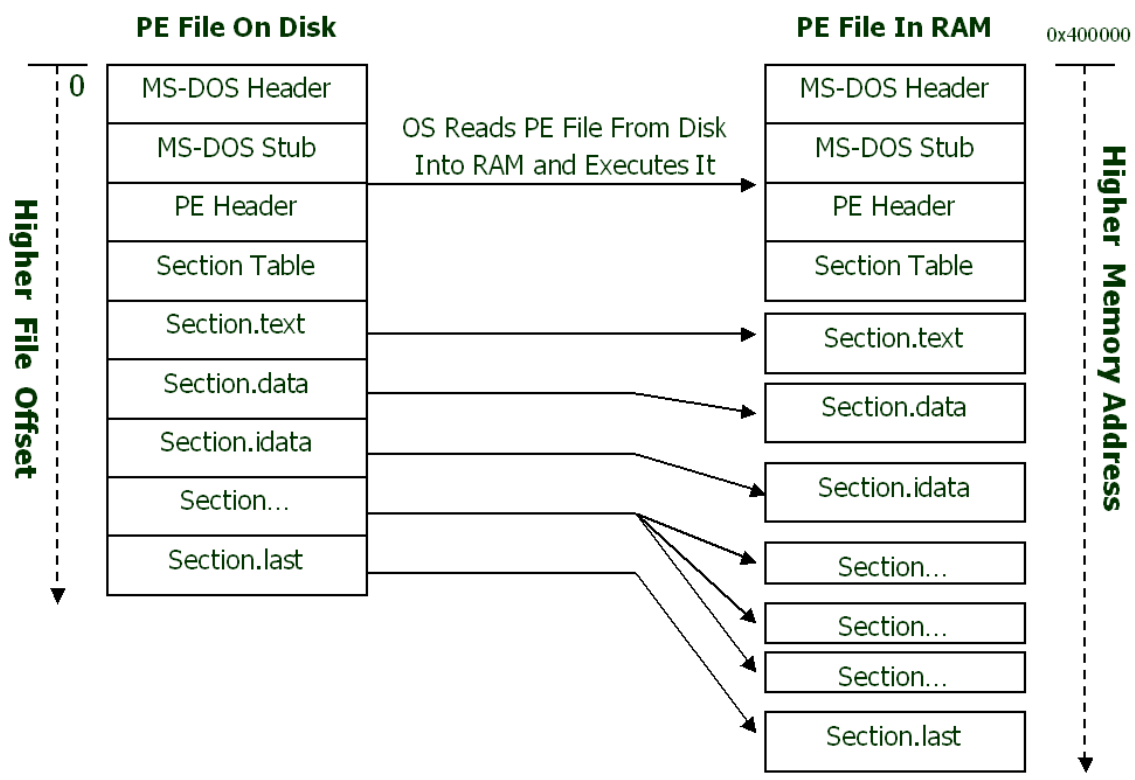

图 1　通用 PE 文件结构

Win32 系统在执行一个 PE 文件前，先由 PE 装载器将文件从磁盘装入到内存 RAM 中，通常是从内存地址 0x400000 开始存放可执行性程序. 由于装载器在装入 PE 程序过程中，有可能还要装入 DLLs 文件，因此图 1 中右侧将各种不同的数据段留下一定的空间，表示 DLLs 插入后占用的空间. PE 程序装入完成后，再由 PE 装载器对 PE 文件的结构进行检查. 结构检查过程如下：(1)当 PE 文件被执行时，PE 装载器检查 MS-DOS Header 里的 PE Header 偏移量．如果找到，则跳转到 PE Header； (2)PE 装载器检查 PE Header 的有效性．如果有效，就跳转到 PE header 的尾部； (3)紧跟 PE header 的是段表，PE 装载器读取其中的段信息，并采用文件映射方法将这些段映射到内存，同时附上段表里指定的段属性； (4)PE 文件映射到内存后，PE 装载器将处理 PE 文件中 import table 的逻辑部分．

PE 文件的有效性检查过程：(1)首先检验文件头部第一个字的值是否等于 IMAGE_DOS_SIGNATURE，是则 MS-DOS Header 有效； (2)当证明文件的 MS-DOS Header 有效后，用 e_lfanew 来定位 PE Header ； (3)比较 PE Header 的第一个字的值是否等于 IMAGE_NT_HEADER．如果前后两个值都匹配，则认为该文件是一个有效的 PE 文件．

2　PE 文件隐藏信息的原理

PE 文件通常是用高级语言编写，每个 section 存在大量的冗余空间，在不影响程序执行的基础上可以把用户要隐藏的信息存放到程序空闲的物理空间上．从分析图 1 的段表结构可以发现 PE 文件的头部和段内都有空闲空间，文件头部考虑到扩充的需要，除了 MS-

DOS Header 到 Section Table 之间是有意义的结构外，一般文件在头部后面剩下大量的空闲空间，大部分文件的头部大小是 1K，也有些文件的头部超过 1K，如本文所作的实验分析时采用的“cliconfg.exe”文件头部就是 4K. 实际上有些计算机病毒就是利用 PE 文件这部分头部空间进行隐藏病毒代码的，如 CIH 病毒就是如此. 在段结构中，PointerToRawData 是段的段首地址， SizeOfRawData 是物理长度， VirtualSize 是实际长度， SizeOf RawData-VirtualSize 是用户可利用的空间大小， PointerToRawData+VirtualSize 就是段空闲空间的起始偏移地址. 通过这些参数，用户就可以进行隐藏信息内容，PE 文件的头部结构和段结构的具体构成见图 2.

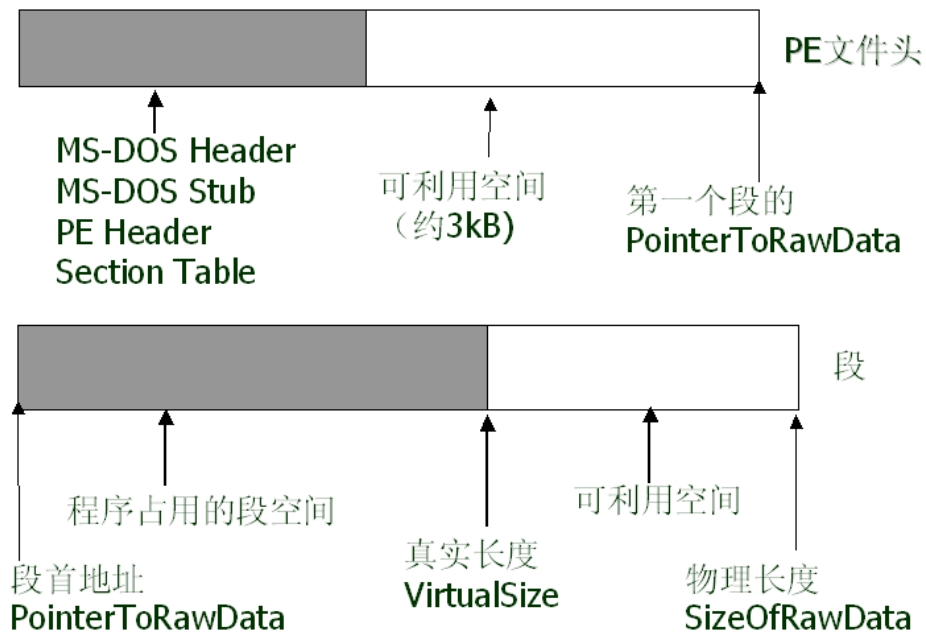

图2 PE文件头和段的可用空间

PE 文件头结构和含义在 WINNT.H 文件中的定义如下：

```
typedef struct _IMAGE_NT_HEADERS {
    DWORD Signature;  //PE 文件头标志，在 MS-DOS Header 的偏移 3CH 处所指向的地址开始
    IMAGE_FILE_HEADER FileHeader; //PE 文件物理分布的信息
    IMAGE_OPTIONAL_HEADER OptionalHeader; //PE 文件逻辑分布的信息
        } IMAGE_NT_HEADERS32， *PIMAGE_NT_HEADERS32;

typedef struct _IMAGE_FILE_HEADER {
    WORD    Machine; //该文件运行所需要的 CPU，对于 Intel 平台是 14Ch
    WORD    NumberOfSections; //文件的段数目
    DWORD   TimeDateStamp; //文件创建日期和时间
    DWORD   PointerToSymbolTable; //用于调试
    DWORD   NumberOfSymbols; //符号表中符号个数
    WORD    SizeOfOptionalHeader; //OptionalHeader 结构大小
    WORD    Characteristics; //文件信息标记，区分文件是 exe 还是 dll
        } IMAGE_FILE_HEADER， *PIMAGE_FILE_HEADER;
    typedef struct _IMAGE_OPTIONAL_HEADER {
        WORD    Magic; //标志字(总是 010bH)
        BYTE    MajorLinkerVersion; //连接器版本号
        BYTE    MinorLinkerVersion;
        DWORD   SizeOfCode; //代码段大小
        DWORD   SizeOfInitializedData; //已初始化数据块大小
        DWORD   SizeOfUninitializedData; //未初始化数据块大小
        DWORD   AddressOfEntryPoint; //PE装载器准备运行的 PE 文件的第一个指令的 RVA，若要改变整个执行的流程，可以将该值指定到新的 RVA，这样新 RVA 处的指令首先被执行.
        DWORD   BaseOfCode; //代码段起始 RVA (Relative Virtual Address)
        DWORD   BaseOfData; //数据段起始 RVA
        DWORD   ImageBase; //PE 文件的装载地址
        DWORD   SectionAlignment; //段对齐
        DWORD   FileAlignment; //文件对齐
        WORD    MajorOperatingSystemVersion;//所需操作系统版本号
        WORD    MinorOperatingSystemVersion;
        WORD    MajorImageVersion; //用户自定义版本号
        WORD    MinorImageVersion; //
```

```
    WORD    MajorSubsystemVersion; //win32 子系统版本．若 PE 文件是专门为 Win32 设计的
    WORD    MinorSubsystemVersion; //该子系统版本必定是 4.0 否则对话框不会有 3 维立体感
    DWORD   Win32VersionValue; //保留
    DWORD   SizeOfImage; //内存中整个 PE 映像体的尺寸
    DWORD   SizeOfHeaders; //所有头+段表的大小
    DWORD   CheckSum; //校验和
    WORD    Subsystem; //NT 用来识别 PE 文件属于哪个子系统
    WORD    DllCharacteristics;
    DWORD   SizeOfStackReserve;
    DWORD   SizeOfStackCommit;
    DWORD   SizeOfHeapReserve;
    DWORD   SizeOfHeapCommit;
    DWORD   LoaderFlags;
    DWORD   NumberOfRvaAndSizes;
    IMAGE_DATA_DIRECTORY          DataDirectory[IMAGE_NUMBEROF_DIRECTORY_ENTRIES];
//IMAGE_DATA_DIRECTORY 结构数组．每个结构给出一个重要数据结构的 RVA，比如引入地址表等
     } IMAGE_OPTIONAL_HEADER32， *PIMAGE_OPTIONAL_HEADER32;

  typedef struct _IMAGE_DATA_DIRECTORY {
    DWORD   VirtualAddress; // RVA 地址
    DWORD   Size; //大小
    } IMAGE_DATA_DIRECTORY， *PIMAGE_DATA_DIRECTORY;
```

PE 文件头后是段表，WINNT.H 中段表定义如下：

```
  typedef struct _IMAGE_SECTION_HEADER {
    BYTE    Name[IMAGE_SIZEOF_SHORT_NAME];//段表名称，如“.text”
    union {
      DWORD   PhysicalAddress; //物理地址
      DWORD   VirtualSize; //真实长度
    } Misc;
    DWORD   VirtualAddress; //RVA
    DWORD   SizeOfRawData; //物理长度
    DWORD   PointerToRawData; //段基于文件的偏移量
    DWORD   PointerToRelocations; //重定位的偏移
    DWORD   PointerToLinenumbers; //行号表的偏移
    WORD    NumberOfRelocations; //重定位项数目
    WORD    NumberOfLinenumbers; //行号表的数目
    DWORD   Characteristics; //段属性
    } IMAGE_SECTION_HEADER， *PIMAGE_SECTION_HEADER;
```

为了加强隐蔽性，防止探测程序清除隐藏在 PE 文件中的信息，如 PE Corrector[5] 本文在实现时把 VirtualSize 的值用 “VirtualSize+隐藏信息段的长度 ”替换，这种操作使 PE Corrector 无法探测到程序的改变. 另一方面，为了方便接收方还原隐藏的信息，在文件中加入一段特殊的字符串进行标识，这样在还原时根据该字符串能得到隐藏信息的起始位置．本文在实现方案中将用户输入的密码作为特殊字符串．这样不知道密码者就难以获得其中的隐藏信息.

3 方案实现

在基于 PE 文件实现信息隐藏方案的基础上，本文的结论是在 VC6.0 中进行模拟实现，程序的主要内容如下：

1)首先是获得 PE 文件头的信息，函数代码如下：

```
IMAGE_DOS_HEADER dos_head;
   DWORD signature;
   IMAGE_FILE_HEADER _head;
   IMAGE_OPTIONAL_HEADER opt_head;

Void getpeheader(CString filename)
 {CFile cf;
   cf.Open (filename，CFile::modeRead);
   cf.ReadHuge(&dos_head，sizeof(IMAGE_DOS_HEADER));
```

```
        cf.Seek(dos_head.e_lfanew，0);
        cf.ReadHuge(&signature，sizeof(DWORD));
        cf.ReadHuge(&_head，sizeof(IMAGE_FILE_HEADER));
        cf.ReadHuge(&opt_head，sizeof(IMAGE_OPTIONAL_HEADER));
        cf.Close();
       }
```

其中 dos_head 就是 MS-DOS Header，cf.Seek(dos_head.e_lfanew，0)的作用是跳过 DOS Stub，signature 是 PE 文件的标志，opt_head 是 PE Header.

2)获得段信息

段末尾空闲空间就是我们寻找的空间，_head.NumberOfSections 是 PE 文件中的总段数，定义 IMAGE_SECTION_HEADER section_header[20]是为了获得各段信息 . 代码如下：

```
Void getsection(Cstring filename)
   {
     CFile cf;
     cf.Open (filename，CFile::modeRead);
      long secpos= sizeof(IMAGE_DOS_HEADER)+dos_head.e_lfanew+ sizeof(DWORD)
        +sizeof(IMAGE_FILE_HEADER) +sizeof(IMAGE_OPTIONAL_HEADER);
     cf.Seek(secpos，0);
     for (int I=0 ;I<_head.NumberOfSections;I++)
       cf.ReadHuge(&section_header[I]，sizeof(IMAGE_OPTIONAL_HEADER));
      cf.Close();
   }
```

3)写文件，实现信息隐藏.

在前两步的基础上，找到了空闲区域，下面将要隐藏的信息写入到 PE 文件的空闲区域，函数如下：

```
Void  writefile(Cstring filename，int section)
  { CFile cf;
    cf.Open (filename，CFile::modeWrite);
    if((section_header[section].SizeOfRawData- section_header[section].Misc.VirtualSize)<=0)
       { cf.Close();
           return;
        }
    cf.Seek(section_header[section].PointerToRawData，0);
    long I=0;
    chae ch;
   while(I< section_header[section].SizeOfRawData -section_header[section].Misc.VirtualSize)
    { if((ch=mygetch()))cf.WriteHuge(&ch，1);
        I++; }
   cf.Close();
  }
```

程序执行过程中先检查段空间是否小于 0，然后调用函数 mygetch()获得已处理好的待隐藏资料.然后把资料写入可执行程序文件中.

4）对隐藏信息的安全性保护

信息隐藏是以二进制代码的行式进行操作，为了在隐藏消息前对消息进行加密，本文实现中采用 rijndael 算法对隐藏的消息进行加解密处理. 加解密处理所用的函数算法如下：

```
rijndael::set_key(const u1byte in_key[]， const u4byte key_len)     //设置密码
rijndael::encrypt(const u1byte in_blk[16]， u1byte out_blk[16])     //实现 128bits(16 个字节)加密
rijndael:: decrypt(const u1byte in_blk[16]， u1byte out_blk[16])   //对 128bits 解密
```

为了防止攻击者或其它程序对隐藏信息进行窜改，需要对隐藏的信息进行完整性保护. 本方案在实现时首先对每一字节采用一位的奇/偶校验位．这样每 8 字节就在后边加一字节的用于校验．处理完后再构造一函数把原来的排列顺序致乱， 这个混淆函数要根据密码进行处理. 实现方案中采用约瑟夫环和口令密码实现．这样分块的密文消息通过起始地址的转换，就实现了对原来顺序的致乱操作．实现代码如下：

```
Char  ch;
Ch=getjojj(char * src);   //ch 为 src 的奇/偶校验字节，实现奇/偶校验
long newpos;
newpos=myhash(char * pwd，long oldpos);
```

4 结果分析

根据文献[6]， PE文件平均可利用的空闲空间是2325个年字节， 在一个可执行文件(.exe或.dll)中平均可供利用的最大空间是3347字节．本文选取Windows XP下的一个系统文件“cliconfg.exe”进行分析，具体的执行效果如图3所示：

图 3　基于 PE 文件的信息隐藏实现界面

图 3 的实验过程如下：在图 3 的源文件处通过浏览功能打开 PE 文件“cliconfg.exe”，程序自动分析出该文件的结构是：文件头实际大小 648 字节，由于该文件头部结构是 4K，因此文件头可用自由空间是 3447 字节，PE 文件“cliconfg.exe”共有 4 个段，第 0 号段的段类型是 text，段物理偏移为 0x1000 开始，可用空间 2664 字节；第 1 号段类型是 rdata，段的物理偏移量是 0x6000 ，可用空间为 1566 字节，第 2 段类型是 data，段的物理偏移是为 0x7000 ，可用空间是 1380 字节；第 3 段类型为 rsrc，段的物理偏移量是 0xA000，可用空间为 1328 字节；总可用空间是 10385 字节．文件“ cliconfg.exe”的空间空闲率为 10.3/44.5=23.15%．

由于文件“cliconfg.exe”的空闲空间是 10.3K，本文选取了 Windows XP 下的《最终用户许可协议》（LICENSE.TXT,10K）作为实验用的隐藏信息，在图 3 中的“导入文件”按钮将隐藏文件进行加密，并输入用户提取信息的口令密码，利用口令密码从空闲空间中计算出一个存放信息的偏移量，通过“写入”操作将经过加密后的密文文件隐藏到 PE 文件中．通过图 3 中的“执行”按钮执行“cliconfg.exe”文件，结果与在 Windows XP 下执行效果相同，表明隐藏的文件不影响程序的正常执行．经过图 3 中的“导出文件”按钮，在正确输入密码后可以提取出被隐藏的信息，将其命名为“LICENSE12.TXT”．

图 4 和图 5 是通过 Grig Software 公司的“Compare It”[7]比较工具对实验结果进行分析的过程．图 4 是提取出来的隐藏信息文件（LICENSE12.TXT）与隐藏前的源文件（LICENSE.TXT）进行比较的界面，分析结果显示两文件完全相同．

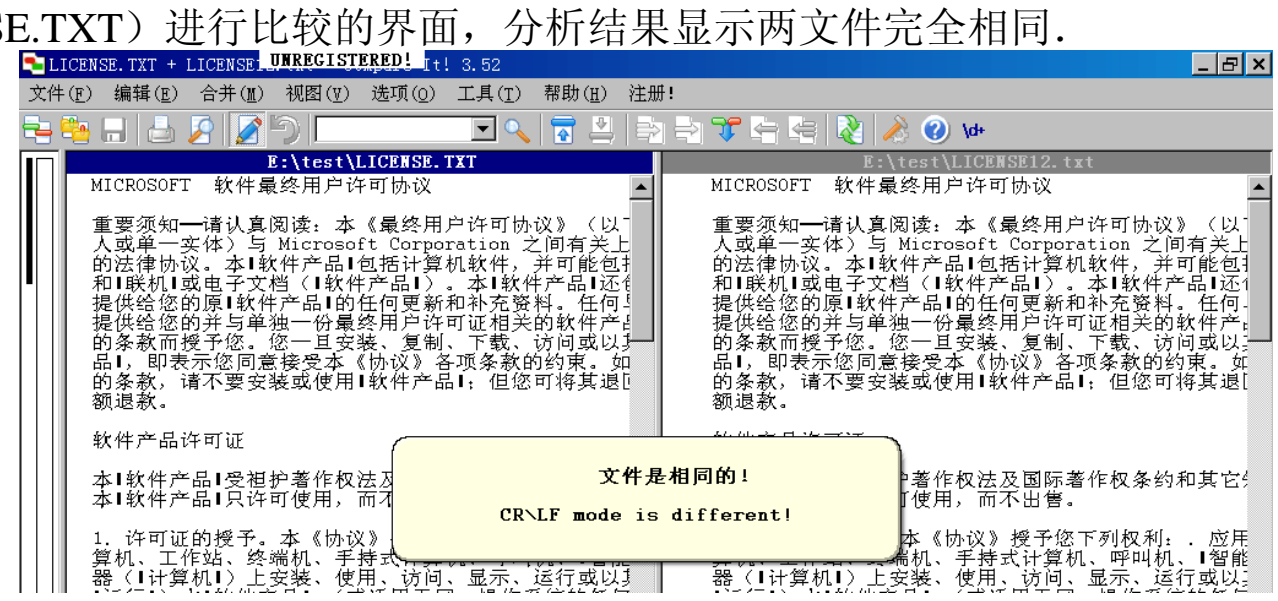

图 4　被隐藏的源文件与提取文件比较

图 5 是将掩护媒体文件“cliconfg.exe”在隐藏信息后与源文件（更名为“cliconfgbak.exe”）进行比较的界面，从分析工具可以看出有 231 行发生了改变． 由于在实际应用时，隐藏信息经过了加密处理，接收方可能只有一个隐藏有信息的伪装媒体，而没有源文件，所以也无法知道隐藏信息的内容，图 5 的左下方就是隐藏的密文．

经过分析可以看出，在 PE 文件中嵌入信息后，并不影响 PE 程序文件的执行，即隐藏信息并不影响用户对 PE 文件的感知特性；同时，基于 PE 文件的信息隐藏技术是利用

PE 文件的空闲空间，并不改变文件的大小，不影响 PE 文件的存贮空间和网络传输带宽，因此隐藏信息后的 PE 文件也不影响系统的感知性；另一方面， PE 文件作为可执行性文件，通常所作的变换只是无损压缩，这种操作对 PE 文件无任何影响，自然也就不会影响到隐藏的信息，即 PE 文件具有一定的鲁棒性. 由于 PE 文件作为掩护媒体完全满足文献[1]中的三个条件，因此本文认为 PE 文件也完全可以作为信息隐藏技术的掩护媒体.

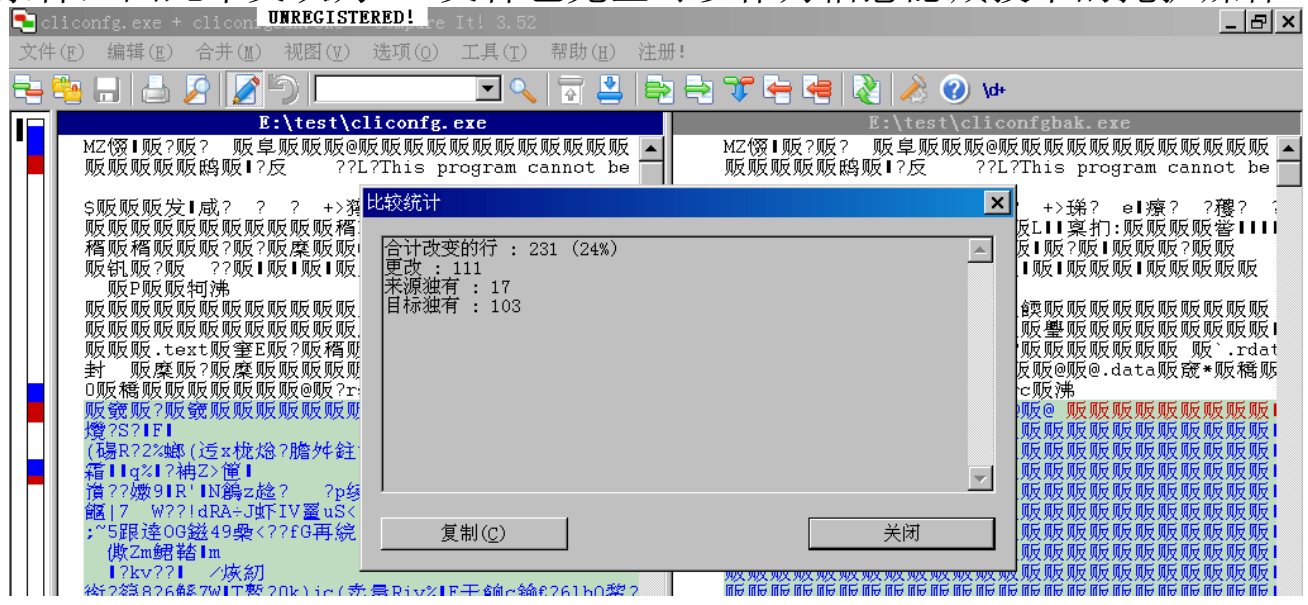

图 5　掩护媒体文件隐藏信息前后比较

现有信息隐藏技术主要是利用人们视觉和听觉冗余的原理进行隐藏秘密信息，本文提出的 PE 文件隐藏方法，是以二进制的形式对文件信息进行隐藏，因此可以隐藏的信息并不局限于本文分析时采用的文本文件，可以是任何形式的文件. PE 文件隐藏的信息是人们无法感觉到的，不影响程序的执行，不增加文件的空间，不同的用户间可以采用口令密码加密的方法，实现隐藏通信. 在不知道口令的情况下，攻击者无法检测和还原隐藏的信息. 同时本文所提方案实际上是实现了两重密码保护，一层是 rijndael 实现的对称密码加密，另一层是通过口令密码实现信息的移位和混淆，并且还进行了完整性保护. 因此本文所提的方案和算法具有通用性，具有较好的安全性. 采用密码控制进行信息隐藏和保密，而不是以算法进行保密，这也符合现代密码学的发展思想.

5　小结

本文提出了一种全新的基于 PE 文件的信息隐藏方法，该方法不仅可以实现用户间的隐蔽通信，还可以实现信息隐藏技术中的数字指纹和软件水印等，为计算机软件系统开发商保护自己的知识产权提供了一条新的途径，如软件开发商可以采用保存密钥的门限方法，将水印或数字指纹分成多块，分散存放到系统中的多个 PE 文件 (如 exe, dll, sys, bpl, dpl, cpl, ocx, acm, ax, scr) 之中，这样就避免了某个信息集中保存的不足，提高了隐藏信息的抗攻击和抗破坏能力，提高了隐藏信息的可生存性. 也可以利用 PE 文件的随机性特点，实现信息隐藏中的猝发式通信，由于这种通信具有随机性，攻击者难以进行有效地监测和破解，因此还可以采用 PE 文件的隐藏通信方式进行偶然性和不定期的密钥交换等.

PE 文件进行信息隐藏的不足是隐藏空间相对集中，是否存在新的问题还有待进一步研究．本文只是提出了一个思想，认为基于 PE 文件的信息隐藏方法是一种很有前景的技术，因此建立严格科学的数学理论模型、研究 Windows 和 Linux 不同环境下的文件隐藏方法、研究基于文件的信息隐藏技术的攻击方法等都将是未来重点研究的内容.

参考文献

[1] W.Bender， D.Gruhl， and N.Morimoto. Techniques for data hiding. Technical Report[R]， MIT Media Lab， 1994.

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

[3] 刘振华，尹萍，信息隐藏技术及其应用[M]，科学出版社，2002

[4] Microsoft Corporation， Microsoft Portable Executable and Common Object File Format Specification ， Revision 6.0[EB/OL] ， February 1999.

[5] GigaMind Systems，PE Corrector Overview[CP/OL], http://www.gigamindsystems.com/pec/pec.php

[6] Yinrong Huang ， Vulnerabilities in Portable Executable (PE) File Format For Win32 Architecture[EB/OL]， April 9， 2003， http://members.rogers.com/exurity/

[7] Grig Software, Compare It V3.5.2[CP/OL], http://www.grigsoft.com/